INTERNATIONAL CONFERENCE
RECENT ADVANCES
IN NATURAL LANGUAGE PROCESSING

RANLP 2019

Natural Language Processing in a Deep Learning World

PROCEEDINGS

Edited by Galia Angelova, Ruslan Mitkov, Ivelina Nikolova, Irina Temnikova

Varna, Bulgaria
2–4 September, 2019

INTERNATIONAL CONFERENCE
RECENT ADVANCES IN
NATURAL LANGUAGE PROCESSING 2019

Natural Language Processing
in a Deep Learning World

PROCEEDINGS

Varna, Bulgaria
2–4 September, 2019

Print ISBN 978-954-452-055-7
Online ISBN 978-954-452-056-4
Series Print ISSN 1313-8502
Series Online ISSN ISSN 2603-2813

Designed and Printed by INCOMA Ltd.
Shoumen, BULGARIA

Preface

Welcome to the 12th International Conference on “Recent Advances in Natural Language Processing” (RANLP 2019) in Varna, Bulgaria, 2-4 September 2019. The main objective of the conference is to give researchers the opportunity to present new results in Natural Language Processing (NLP) based on modern theories and methodologies.

The Conference is preceded by the First Summer school on Deep Learning in NLP (29-30 August 2019) and two days of tutorials (31 August – 1 September 2019).

The Summer School lectures are given by Kyunghyun Cho (New York University), Marek Rei (University of Cambridge), Tim Rocktäschel (University College London) and Hinrich Schütze (Ludwig Maximilian University, Munich). Training in practical sessions is provided by Heike Adel (University of Stuttgart), Alexander Popov (Institute of Information and Communication Technologies, Bulgarian Academy of Sciences), Omid Rohanian and Shiva Taslimipoor (University of Wolverhampton).

Tutorials are given by the following lecturers: Preslav Nakov (Qatar Computing Research Institute, HBKU), Valia Kordoni (Humboldt University, Berlin), Antonio Miceli Barone (University of Edinburgh) and Sheila Castilho (Dublin City University), Vlad Niculae and Tsvetomila Mihaylova (Institute of Telecommunications, Lisbon).

The conference keynote speakers are:

- Kyunghyun Cho (New York University),
- Ken Church (Baidu),
- Preslav Nakov (Qatar Computing Research Institute, HBKU),
- Sebastian Padó (Stuttgart University),
- Hinrich Schütze (Ludwig Maximilian University, Munich).

This year 18 regular papers, 37 short papers, 95 posters, and 7 demos have been accepted for presentation at the conference. The proceedings cover a wide variety of NLP topics, including but not limited to: deep learning; machine translation; opinion mining and sentiment analysis; semantics and discourse; named entity recognition; coreference resolution; corpus annotation; parsing and morphology; text summarisation and simplification; event extraction; fact checking and rumour analysis; NLP for healthcare; and NLP for social media.

In 2019 RANLP hosts four post-conference workshops on influential NLP topics: the 2nd Workshop on Human-Informed Translation and Interpreting Technology (HiT-IT 2019), the 12th Workshop on Building and Using Comparable Corpora (BUCC), the Multiling 2019 Workshop: Summarization Across Languages, Genres and Sources as well as an Workshop on Language Technology for Digital Historical Archives with a Special Focus on Central-, (South-)Eastern Europe, Middle East and North Africa.

We would like to thank all members of the Programme Committee and all additional reviewers. Together they have ensured that the best papers were included in the Proceedings and have provided invaluable comments for the authors.

Finally, special thanks go to the University of Wolverhampton, the Institute of Information and Communication Technologies at the Bulgarian Academy of Sciences, the Bulgarian National Science Fund, Ontotext and IRIS.AI for their generous support of RANLP.

Welcome to Varna and we hope that you enjoy the conference!

The RANLP 2019 Organisers

The International Conference RANLP–2019 is organised by:

Research Group in Computational Linguistics,
University of Wolverhampton, UK

Linguistic Modelling and Knowledge Processing Department,
Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences, Bulgaria

RANLP–2019 is partially supported by:

National Scientific Fund, Ministry of Education and Science, Bulgaria

Ontotext AD

IRIS.AI

Programme Committee Chair:

Ruslan Mitkov, University of Wolverhampton, UK

Organising Committee Chair:

Galia Angelova, Bulgarian Academy of Sciences, Bulgaria

Workshop Coordinator:

Kiril Simov, Bulgarian Academy of Sciences, Bulgaria

Tutorial Coordinator:

Preslav Nakov, Qatar Computing Research Institute, HBKU, Qatar

Proceedings Printing:

Nikolai Nikolov, INCOMA Ltd., Shoumen, Bulgaria

Programme Committee Coordinators:

Ivelina Nikolova, Bulgarian Academy of Sciences
Irina Temnikova, Bulgarian Academy of Sciences

Programme Committee:

Ahmed Abdelali (Hamad Bin Khalifa University, Qatar)
Cengiz Acarturk (Middle East Technical University, Turkey)
Guadalupe Aguado-de-Cea (Polytechnic University of Madrid, Spain)
Luis Alfonso Ureña López (University of Jaén, Spain)
Hassina Aliane (Research Center on Scientific and Technical Information, Algeria)
Pascal Amsili (University of Paris Diderot, France)
Galia Angelova (Bulgarian Academy of Sciences, Bulgaria)
Riza Batista-Navarro (University of Manchester, United Kingdom)
Kalina Bontcheva (University of Sheffield, United Kingdom)
Svetla Boytcheva (Bulgarian Academy of Sciences, Bulgaria)
António Branco (University of Lisbon, Portugal)
Chris Brew (Digital Operatives)
Nicoletta Calzolari (Italian National Research Council, Italy)
Sheila Castilho (Dublin City University, Ireland)
Key-Sun Choi (Korea Advanced Institute of Science and Technology, South Korea)
Kenneth Church (Baidu, United States of America)
Kevin Cohen (University of Colorado School of Medicine, United States of America)
Gloria Corpas Pastor (University of Málaga, Spain)
Dan Cristea (University of Lași, Romania)
Antonio Ferrández Rodríguez (University of Alicante, Spain)
Fumiyo Fukumoto (University of Yamanashi, Japan)
Prószéky Gábor (Pázmány Péter Catholic University & Bionics, Hungary)
Alexander Gelbukh (National Polytechnic Institute, Mexico)
Yota Georgakopoulou (Athena Consultancy, Greece)
Ralph Grishman (New York University, United States of America)
Veronique Hoste (Ghent University, Belgium)
Diana Inkpen (University of Ottawa, Canada)
Hitoshi Isahara (Toyohashi University of Technology, Japan)
Miloš Jakubíček (Lexical Computing Ltd)
Alma Kharrat (Microsoft)
Udo Kruschwitz (University of Essex, United Kingdom)
Sandra Kübler (Indiana University, United States of America)
Katia Lida Kermanidis (Ionian University, Greece)
Natalia Loukachevitch (Lomonosov Moscow State University, Russia)
Eid Mohamed (Doha Institute for Graduate Studies, Qatar)
Emad Mohamed (University of Wolverhampton, United Kingdom)
Johanna Monti (University of Naples L'Orientale, Italy)
Andrés Montoyo (University of Alicante, Spain)
Alessandro Moschitti (Amazon)
Rafael Muñoz Guillena (University of Alicante, Spain)
Preslav Nakov (Qatar Computing Research Institute, Qatar)
Roberto Navigli (Sapienza University of Rome, Italy)
Raheel Nawaz (Manchester Metropolitan University, United Kingdom)
Mark-Jan Nederhof (University of St Andrews, United Kingdom)
Ivelina Nikolova (Bulgarian Academy of Sciences, Bulgaria)
Kemal Oflazer (Carnegie Mellon University, Qatar)
Maciej Ogrodniczuk (Polish Academy of Sciences, Poland)

Constantin Orasan (University of Wolverhampton, United Kingdom)
Petya Osenova (Sofia University and Bulgarian Academy of Sciences, Bulgaria)
Sebastian Padó (Stuttgart University, Germany)
Noa P. Cruz Diaz (Artificial Intelligence Excellence Center, Bankia, Spain)
Liviu P. Dinu (University of Bucharest, Romania)
Pavel Pecina (Charles University, Czech Republic)
Stelios Piperidis (Athena Research Center, Greece)
Massimo Poesio (University of Essex, United Kingdom)
Horacio Rodríguez (Polytechnic University of Catalonia, Spain)
Paolo Rosso (Polytechnic University of Valencia, Spain)
Vasile Rus (The University of Memphis, United States of America)
Frédérique Segond (Viseo)
Kiril Simov (Bulgarian Academy of Sciences, Bulgaria)
Vilelmini Sosoni (Ionian University, Greece)
Keh-Yih Su (Institute of Information Science, Academia Sinica, Taiwan)
Stan Szpakowicz (University of Ottawa, Canada)
Hristo Tanev (European Commission, Belgium)
Shiva Taslimipoor (University of Wolverhampton, United Kingdom)
Irina Temnikova (Sofia University, Bulgaria)
Dan Tufiș (Romanian Academy of Sciences, Romania)
Aline Villavicencio (University of Essex, United Kingdom and Federal University of Rio Grande do Sul, Brazil)
Yorick Wilks (Florida Institute for Human and Machine Cognition, United States of America)
Mai Zaki (American University of Sharjah, United Arab Emirates)
Marcos Zampieri (University of Wolverhampton, United Kingdom)
Michael Zock (University of Aix-Marseille, France)

Reviewers:

Ahmed AbuRa'ed (University Pompeu Fabra, Spain)
Mattia A. Di Gangi (University of Trento, Italy)
Itziar Aldabe (University of País Vasco, Spain)
Ahmed Ali (Hamad Bin Khalifa University, Qatar)
Ahmed Amine Aliane (Research Center on Scientific and Technical Information, Algeria)
Le An Ha (University of Wolverhampton, United Kingdom)
Atefeh (Anna) Farzindar (University of Southern California, United States of America)
João António Rodrigues (University of Lisboa, Portugal)
Pepa Atanasova (University of Copenhagen, Denmark)
Mohammed Attia (George Washington University, United States of America)
Parnia Bahar (Aachen University, Germany)
Belahcene Bahloul (University of Khemis Miliana, Algeria)
Eduard Barbu (University of Tartu, Estonia)
Alberto Barrón-Cedeño (University of Bologna, Italy)
Leonor Becerra (Jean Monnet University, France)
Andrea Bellandi (National Research Council, Italy)
Fernando Benites (ZHAW School of Engineering, Switzerland)
Victoria Bobicev (Technical University of Moldova, Moldova)

Antonina Bondarenko (Lipetsk State Technical University, Russia)
Aurélien Bossard (University Paris 8, France)
Aljoscha Burchardt (German Research Centre for Artificial Intelligence, Germany)
Lindsay Bywood (University of Westminster, United Kingdom)
Ruket Cakici (Middle East Technical University, Turkey)
Iacer Calixto (University of Amsterdam, Netherlands and New York University, United States of America)
Pablo Calleja (Polytechnic University of Madrid, Spain)
Erik Cambria (Nanyang Technological University, Singapore)
Kai Cao (New York University, United States of America)
Thiago Castro Ferreira (Tilburg University, Netherlands)
Yue Chen (Queen Mary University of London, United Kingdom)
Mihaela Colhon (University of Craiova, Romania)
Daniel Dakota (Indiana University, United States of America)
Kareem Darwish (Hamad Bin Khalifa University, Qatar)
Orphee De Clercq (Ghent University, Belgium)
Kevin Deturck (Viseo)
Asma Djaidri (University of Science and Technology Houari Boumediene, Algeria)
Mazen Elagamy (Staffordshire University, United Kingdom)
Can Erten (University of York, United Kingdom)
Luis Espinosa Anke (Cardiff University, United Kingdom)
Kilian Evang (University of Düsseldorf, Germany)
Richard Evans (University of Wolverhampton, United Kingdom)
Stefan Evert (Friedrich–Alexander University, Germany)
Anna Feherova (University of Wolverhampton, United Kingdom)
Mariano Felice (University of Cambridge, United Kingdom)
Corina Forascu (The Alexandru Ioan Cuza University, Romania)
Vasiliki Foufi (University of Geneva, Switzerland)
Thomas Francois (Université catholique de Louvain, Belgium)
Adam Funk (University of Sheffield, United Kingdom)
Björn Gambäck (Norwegian University of Science and Technology, Norway)
Aina Garí Soler (The Computer Science Laboratory for Mechanics and Engineering Sciences, France)
Federico Gaspari (Dublin City University, Ireland)
José G. C. de Souza (eBay)
Goran Glavaš (University of Mannheim, Germany)
Darina Gold (University of Duisburg-Essen, Germany)
Reshmi Gopalakrishna Pillai (University of Wolverhampton, United Kingdom)
Rohit Gupta (University of Wolverhampton, United Kingdom)
Amir Hazem (Nantes University, France)
Tomáš Hercig (University of West Bohemia, Czech Republic)
Yasser Hifny (University of Helwan, Egypt)
Diliara Iakubova (Kazan Federal University, Russia)
Adrian Iftene (The Alexandru Ioan Cuza University, Romania)
Camelia Ignat (European Commission, Belgium)
Dmitry Ilvovsky (National Research University Higher School of Economics, Russia)
Miloš Jakubíček (Masaryk University, Czech Republic)
Arkadiusz Janz (Wroclaw University of Science and Technology, Poland)

Héctor Jiménez-Salazar (The Metropolitan Autonomous University, Mexico)
Olga Kanishcheva (National Technical University, Ukraine)
Georgi Karadzhov (Sofia University, Bulgaria)
David Kauchak (Pomona College, United States of America)
Yasen Kiprov (Sofia University, Bulgaria)
Jan Kocoń (Wroclaw University of Science and Technology, Poland)
Sarah Kohail (Hamburg University, Germany)
Yannis Korkontzelos (Edge Hill University, United Kingdom)
Venelin Kovatchev (University of Barcelona, Spain)
Peter Krejzl (University of West Bohemia, Czech Republic)
Sudip Kumar Naskar (Jadavpur University, India)
Maria Kunilovskaya (University of Tyumen, Russia)
Andrey Kutuzov (University of Oslo, Norway)
Sobha Lalitha Devi (Anna University, India)
Gabriella Lapesa (Universität Stuttgart, Institut für Maschinelle Sprachverarbeitung, Germany)
Todor Lazarov (Bulgarian Academy of Sciences, Bulgaria)
Els Lefever (Ghent University, Belgium)
Ladislav Lenc (University of West Bohemia, Czech Republic)
Elena Lloret (University of Alicante)
Pintu Lohar (Dublin City University, Ireland)
Epida Loupaki (Aristotle University of Thessaloniki, Greece)
Lieve Macken (Ghent University, Belgium)
Mireille Makary (University of Wolverhampton, United Kingdom)
Michał Marcińczuk (Wroclaw University of Technology, Poland)
Angelo Mario Del Grosso (National Research Council of Italy, Italy)
Federico Martelli (Babelscape, Italy)
Patricia Martin Chozas (Polytechnic University of Madrid, Spain)
Eugenio Martínez-Cámara (University of Granada, Spain)
Irina Matveeva (NexLP, Unites States of America)
Flor Miriam Plaza del Arco (University of Jaén, Spain)
Arturo Montejo-Ráez (University of Jaén, Spain)
Paloma Moreda Pozo (University of Alicante, Spain)
Diego Moussallem (University of Paderborn, Germany)
Sara Moze (University of Wolverhampton, United Kingdom)
Nona Naderi (University of Toronto, Canada)
Marcin Oleksy (Wrocław University of Science and Technology, Poland)
Antoni Oliver (The Open University of Catalonia, Spain)
Mihaela Onofrei (University of Iasi, Romania)
Arzucan Özgür (Bogazici University, Tyrkey)
Santanu Pal (Saarland University, Germany)
Alexander Panchenko (University of Hamburg, Germany)
Sean Papay (University of Stuttgart, Germany)
Ljudmila Petković (University of Belgrade, Serbia)
Maciej Piasecki (Wroclaw University of Science and Technology, Poland)
Paul Piwek (The Open University, United Kingdom)
Alistair Plum (University of Wolverhampton, United Kingdom)
Alberto Poncelas (Dublin City University, Ireland)

Alexander Popov (Bulgarian Academy of Sciences, Bulgaria)
Maja Popović (Dublin City University, Ireland)
Dan Povey (Johns Hopkins University, United States of America)
Ondřej Pražák (University of West Bohemia, Czech Republic)
Prokopis Prokopidis (Research and Innovation Center in Information, Greece)
Tharindu Ranasinghe (University of Wolverhampton, United Kingdom)
Natalia Resende (Dublin City University, Ireland)
Pattabhi RK Rao (Anna University, India)
Omid Rohanian (University of Wolverhampton, United Kingdom)
Josef Ruppenhofer (Institute for the German Language, Germany)
Pavel Rychlý (Masaryk University, Czech Republic)
Magdaléna Rysová (Charles University, Czech Republic)
Branislava Šandrih (Belgrade University, Serbia)
Estela Saquete (University of Alicante, Spain)
Leah Schaede (Indiana University, United States)
Ineke Schuurman (University of Leuven, Belgium)
Olga Seminck (Paris Diderot University, France)
Nasredine Semmar (Laboratory for Integration of Systems and Technology, France)
Matthew Shardlow (Manchester Metropolitan University, United Kingdom)
Artem Shelmanov (Russian Academy of Sciences, Russia)
Dimitar Shterionov (Dublin City University, Ireland)
Jennifer Sikos (University of Stuttgart, Germany)
João Silva (University of Lisboa, Portugal)
Vasiliki Simaki (Lancaster University, United Kingdom)
Sunayana Sitaram (Microsoft Research, India)
Mihailo Skoric (Researcher, Serbia)
Felix Stahlberg (University of Cambridge, Department of Engineering, United Kingdom)
Kenneth Steimel (Indiana University, United States)
Sebastian Stüker (Karlsruhe Institute of Technology, Germany)
Yoshimi Suzuki (Shizuoka University, Japan)
Liling Tan (Nanyang Technological University, Singapore)
Segun Taofeek Aroyehun (National Polytechnic Institute, Mexico)
Laura Toloși (Self employed data scientist)
Elena Tutubalina (Kazan Federal University, Russia)
Eleni Tziafa (National and Kapodistrian Unversity of Athens, Greece)
Antonio Valerio Miceli Barone (University of Edinburgh, United Kingdom)
Mihaela Vela (Saarland University, Germany)
Cristina Vertan (University of Hamburg, Germany)
Manuel Vilares Ferro (University of Vigo, Spain)
Veronika Vincze (University of Szeged, Hungary)
Pidong Wang (National University of Singapore, Singapore)
Michael Wiegand (Heidelberg University, Germany)
Victoria Yaneva (University of Wolverhampton, United Kingdom)
Kristina Yordanova (University of Rostock, Germany)
Juntao Yu (Queen Mary University of London, United Kingdom)
Wajdi Zaghouani (Hamad Bin Khalifa University, Qatar)
Kalliopi Zervanou (Eindhoven University of Technology, Netherlands)
Inès Zribi (University of Sfax, Tunisia)

Table of Contents

Table Structure Recognition Based on Cell Relationship, a Bottom-Up Approach

Darshan Adiga **Shabir Bhat** **Muzaffar Shah** **Viveka Vyeth**
Datoin, Bengaluru, India
firstname.lastname@datoin.com

Abstract

In this paper, we present a relationship extraction based methodology for table structure recognition in PDF documents. The proposed deep learning-based method takes a bottom-up approach to table recognition in PDF documents. We outline the shortcomings of conventional approaches based on heuristics and machine learning-based top-down approaches. In this work, we explain how the task of table structure recognition can be modeled as a cell relationship extraction task and the importance of the bottom-up approach in recognizing the table cells. We use Multilayer Feedforward Neural Network for table structure recognition and compare the results of three feature sets. To gauge the performance of the proposed method, we prepared a training dataset using 250 tables in PDF documents, carefully selecting the table structures that are most commonly found in the documents. Our model achieves an overall accuracy of 97.95% and an F1-Score of 92.62% on the test dataset.

1 Introduction

Usage of digital documents have elevated drastically over the last two decades and a need for automatic information extraction from these documents has increased. Portable Document Format (PDF) has been introduced by Adobe in 1993. PDF documents are the most common format of digital documents and are extensively used in scientific research, finance, enterprises etc. As the production and usage of PDF documents have increased massively, substantial research work has focused on automating the methods for document analysis (Correa and Zander, 2017; Kavasidis et al., 2018).

Tabular data is a powerful way to represent the data, among other elements of a document like charts, images etc. Tables are found in a variety of classes of digital documents and are very useful to readers to capture, search and compare the facts, summarizations and draw conclusions. Automatically extracting the information from the tables and representing the information in more convenient formats for digital consumption add immense value in the field of document understanding (Gilani et al., 2017; Hao et al., 2016).

Tables contain structured data but often are rendered as semi-structured and unstructured on the digital documents for human consumption. Data can be represented using a variety of layouts in tables without losing the meaning of data (Anand et al., 2019). The layout of tables can vary in alignment, line and word spaces, column and row spans, borders and other styling information. Depending on the type of documents and authors, the tables may not contain any border lines and the structure of the tables will still be understandable to readers. The data represented by the tables in itself can have different semantics. For example, in a table, a column may contain a list of prices in dollars, indicating that all the values of that column contain numeric data only. Similar semantic information is embedded in the table rows as well. Further, a column may have multiple subcolumns, making the original column to span multiple table cells horizontally. In rare cases, rows can also span multiple table cells vertically. All these characteristics of a table make the automatic extraction of table information more challenging.

Table extraction is a sub-problem of document understanding, that deals with information extraction and representation of tabular data. Extraction of information from tables in documents has challenged the researchers over the last two decades. An ample amount of research work has been carried out leading to a diverse list of approaches including heuristics, rule-engine, and recently machine learning based proposals.

We believe, generalizing the patterns across the variety of table layouts in diverse type of documents, is best solved by machine learning approach. We propose a bottom-up approach for table structure recognition as a cell relation extraction task between the table text tokens using deep learning. The way a human understands the tables can be analogous to the proposed approach. Often, relation extraction task involves classification of an entity pair to a set of known relations, using documents containing mentions of the entity pair (Kumar, 2017). By considering the table recognition task as a relationship extraction problem, we introduce a novel approach suitable for several document understanding solutions.

The proposed method deals with the basic building blocks of any table, the table cells. With this approach, we hope to solve the column and row spanning, the presence or absence of borders, and other challenges mentioned earlier. The table recognition system operates at token-level and involves learning the complex patterns in order to extract the cell relationships among the table text tokens using deep learning.

2 Related Work

According to the well-known ICDAR 2013 Table Competition (Gbel et al., 2013), the problem of table understanding can be split into table location detection, table structure recognition, and table interpretation. Each of these sub-problems has attracted a great deal of attention from researchers and has extensive work.

A peek at the literature shows that many heuristic solutions have been proposed for table structure recognition. Most of those work consider the white space and layout analysis. Yildiz et al. (2005) propose an algorithm to recognize the columns of a table using distances between lines and then identify the cells to find the rows. The algorithm makes a few assumptions about the structure of the tables. The work of Krüpl and Herzog (2006) takes a bottom-up approach towards structure recognition using heuristics but works on browser-rendered documents. The methodology aggregates words into columns by considering the spatial distance of neighboring words.

Klampfl et al. (2014) experimented with two unsupervised approaches for table recognition and showcased the importance of spatial distances between words of a table using vertical and horizontal histogram projection of words coordinates.

Experiments using rule-engine has been proposed by Shigarov (2015), considers the physical layout of a rendered table, and the logical layout representing the relationships between the elements of a table, differently. Another work of Shigarov et al. (2016) shows promising results in recognizing the columns and rows of tables by using the word and line distances, the order of appearance of text chunks. The methodology makes use of configurable thresholds in its heuristic decision making.

The heuristic and rule-based solutions make various assumptions on the visual, type and content, structural details of tables and the thresholds used in the algorithms. These assumptions may not hold on heterogeneous documents and may even break the system.

Perez-Arriaga et al. (2016) have made use of both k-nearest neighbor and layout heuristics, making it a hybrid methodology to recognize the table structure. The method groups the words into rows and columns using spatial distances of words heuristically. Interestingly, the spatial distance thresholds are learned using the k-nearest neighbor algorithm. Their work also proposes a heuristic method to identify the headers of the table. Deep learning based semantic segmentation has been used by Schreiber et al. (2017) where an image of a document is fed to the neural network to identify the rows and columns of a table. However, the work makes use of a heuristic post-processing step to improve the table structure recognition.

Clinchant et al. (2018) have made an extensive comparison of three different Machine Learning approaches to recognize the table structure in hand-written register books. The method first recognizes the cell locations and then groups the cells into rows. The experimentations do a thorough comparison of CRF, a variation of Graph-CN called Edge-CN, and conventional Logistic Regression algorithms. However, the method works on already recognized headers and columns of the table and addresses only row recognition task.

To the best of our knowledge, most of the related work of table recognition try to identify the columns and rows of tables first and then locate the intersections of rows and columns as table cells. A few heuristics based works have considered the grouping of words into blocks and then aggregating blocks into rows and columns. A common downside of these methodologies is that they fail to capture the information about rows and columns spanning multiple table cells. Few of the heuristic approaches do try to solve this issue however, they fail to generalize the solution.

We propose a purely bottom-up approach by building the table structure by recognizing the individual cells of the table and their location in the document. The task of recognizing the table cells is addressed as cell-relation extraction between the tokens present in the table.

3 Methodology

In this section, we first explain how we modeled the table structure recognition as a relation extraction task, then the training data preparation, and finally describe how the binary relationship classification is modeled using a Multilayer Feedforward Neural Network.

3.1 Cell Relationship Extraction in Table Structure Recognition

Humans will recognize the table structure even without a need for borders, based on visual clues, spatial distances and the content of the cells. These visual clues present in the tables help the readers to recognize the location of table cells easily, by bringing all the words of a cell together both visually and semantically. The proposed method is based on this idea of identification of cell relationship among the table words. The first step towards the recognition of rows or columns is the identification of table cells and thus the whole process of table structure recognition is a bottom-up process. This reasoning is based on the underlying definition of any table: Unit of a table is a cell, horizontal and vertical alignment of cells forms rows and columns, respectively. Tokens are generated by using white-space and new-line characters as delimiters. In this paper the terms *token* and *word* are used interchangeably.

Relation extraction is a well-known task in Natural Language Processing, which deals with classifying whether a given set of *n* samples have any of *m* different relationships. For example, in linguistics, determining whether two or more expressions in a text refer to the same person or thing, is a relation extraction task. We take the idea of relation extraction and formulate the task of table structure recognition as identifying the cell-relationship among all the content of individual cells of a table. When the tokens that are part of a table are considered as the smallest possible elements of a table, the relation extraction task will be to identify whether given two tokens of table text, belong to the same cell or not. If two tokens belong to the same cell, then those two tokens have a *belong-to-same-cell* relation. In our experiments, this task of binary relationship extraction is considered as a binary classification problem.

For every pair of tokens, the goal of binary relationship classifier is to determine whether the two tokens belong to the same cell or not. An important thing to note here is that this relationship between the two tokens is transitive. If a token A is related to the token B and the token B is related to the token C, then the token A is related to C. Hence, we don't need to generate feature vectors of all the possible pairs of tokens in a cell to determine all the tokens of a cell, we only need to make sure that all the tokens of a cell are connected via a chain of such transitive dependencies.

Once we predict the *belong-to-same-cell* relation between token pairs, we group all the table tokens into different cells. This is a simple task of aggregating different token pairs into their respective cells. Using this data of all the table cells and the tokens in each of the cells, we can model the recognition of rows and columns of the table again as a relation extraction task between the pairs of table cells themselves. However, in this work, we concentrate only on cell recognition.

3.2 Data Preparation

Detecting Tables and Training Data Generation

The detection of the location of tables in PDF documents is the first task in the process of table extraction and this location information is prerequisite to our system. There are several open source and free of charge tools for detecting table locations in PDF documents. In our experiments, we used *Tabula* to obtain the location details of a table in the document. The location of a table is represented by five values, *pageNum*, *(startX, startY)* and *(endX, endY)*. All the coordinates are assigned

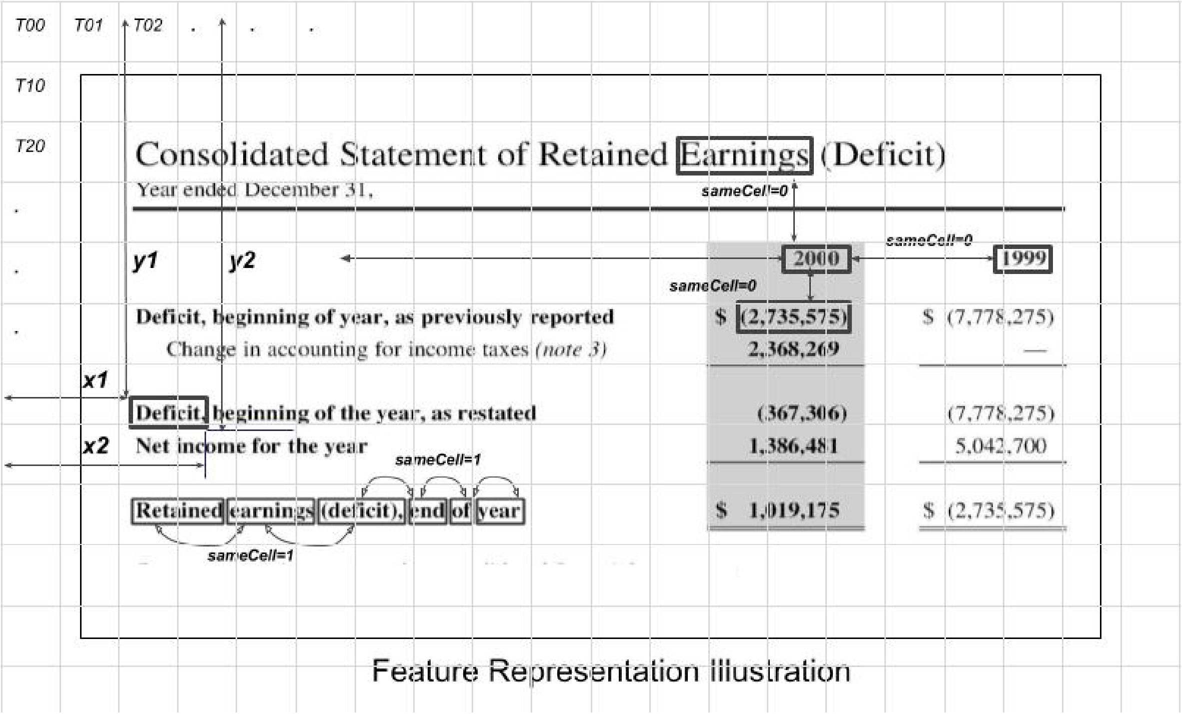

T00
T01
T02
T10
T20
Consolidated Statement of Retained Earnings (Deficit)
Year ended December 31,
sameCell=0
y1
y2
2000
sameCell=0
1999
sameCell=0
Deficit, beginning of year, as previously reported
$ (2,735,575)
$ (7,778,275)
Change in accounting for income taxes (note 3)
2,368,269
—
x1
Deficit, beginning of the year, as restated
(367,306)
(7,778,275)
x2
Net income for the year
1,386,481
5,042,700
sameCell=1
Retained earnings (deficit), end of year
$ 1,019,175
$ (2,735,575)
sameCell=1
Feature Representation Illustration

Figure 1: Features used in cell-relation extraction. A) T00, T01, T02,... are the tiles in the first row of tile matrix. B) T00, T10, T20,... are the tiles in the first column of tile matrix. C) (x1,y1) and (x2,y2) are the start and end coordinates of a token. D) sameCell=1, Pair of tokens which have sameCell relationship. E) samceCell=0, Pair of tokens which do not have sameCell relationship.

with respect to a Cartesian plane centered at the top-left corner of the document page. We used an off-the-shelf library to get the content in a PDF document inside a given region, that, along with the coordinates of characters *(x,y)*, provides the font style for every character in the document. The characters along with their coordinates and font styles are further aggregated into tokens, by using the white-space and new-line characters as delimiters.

With the help of table location and the location of individual tokens in the document, only those tokens which are within the given table location are collected by comparing their corresponding coordinate values. Specifically, all the tokens, whose *x* coordinate is between *startX* and *endX* and whose *y* coordinate is between *startY* and *endY* are collected as table text tokens.

After collecting all the tokens from table text using the table coordinates, we generate the training data for the binary relationship classification. Training data requires a pair of tokens and a target label indicating whether or not those two tokens belong to the same cell or not. Once we have a list of all the tokens that are part of the table, for every token, we create a pair of current token with every token, which is located within an imaginary rectangular window around the current token. The size of this imaginary rectangular window will help us determine the number of pairs of tokens to generate.

Training sample is a vector of all the features of a pair of tokens as denoted by 1.

$$V = [W_1F_i, W_2F_i, sameCell] \qquad (1)$$

Where, W_1F_i are *n* features of first token, W_2F_i are *n* features of second token, and *sameCell* is the target class indicating *True* if the two tokens belong to the same cell, *False* otherwise (see Figure 1).

The target class, *sameCell* is captured using *Datoin's WYSIWYG annotation tool*, that allows to select a sequence of words on the PDF document and tag those words as a table cell. The training data is generated using the annotated PDF documents and the target label *sameCell* is assigned accordingly for all the pair of words.

Cell Relationship Features

For each token in the table, we generate a set of locational and visual features. Use of semantic features of tabular data along with the mentioned features of this work can be one of the future works with the intention of improving the accuracy of the system.

We group the features used in the relation learning task as four categories as below.

Location and Tile features(LTF). The absolute location of a token is important evidence to indicate that it is indeed part of the table. Along with the absolute location, a more generalized positional information of tokens relative to the document makes the contextualization and localization of tokens easier for a reader.

To capture these relative location and distance information, for each token in the table, we consider *(x,y)* of starting of the token, and *(x,y)* of the ending of the token in documents (see Figure 1). In order to incorporate contextual information about a token, we split the entire document page into an imaginary matrix of tiles of size *(n X m)* and for each tile, we assign a tile number. For each table token, based on its coordinates we find in which tile the token is located, and we include its tile number, the row, and columns of the tile as features of that token (see Figure 1).

Neighborhood features(NF). The position of the surrounding tokens of a given token indicates the relative position of a token to its neighbors and captures the empty spatial distance around a token. For a given table token, we find a list of n nearest tokens in all the four directions, left, right, top and bottom based on the neighboring tokens' spatial distances with respect to the current token (see Figure 1). The horizontal and vertical relative distances between these neighboring tokens are used as features. The *Location and Tile features* of neighboring tokens are also included as part of the given token's feature set. This feature ensures that there exists a chain of transitive dependency connecting all the tokens of a cell.

Clustering and Alignment features(CAF). A human reader makes use of the relative closeness and horizontal and vertical alignments of a given word, especially when a table is not completely bordered, to decide which cell the word belongs to. The proximity of a pair of tokens and the presence of neighboring tokens for each token in four different directions captures the information about the relative closeness.

Among all the neighborhood tokens, we identify whether a given token is nearer to the left neighbor or right neighbor. Similarly, we identify whether that token is nearer to the top neighbor or bottom neighbor (see Figure 1). We have used the absence of neighborhood tokens as a set of four categorical features as well, indicating whether or not a given token has left, right, top and bottom neighbor token.

Type and Style features(TSF). Another significant visual clue used by humans in determining whether two words belong to the same cell or not is the content and the styles used in the words. A binary feature representing whether a token is a number or not was used to capture the data similarity within a row or a column. For every pair of tokens, the comparison of font size and bold styles are used to indicate whether the two tokens have a similar font style or not. Use of semantic features of the content of words could be another important clue in differentiating the words into cells.

We find that Neighborhood, Clustering and Alignment features play a critical role in distinguishing the tokens that do not belong to the same cell. All of the feature generation techniques are based on the coordinates of each of the tokens and the coordinates of the table itself. The number of tiles and the number of neighboring tokens are the parameters which can be tuned to achieve better table structure recognition accuracy.

3.3 Relation Classification Using Multilayer Feedforward Neural Network

We have used a *Multilayer Feedforward Neural Network* to model the binary relationship classifier in the experiments. In order to learn the complex patterns that exist in the table layouts, and generalize these patterns we decided that deep learning is the right tool. Working at token-level, we have huge training data as well and deep neural networks seemed a right candidate for the task.

The generated training data is fed into the Multilayer Feedforward Neural Network that uses *relu* activation function in the hidden layers and a sigmoid activation function in the output layer. The models were trained using *Adam optimizer* and *Binary cross-entropy* loss function as defined in 2 (Zhang, 2019).

$$Loss = -\left[y\log(p) + (1-y)\log(1-p)\right] \quad (2)$$

Where y is a binary indicator of correct prediction of a sample, p is the predicted probability for a training sample.

Feature Parameter	Value
Number of left, right, top and bottom neighbor tokens	1
Window size for token pair generation	30 x 30 pixels
Number of tiles	20 tile rows x 20 tile columns

Table 1: Word-level feature generation parameters

The input feature vector of *N* dimension is fed into the network and the sigmoid output value is decoded as binary classes, 0 indicating that the two tokens do not belong to the same cell, 1 indicating that the two tokens belong to the same cell. In our experiments, the Multilayer Feedforward neural network has been built using *Keras* backed by *Tensorflow*, for quick experimentation and development.

4 Experiments and Results

4.1 Dataset and Evaluation Metrics

Due to the lack of publicly available datasets that suit our methodology, we prepared the training data on our own. The dataset used for the experiments contains a total of 250 PDF documents, having one table per document. We ensured that the tables present in our dataset represent the possible diverse type of tables that are most commonly used. Our dataset has tables with and without borders, with and without column headings, with column and row spans, with all types of text alignments, varying line, and word spacing, and font styles. All the PDF documents were annotated using *Datoin's WYSIWYG annotation tool*.

Using the parameters listed in Table 1, we created approximately 0.3 million training samples from all the tokens of 250 tables, containing 83 different features. Training samples are split by a 9 to 1 ratio for training and testing, keeping approximately 30,000 samples for testing.

	Training data	Test Data
True Class	60,000	9,000
False Class	2,10,000	21,000
Samples size	2,70,000	30,000

Table 2: Approximate distribution of target labels

The distribution of target labels in our training and testing dataset is shown in Table 2. The imbalance in the distribution of classes makes sense because for every token in the table, within an imaginary rectangular window around that token, the number of tokens that are in the same cell will be less than the number of tokens that are not in the same cell.

Measuring how many predicted cells are actual cells in a given table, would be a more explanatory metric for evaluation. However, if one token among all the tokens of a cell is wrongly predicted by the relationship classification model as belonging to a different cell, then measuring the correctness of this prediction at a cell-level would be challenging. So we decided to use the accuracy of the binary classification model itself as our evaluation metric. This token-level metric is simpler and straightforward.

4.2 Hyperparameters

We have experimented with the hyperparameters of the neural network architecture itself. Table 3 defines the set of hyperparameters used in our experiments. In terms of the number of weights, *Set-1* is a simpler network with fewer weights to learn and *Set-2* is a more complex network.

Hyperparameter	Set-1	Set-2
Number of layers	4	5
Number of Epochs	200	300
Batch size	300	100
Learning Rate	0.001	0.001

Table 3: Hyperparameters used in Multilayer Feedforward Neural Network

Feature Set	Features
LTF	Location and Tile features
NF_CAF	Neighborhood features, Clustering and Alignment features
TSF	Type and Style features

Table 4: Feature sets used in the experiments

It is important to note here that a smaller batch size and a higher number of epochs do increase the F1-Scores and help the model to learn more com-

Feature Set	Class	Precision	Recall	F1-Score
LTF	True	77.89%	91.33%	83.51%
	False	95.13%	85.91%	90.29%
LTF & NF_CAF	True	93.65%	89.50%	91.07%
	False	94.07%	96.10%	95.73%
LTF, NF_CAF & TSF	**True**	**94.10%**	**90.85%**	**92.62%**
	False	**97.56%**	**98.27%**	**98.15%**

Table 5: Results of binary relation classification using Hyperparameter Set-2

plex patterns in the data, at the cost of increased training time.

4.3 Experiments

We have experimented with many combinations of feature sets and feature generation parameters and selected the best three sets of features, as listed in Table 4. The *Neighborhood features* and *Clustering and Alignment features* are combined into one set because both the features measure the togetherness of two given table tokens.

In each of these experiments, we have further experimented with 2 sets of neural network hyperparameters listed in Table 3.

4.4 Results and Discussion

All the experiments with the two sets of neural network hyperparameters listed in Table 3, indicated that *Set-2* outperforms the *Set-1*. So, we have listed only the results of experiments carried out using hyperparameters *Set-2* in Table 5.

Multiple experiments indicated that neighborhood features have sufficient information to capture the table structure and the use of visual clues does increase accuracy. However, one experiment showed that increasing the number of neighboring tokens for each token, reduces the *Recall* measure of *True* class. Increased number of *False* classes could be a possible explanation for this behavior. Also, increasing the number of hidden layers or hidden units of the network did not improve the accuracy further.

The model achieved an overall accuracy of *97.95%* on the test set after training the network for about an hour. Clearly, the model is predicting the cell-relationships on unseen token pairs with very high accuracy. A set of 20 documents containing a variety of tables, which are not part of training documents, are considered as a validation set.

The F1-Score of *False* class is much better than that of *True* class. One possible reason for this could be, the tokens that are not likely to be in the same cell will clearly have a distinguishable set of locational and neighboring features. It is clear that the recall of *True* class is causing the F1-Score to be low. Our training dataset has comparably fewer training samples for the *True* class this could be a possible reason for the low recall scores.

Manual verification of individual table cells with the prediction of the relation classifier shows that the model is able to generalize the cell recognition task across a variety of table cells. The cell relationships are identified accurately irrespective of the presence of borders lines, column, and row spans and text alignments. The relationship among the tokens of a table is learned by the model based on Neighborhood, Clustering and Alignment features of the tokens. However, for a few tokens where the neighborhood features do not have a clear separation with tokens from nearby cells, the model combines the tokens from adjacent cells, producing wrong predictions. Because of the absence of visual separation among the tokens of two closely aligned cells, the model predicts those multiple cells as a single cell.

5 Conclusion and Future Work

By applying the idea of relation extraction in table structure recognition task, we have shown the possibility of high accuracy information extraction in unstructured documents. Table structure recognition as relation extraction task is a novel approach in table extraction process and to the best of our knowledge has never been explored. We have taken the first step towards this direction and have proved that a bottom-up approach of cell relationship extraction is the right way towards table structure recognition task. We have compared three sets of features and showcased the significance of cognitive features in our experiments.

For a few of the tables, closely aligned adjacent cells are wrongly identified as one cell. Incorporating the semantic features of the content of the words, especially using Natural Language Processing, will enrich the feature vector and should help the model to do better generalizations. Exploring the different layout and visual features and improving the accuracy of the proposed method could be one of the possible future works.

Building on top of cell-relationship recognition work, we hope to explore the table structure extraction further. The knowledge of table cells can be used to build up the rest of the table structures from bottom-up. We believe that the relation extraction methodologies apply to other document understanding tasks and we hope to explore them as well.

References

Rahul Anand, Hye-Young Paik, and Cheng Wang. 2019. Integrating and querying similar tables from pdf documents using deep learning. *CoRR* abs/1901.04672.

S. Clinchant, H. Djean, J. Meunier, E. M. Lang, and F. Kleber. 2018. Comparing machine learning approaches for table recognition in historical register books. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. pages 133–138. https://doi.org/10.1109/DAS.2018.44.

Andreiwid Correa and Pr-ola Zander. 2017. Unleashing tabular content to open data: A survey on pdf table extraction methods and tools. pages 54–63. https://doi.org/10.1145/3085228.3085278.

Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. 2017. Table detection using deep learning. https://doi.org/10.1109/ICDAR.2017.131.

M. Gbel, T. Hassan, E. Oro, and G. Orsi. 2013. Icdar 2013 table competition. In *2013 12th International Conference on Document Analysis and Recognition*. pages 1449–1453. https://doi.org/10.1109/ICDAR.2013.292.

L. Hao, L. Gao, X. Yi, and Z. Tang. 2016. A table detection method for pdf documents based on convolutional neural networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. pages 287–292. https://doi.org/10.1109/DAS.2016.23.

Isaak Kavasidis, Sergio Palazzo, Concetto Spampinato, Carmelo Pino, Daniela Giordano, Daniele Giuffrida, and Paolo Messina. 2018. A saliency-based convolutional neural network for table and chart detection in digitized documents. *CoRR* abs/1804.06236.

Stefan Klampfl, Kris Jack, and Roman Kern. 2014. A comparison of two unsupervised table recognition methods from digital scientific articles. *D-Lib Magazine* https://doi.org/10.1045/november14-klampfl.

Bernhard Krüpl and Marcus Herzog. 2006. Visually guided bottom-up table detection and segmentation in web documents. In *WWW*.

Shantanu Kumar. 2017. A survey of deep learning methods for relation extraction. *CoRR* abs/1705.03645.

Martha O. Perez-Arriaga, Trilce Estrada, and Soraya Abad-Mota. 2016. Tao: System for table detection and extraction from pdf documents. In *FLAIRS Conference*.

S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. volume 01, pages 1162–1167. https://doi.org/10.1109/ICDAR.2017.192.

Alexey Shigarov. 2015. Table understanding using a rule engine. *Expert Systems with Applications* 42:929937. https://doi.org/10.1016/j.eswa.2014.08.045.

Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. 2016. Configurable table structure recognition in untagged pdf documents. https://doi.org/10.1145/2960811.2967152.

Burcu Yildiz, Katharina Kaiser, and Silvia Miksch. 2005. pdf2table: A method to extract table information from pdf files. pages 1773–1785.

Jiawei Zhang. 2019. Gradient descent based optimization algorithms for deep learning models training. *CoRR* abs/1903.03614.

Identification of Good and Bad News on Twitter

Piush Aggarwal
University of Duisburg-Essen
piush.aggarwal@stud.uni-due.de

Ahmet Aker
University of Duisburg-Essen
a.aker@is.inf.uni-due.de

Abstract

Social media plays a great role in news dissemination which includes good and bad news. However, studies show that news, in general, has a significant impact on our mental stature and that this influence is more in bad news. An ideal situation would be that we have a tool that can help to filter out the type of news we do not want to consume. In this paper, we provide the basis for such a tool. In our work, we focus on Twitter. We release a manually annotated dataset containing 6,853 tweets from 5 different topical categories. Each tweet is annotated with good and bad labels. We also investigate various machine learning systems and features and evaluate their performance on the newly generated dataset. We also perform a comparative analysis with sentiments showing that sentiment alone is not enough to distinguish between good and bad news.

1 Introduction

Social media sites like Twitter, Facebook, Reddit, etc. have become a major source of information seeking. They provide chances to users to shout to the world in search of vanity, attention or just shameless self-promotion. There is a lot of personal discussions but at the same time, there is a base of useful knowledgeable content which is worthy enough to consider for the public interest. For example in Twitter, tweets may report about news related to recent events such as natural or man-made disasters, discoveries made, local or global election outcomes, health reports, financial updates, etc. In all cases, there are good and bad news scenarios.

Studies show that news, in general, has a significant impact on our mental stature (Johnston and Davey, 1997). However, it is also demonstrated that the influence of bad news is more significant than good news (Soroka, 2006; Baumeister et al., 2001) and that due to the natural negativity bias, as described by (Rozin and Royzman, 2001), humans may end up consuming more bad than good news. Since bad news travels faster than good news (Kamins et al., 1997; Hansen et al., 2011) the consumption may increase. This is a real threat to the society as according to medical doctors and, psychologists exposure to bad news may have severe and long-lasting negative effects for our well being and lead to stress, anxiety, and depression (Johnston and Davey, 1997). (Milgrom, 1981; BRAUN et al., 1995; Conrad et al., 2002; Soroka, 2006) describe crucial role of good and bad news on financial markets. For instance, bad news about unemployment is likely to affect stock markets and in turn, the overall economy (Boyd et al., 2005). Differentiating between good and bad news may help readers to combat this issue and a system that filters news based on the content may enable them to control the amount of bad news they are consuming.

The aim of this paper is to provide the basis to develop such a filtering system to help readers in their selection process. We focus on Twitter and aim to develop such a filtering system for tweets. On this respect the contributions of this work are:

- We introduce a new task, namely the distinction between good and bad news on Twitter.
- We provide the community with a new gold standard dataset containing 6,893 tweets. Each tweet is labeled either as good or bad. To the best of our knowledge, this is the first dataset containing tweets with good and bad

labels. The dataset is publicly accessible and can be used for further research[1].

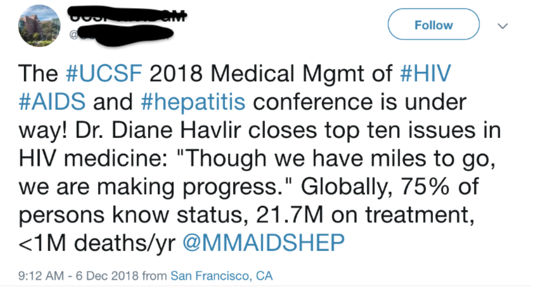
- Provide guidelines to annotate good/bad news on Twitter.
- We implement several features approaches and report their performances.
- The dataset covers diverse domains. We also show out-of-domain experiments and report system performances when they are trained on in-domain and tested on out-of-domain data.

In the following, we first discuss related work. In Section 3 we discuss the guidelines that we use to annotate tweets and gather our dataset. Section 4 provides description about the data itself. In Section 5 we describe several baseline systems performing the good and bad news classification as well as features used to guide the systems. Finally, we conclude the paper in Section 7.

2 Related Work

In terms of classifying tweets into the good and bad classes no prior work exists. The closest studies to our work, are those performing sentiment classification in Twitter (Nakov et al., 2016; Rosenthal et al., 2017). Kouloumpis et al. (2011) use n-gram, lexicon, part of speech and micro-blogging features for detecting sentiment in tweets. Similar features are used by Go (2009). More recently researchers also investigated deep learning strategies to tackle the tweet level sentiment problem (Severyn and Moschitti, 2015; Ren et al., 2016). Twitter is multi-lingual and in Mozetič et al. (2016) the idea of multi-lingual sentiment classification is investigated. The task, as well as approaches proposed for determining tweet level sentiment, are nicely summarized in the survey paper of Kharde et al. (2016). However, Balahur et al. (2010) reports that there is no link between good and bad news with positive and negative sentiment respectively.

Thus, unlike related work, we do tweet level good vs. bad news classification. We also show that similar to Balahur et al. (2010), there is no evidence that positive sentiment implies good news and negative sentiment bad news.

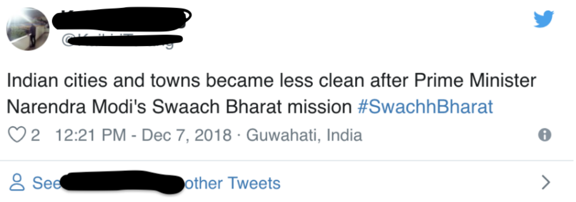


[redacted]

The #UCSF 2018 Medical Mgmt of #HIV #AIDS and #hepatitis conference is under way! Dr. Diane Havlir closes top ten issues in HIV medicine: "Though we have miles to go, we are making progress." Globally, 75% of persons know status, 21.7M on treatment, <1M deaths/yr @MMAIDSHEP

9:12 AM - 6 Dec 2018 from San Francisco, CA

Figure 1: Good news tweet



[redacted]

Indian cities and towns became less clean after Prime Minister Narendra Modi's Swaach Bharat mission #SwachhBharat

♡ 2 12:21 PM - Dec 7, 2018 · Guwahati, India

See [redacted] other Tweets

Figure 2: Bad news tweet

3 Good vs Bad News

News can be good for one section of society but bad for other section. For example, win or loss related news are always subjective. In such cases, agreement towards news types (good or bad) is quite low. On the other hand, news related to natural disaster, geographical changes, humanity, women empowerment, etc. show very high agreement. Therefore, while defining news types, topicality plays an important role.

We consider news as good news if it relates to low subjective topics and includes positive overtones such as recoveries, breakthroughs, cures, wins, and celebrations (Harcup and ONeill, 2017) and also beneficial for an individual, a group or society. An example of good news is shown in Figure 1. In contrary to that, the bad news is defined as when it relates to the low subjective topic and include negative overtones such as death, injury, defeat, loss and is not beneficial for an individual, a group or society. An example of bad news is shown in Figure 2. Based on these definitions/guidelines we have gathered our dataset (see next Section) of tweets containing the good and bad labels.

4 Dataset

Data collection To collect tweets for annotation, we first choose low subjective ten topics which can be divided into five different categories. Then,

[1] https://github.com/aggarwalpiush/goodBadNewsTweet

Category	**Topics**	**Collected**	**Annotated**
Health	Ebola	892	852
	Hiv	663	630
Natural Disaster	Hurricane Harvey	2,073	1,997
	Hurricane Irma	795	772
Terrorist Attack	Macerata oohmm	668	625
	Stockholm Attack	743	697
Geography and Env.	AGU17	652	592
	Swachh Bharat	21	21
Science and Edu.	IOT	627	602
	Nintendo	78	65
	Total	**7,212**	**6,853**

Table 1: Categories, their topics, and distributions for the dataset generation.

we retrieve the examples from Twitter using its API[2]. Next, we discard non-English tweets and re-tweets. We also remove duplicates based on lower-cased first four words of tweets keeping only the first one. Thereafter, we filter only those tweets which can be regarded as news by using an in house SVM classifier (Aggarwal, 2019). This classifier is trained on tweets annotated with the labels news and not news. We use this classifier to remove *not news* tweets from the annotation task[3]. We select only tweets where the classifier prediction probability is greater than or equal to 80%. In Table 1, we provide information about the topics and categories as well as statistics about the collected tweets that will be used for annotation (column *collected*).

Data Annotation For data annotation, we use the figure-eight crowdsourcing service[4]. Before uploading our collected examples, we carried out a round of trial annotation of 300 randomly selected instances from our tweet collection corpus. The aim of the trial annotation was

- to ensure the newsworthiness quality of our collected examples.
- to create test questions to ensure the quality of the annotators, for the rest of the data, which was carried out using crowdsourcing.
- to test our guidelines described in Section 3.

We ask three annotators[5] to classify the selected examples into good and bad news. We also allowed a third category *cannot say*. We computed Fleiss' kappa Fleiss (1971) on the trial dataset for the three annotators. The value is 0.605 which indicates rather a high agreement. We used 247 instances agreed by all the three annotators as test questions for the crowdsourcing platform.

During the crowd annotation, we showed each annotator 5 tweets per page and paid 3 US Cents per tweet. For maintaining quality standards, in addition to the test questions, we applied a restriction so that annotation could be performed only by people from English speaking countries. We also made sure that each annotation was performed maximum by 7 annotators and that an annotator agreement of min. 70% was met. Note if the agreement of 70% was met with fewer annotators then the system would not force an annotation to be done by 7 annotators but would finish earlier. The system requires 7 annotators if the minimum agreement requirement is not met. We only choose instances that are annotated by atleast 3 annotators. In addition to the good and bad news categories we also ask annotators to mandatory provide their confidence score (range between 0-100%) for the label they have annotated[6]. We discarded all the tweets where we did not have at least 3 annotators with each having min. 50% confidence value. We also discarded tweets that are annotated by less than three annotators. We

[2] https://www.tweepy.org

[3] Since we want humans to annotate tweets as good and bad news we apply this approach to filter tweets that are not news at all and so avoid our annotators spending valuable time on annotating tweets that are not our target.

[4] https://www.figure-eight.com/

[5] All are post-graduate students who are fluent in English and use Twitter to post information on a daily basis.

[6] We found this strategy better than providing the option *cannot say* and later allowed us to discard annotations where the confidence score was less than 50%.

use a total 7,212 tweets to annotate. After all filterings, we remained with 6,853 instances which were classified as good and bad news. Topic-wise number of successful annotations are displayed in the fourth column of Table 1.

Inter Annotator Agreement To calculate agreement between the annotators of the crowdsourcing annotation results, we select the top three confident annotator labels for each sample. Based on this, we record an agreement of 0.614 as Fleiss' Kappa (Fleiss, 1971) score indicating a good agreement among the annotators. We also claim stability in our annotation task because of the score similarity with that of trail annotation.

5 Method

We experiment with several machine learning approaches and features. Before using the tweets in decision making, we also apply a simple preprocessing on them. In the following, we briefly outline these.

5.1 Preprocessing

We use the ArkTokenizer (Gimpel et al., 2011) to tokenize the tweets. In addition to tokenization, we do lowercasing and remove digits if available in text.

5.2 Features

We extract nine features for each tweet and divide them into *Structural*, *TF-IDF* and *Embeddings* features.

5.2.1 Structural features

Emoticons: We extract all the emoticons from the training data and use them as a binary feature, i.e. does a tweet contain a particular emoticon or not.

Interjections: We use existing list of interjections[7] and use them similar to *Emoticons* as binary feature.

Lexicons: We use existing positive and negative lexicons[8] and use them as a binary feature.

Sentiment: We use the textblob[9] tool to compute sentiment score over each tweet. The score varies between -1 (negative) to 1 (positive).

POS-Tag: This feature includes 36 different pos-tags (uni-gram) and are used as binary features.

Significant terms: Using tf-idf values we also extract the top 300 terms (uni-gram and bi-gram, 300 in each case) from the training data and use them as binary features. Note, we extract for good and bad news separate uni-grams and bi-grams.

Tweet Characteristics: This feature contains tweet specific *characterstics* such as the number of favorite counts, tweet replies count and number of re-tweets.

5.2.2 TF-IDF

In this case, we simply use the training data to create a vocabulary of terms and use this vocabulary to extract features from each tweet. We use tf-idf representation for each vocabulary term.

5.2.3 Embeddings

Finally, we also use fasttext based embedding (Mikolov et al., 2018) vectors which are trained on common crawl having 600 billion tokens.

5.3 Classifiers

We investigate 8 classifiers for our task including Multi-Layer Perceptron (MLPC), Support Vector Machine with linear (LSVC) and rbf (SVC) kernel, K Nearest Neighbour (KNN), Logistic Regression (LR), Random Forest (RF), XGBoost (XGB) and Decision Tree (DT). In addition, we also fine-tune BERT-base model (Devlin et al., 2018). Each classifier, except the BERT, has been trained and tested on each possible combination of the three feature types.

6 Results

Overall results We performed a stratified 5-fold cross-validation. We evaluate each resulting model on a held-back development dataset containing 264 good news postings and 764 bad news ones. The 5-fold cross validation has been performed on the training data containing 4,332 bad news and 1,493 good news instances. For

[7] https://www.vidarholen.net/contents/interjections/

[8] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[9] https://textblob.readthedocs.io/en/dev/

Feature set	**SVC**	**XGB**	**LSVC**	**KNN**	**RF**	**MLPC**	**DT**	**LR**
Structural	.78	.78	.77	.77	.77	.63	.74	.78
Embeddings	.88	.86	.87	.86	.85	.85	.72	.87
TF-IDF	.86	.85	.86	.83	.84	.84	.83	.87
Structural + Embeddings	.86	.85	.87	.79	.86	.86	.78	.87
Structural + TF-IDF	.87	.87	.87	.80	.87	.86	.81	.87
Embeddings + TF-IDF	**.89**	.87	**.89**	.87	.88	.87	.81	**.89**
ALL	.88	.88	**.89**	.82	.87	.86	.82	.88
BERT-base model with its pre-trained embedding features: .92								

Table 2: F_1(macro) scores of different classifiers on different feature types evaluated on the test data. Multi-Layer Perceptron (MLPC), Support Vector Machine with linear (LSVC) and rbf (SVC) kernel, K Nearest Neighbour (KNN), Logistic Regression (LR), Random Forest (RF), XGBoost (XGB) and Decision Tree (DT).

each model, we use grid-search method to select the hyper-parameters with best model's efficiency. The results reported are those obtained on the test data and are summarized in Table 2. Overall we see that the performances of the classifiers are all highly satisfactory. Among the more traditional approaches, the best performance is obtained through SVC, LSVC, and LR. We see also that these approaches work best when embeddings along with tf-idf features are used, although LSVC achieves the same results when all features are used. However, the best performance is achieved with the BERT-base model leading to 92% F_1 score. We computed also significance test using paired t-test between BERT and more traditional machine learning approaches[10]. However, after Bonferroni correction ($p < 0.007$) we found no significant difference between BERT and the other systems.

Structural feature analysis We also evaluate the structural features of the task independently (Figure 3). For this, we use the SVC classifier as it is one of the best performing traditional methods. From the figure, we see that the significant term feature gives the best performance. The difference to the other features is greater than the 23% F_1 score. The differences are also significant after Bonferroni correction ($p < 0.008$). In Table 3 we list some frequent uni-grams from the significant good and bad term lists. From the table, we see that the terms are certainly good indicators for distinguishing between the two classes.

Bad news	**Good news**
fake	services
racism	cured
fox	resistant
attack	energy
migrants	support
fears	arrested

Table 3: Top uni-grams from the good and bad news significant term lists.

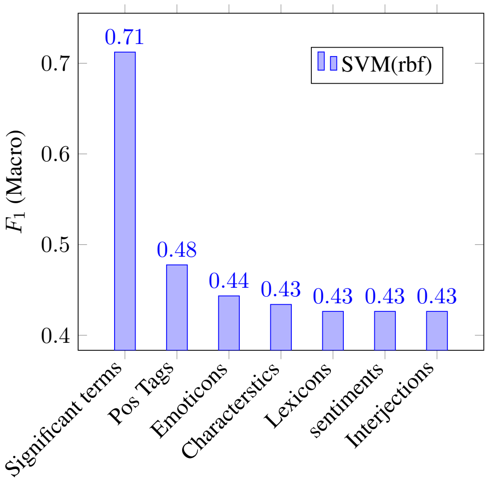


Figure 3: Structural features' performance using the SVM classifier evaluated on the test set.

[10] We always use the best result for every system.

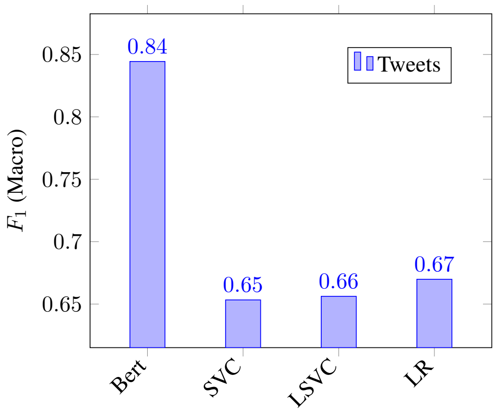


Figure 4: Out-of-domain performance of different systems.

Sentiment for good-vs-bad news We also tested whether sentiment score can predict good vs. bad news as Naveed et al. (2011) found a relationship between these two. For this, we use the textblob sentiment scorer and classify any tweet as good news when its sentiment score is greater than 0 otherwise bad. Using this strategy we could only achieve an F_1 score of 55%. This shows that tackling the good/bad news classification task using sentiment scores is not appropriate. This also confirms the findings of Balahur et al. (2010).

Out-of-domain experiments We also investigate how stable the models are when they are trained on in-domains and tested on out-of-domain data. For this purpose, we split our dataset into a training set consisting of all examples except instances belonging to the health category. We use four of the best-performing systems (BERT, SVC, LSVC, and LR) to train on this training set. The resulting models are tested on the held-out health data. Results are shown in Figure 4. From Figure 4 we see that BERT is stable and achieve an F_1 score of 84%. The performance of the other system drop by a great margin to the max. 67% F_1 score. From this, we can conclude that BERT is a better system to use for good-vs-bad Twitter news classification.

Detailed analysis on BERT Our overall but also the out-of-domain experiments show that BERT is outperforming the more traditional machine learning approaches. On the overall (1,028 testing instances) results, BERT fails only to classify 63 cases correctly. Using t-SNE distribution (van der Maaten and Hinton, 2008), we analyse BERT's 12th layer embedding vectors (having 300 dimensions) for random 100 test points (Figure 5). The analysis shows that BERT can classify semantics of good and bad news instances correctly even the instances are in proximity. From Figure 5, we see that mostly outliers are misclassified.

7 Conclusion and Future Work

In this paper, we presented a new dataset having 6,853 tweet post examples annotated with good and bad news labels. This dataset will be publicly available for the research community. We also presented a comparative analysis of supervised classification methods. We investigated nine different feature types and 8 different machine learning classifiers. The most robust result in our analysis was the contribution of the BERT-base model in in-domain but also in out-of-domain evaluations. Among structural features, significant terms significantly outperform the rest. We also showed that sentiment scores are not appropriate to classify good-vs-bad news.

In our future work, we plan to expand our investigation by including other features. We also plan to propose this model for the good-bad classification of news articles.

8 Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. ZE 915/7-1 "Data and Knowledge Processing (DKPro) - A middleware for language technology", by the Global Young Faculty[11] and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GRK 2167, Research Training Group "User-Centred Social Media".

References

Piush Aggarwal. 2019. Classification approaches to identify informative tweets. In *Proceedings of the Student Research Workshop Associated with RANLP 2019*. Varna, Bulgaria, page To be published.

Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik van der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. 2010. Sentiment analysis in the news.

Roy F. Baumeister, Ellen Bratslavsky, Catrin Finkenauer, and Kathleen D. Vohs. 2001. Bad is stronger than good. *Review of General Psychology* 5(4):323–370. https://doi.org/10.1037/1089-2680.5.4.323.

[11] https://www.global-young-faculty.de/

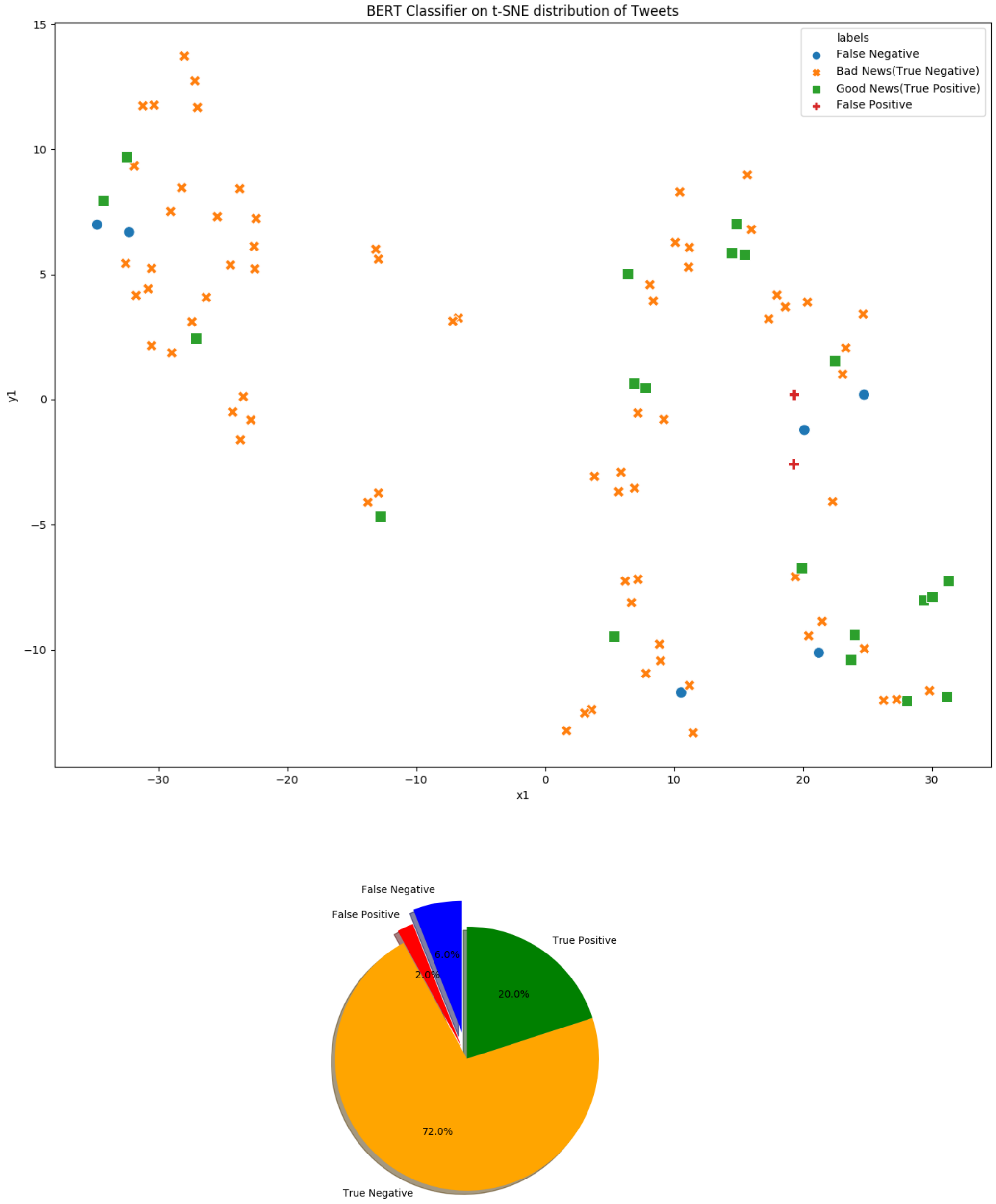


Figure 5: t-SNE distribution of random 100 test points with Bert's performance. The pie chart displays the percentage of BERT's misclassifications on these points.

John H. Boyd, Jian Hu, and Ravi Jagannathan. 2005. The stock market's reaction to unemployment news: Why bad news is usually good for stocks. *Journal of Finance* 60(2):649–672.

PHILLIP A. BRAUN, DANIEL B. NELSON, and ALAIN M. SUNIER. 1995. Good news, bad news, volatility, and betas. *The Journal of Finance* 50(5):1575–1603. https://doi.org/10.1111/j.1540-6261.1995.tb05189.x.

Jennifer Conrad, Bradford Cornell, and Wayne R. Landsman. 2002. When is bad news really bad news? *The Journal of Finance* 57(6):2507–2532. https://doi.org/10.1111/1540-6261.00504.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805. http://arxiv.org/abs/1810.04805.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382. https://doi.org/10.1037/h0031619.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 42–47. https://www.aclweb.org/anthology/P11-2008.

Alec Go. 2009. Sentiment classification using distant supervision.

Lars Kai Hansen, Adam Arvidsson, Finn Aarup Nielsen, Elanor Colleoni, and Michael Etter. 2011. Good friends, bad news - affect and virality in twitter. In *Communications in Computer and Information Science*, Springer Berlin Heidelberg, pages 34–43. https://doi.org/10.1007/978-3-642-22309-9_5.

Tony Harcup and Deirdre ONeill. 2017. What is news? *Journalism Studies* 18(12):1470–1488. https://doi.org/10.1080/1461670X.2016.1150193.

Wendy M. Johnston and Graham C. L. Davey. 1997. The psychological impact of negative tv news bulletins: The catastrophizing of personal worries. *British Journal of Psychology* 88(1):85–91. https://doi.org/10.1111/j.2044-8295.1997.tb02622.x.

Michael A. Kamins, Valerie S. Folkes, and Lars Perner. 1997. Consumer responses to rumors: Good news, bad news. *Journal of Consumer Psychology* 6(2):165–187. https://doi.org/10.1207/s15327663jcp0602_03.

Vishal Kharde, Prof Sonawane, et al. 2016. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971* .

Efthymios Kouloumpis, Theresa Wilson, and Johanna D. Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Paul R. Milgrom. 1981. Good news and bad news: Representation theorems and applications. *The Bell Journal of Economics* 12(2):380–391. http://www.jstor.org/stable/3003562.

Igor Mozetič, Miha Grčar, and Jasmina Smailović. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PloS one* 11(5):e0155036.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*. pages 1–18.

Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. 2011. Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings of the 3rd International Web Science Conference*. ACM, New York, NY, USA, WebSci '11, pages 8:1–8:7. https://doi.org/10.1145/2527031.2527052.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive twitter sentiment classification using neural network. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*. pages 502–518.

Paul Rozin and Edward B. Royzman. 2001. Negativity bias, negativity dominance, and contagion. *Personality and Social Psychology Review* 5(4):296–320. https://doi.org/10.1207/S15327957PSPR0504_2.

Aliaksei Severyn and Alessandro Moschitti. 2015. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. pages 464–469.

Stuart N. Soroka. 2006. Good news and bad news: Asymmetric responses to economic information. *The Journal of Politics* 68(2):372–385.

Stuart N. Soroka. 2006. Good news and bad news: Asymmetric responses to economic information. *The Journal of Politics* 68(2):372–385. https://doi.org/10.1111/j.1468-2508.2006.00413.x.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne.

Bilingual Low-Resource Neural Machine Translation with Round-Tripping: The Case of Persian-Spanish

Benyamin Ahmadnia[1] and **Bonnie J. Dorr**[2]

[1]Department of Computer Science, Tulane University, New Orleans, LA, USA
[2]Institute for Human and Machine Cognition (IHMC), Ocala, FL, USA
ahmadnia@tulane.edu , bdorr@ihmc.us

Abstract

The quality of Neural Machine Translation (NMT), as a data-driven approach, massively depends on quantity, quality and relevance of the training dataset. Such approaches have achieved promising results for bilingually high-resource scenarios but are inadequate for low-resource conditions. This paper describes a round-trip training approach to bilingual low-resource NMT that takes advantage of monolingual datasets to address training data scarcity, thus augmenting translation quality. We conduct detailed experiments on Persian-Spanish as a bilingually low-resource scenario. Experimental results demonstrate that this competitive approach outperforms the baselines.

1 Introduction

Neural Machine Translation (NMT) has made considerable progress in recent years. However, to achieve acceptable translation output, large sets of aligned parallel sentences are required for the training phase. Thus, as a data-driven paradigm, the quality of NMT output strongly depends on the quality as well as quantity of the provided training data (Bahdanau et al., 2015). Unfortunately, in practice, collecting such parallel text by human labeling is very costly and time consuming (Ahmadnia and Serrano, 2017).

Low-resource languages are those that have fewer technologies and datasets relative to some measure of their international importance. The biggest issue with low-resource languages is the extreme difficulty of obtaining sufficient resources. Natural Language Processing (NLP) methods that have been created for analysis of low-resource languages are likely to encounter similar issues to those faced by documentary and descriptive linguists whose primary endeavor is the study of minority languages. Lessons learned from such studies are highly informative to NLP researchers who seek to overcome analogous challenges in the computational processing of these types of languages.

Assuming that large monolingual texts are available, an obvious next step is to leverage these texts to augment the NMT systems' performance. Various approaches have been developed for this purpose. In some approaches, target monolingual texts are employed to train a Language Model (LM) that is then integrated with MT models trained from parallel texts to enhance translation quality (Brants et al., 2007; Gülçehre et al., 2015). Although these approaches utilize monolingual text to train a LM, they do not address the shortage of bilingual training datasets.

In other approaches, bilingual datasets are automatically generated from monolingual texts by utilizing the Translation Model (TM) trained on aligned bilingual text; the resulting sentence pairs are used to enlarge the initial training dataset for subsequent learning iterations (Ueffing et al., 2008; Sennrich et al., 2016). Although these approaches enlarge the bilingual training dataset, there is no quality control and, thus, the accuracy of the generated bilingual dataset cannot be guaranteed (Ahmadnia et al., 2018).

To tackle the issues described above, we apply a new round-tripping approach that incorporates *dual learning* (He et al., 2016) for automatic learning from unlabeled data, but transcends this prior work through effective leveraging of monolingual text. Specifically, the round-tripping approach takes advantage of the bootstrapping methods including self-training and co-training. These methods start their mission from a small set of labelled examples, while also considering one or two weak translation models, and makes improvement through the incorporation of unlabeled data into the training dataset.

In the round-tripping approach, the two translation tasks (forward and backward) together make a *closed loop*, i.e., one direction produces informative *feedback* for training the TM for the other direction, and vice versa. The feedback signals—which consist of the language model likelihood of the output model and the reconstruction error of the original sentence—drive the process of iterative updates of the forward and backward TMs.

For the purpose of evaluation, we apply this approach to a bilingually low-resource language pair (Persian-Spanish) to leverage monolingual data in a more effective way. By utilizing the round-tripping approach, the monolingual data play a similar role to the bilingual data, effectively reducing the requirement for parallel data. In particular, each model provides guidance to the other throughout the learning process. Our results show that round-tripping for NMT works well in the Persian-Spanish low-resource scenario. By learning from monolingual data, this approach achieves comparable accuracy to a NMT approach trained from the full bilingual data for the two translation tasks (forward and backward).

The remainder of this paper is organized as follows; Section 2 presents the previous related work. In Section 3, we briefly review the relevant mathematical background of NMT paradigm. Section 4 describes the round-trip training approach. The experiments and results are presented in Section 5. Conclusions and future work are discussed in Section 6.

2 Related Work

The integration of monolingual data for NMT models was first proposed by (Gülçehre et al., 2015), who train monolingual LMs independently, and then integrate them during decoding through rescoring of the beam (adding the recurrent hidden state of the LM to the decoder state of the encoder-decoder network). In this approach, an additional controller mechanism controls the magnitude of the LM signal. The controller parameters and output parameters are tuned on further parallel training data, but the LM parameters are fixed during the fine tuning stage.

Jean et al. (2015) also report on experiments with reranking of NMT output with a 5-gram LM, but improvements are small. The production of synthetic parallel texts bears resemblance to data augmentation techniques, where datasets are often augmented with rotated, scaled, or otherwise distorted variants of the (limited) training set (Rowley et al., 1998).

A similar avenue of research is self-training (McClosky et al., 2006). The self-training approach as a bootstrapping method typically refers to the scenario where the training dataset is enhanced with training instances with artificially produced output labels (whereas we start with neural network based output, i.e., the translation, and artificially produce an input). We expect that this is more robust towards noise in MT.

Hoang et al. (2018) showed that the quality of back translation matters and proposed an iterative back translation, in which back translated data are used to build better translation systems in forward and backward directions. These, in turn, are used to reback translate monolingual data. This process is iterated several times.

Improving NMT with monolingual source data, following similar work on phrase-based SMT (Schwenk, 2008), remains possible future work. Domain adaptation of neural networks via continued training has been shown to be effective for neural language models by (Ter-Sarkisov et al., 2015).

Round-tripping has already been utilized in SMT by (Ahmadnia et al., 2019). In this work, forward and backward models produce informative feedback to iteratively update the TMs during the training of the system.

3 Neural Machine Translation

NMT consists of an encoder and a decoder. Following (Bahdanau et al., 2015), we adopt an *attention-based* encoder-decoder model, i.e., one that selectively focuses on sub-parts of the sentence during translation. Consider a source sentence $X = \{x_1, x_2, ..., x_J\}$ and a target sentence $Y = \{y_1, y_2, ..., y_I\}$. The problem of translation from the source language to the target is solved by finding the best target language sentence $\hat{y}$ that maximizes the conditional probability:

$$\hat{y} = \arg\max_{y} P(y|x) \tag{1}$$

The conditional word probabilities given the source language sentence and preceding target language words compose the conditional probability as follows:

$$P(y|x) = \prod_{i=1}^{I} P(y_i|y_{<i}, x) \tag{2}$$

where y_i is the target word emitted by the decoder at step i and $y_{<_i} = (y_1, y_2, ..., y_{i-1})$.

To compose the model, both the encoder and decoder are implemented employing Recurrent neural Networks (RNNs) (Rumelhart et al., 1986), i.e., the encoder converts source words into a sequence of vectors, and the decoder generates target words one-by-one based on the conditional probability shown in the Equation (2). More specifically, the encoder takes a sequence of source words as inputs and returns forward hidden vectors $\overrightarrow{h_j}(1 \leq j \leq J)$ of the forward-RNN:

$$\overrightarrow{h_j} = f(\overrightarrow{h_{j-1}}, x) \tag{3}$$

Similarly, we obtain backward hidden vectors $\overleftarrow{h_j}(1 \leq j \leq J)$ of the backward-RNN, in the reverse order.

$$\overleftarrow{h_j} = f(\overleftarrow{h_{j-1}}, x) \tag{4}$$

These forward and backward vectors are concatenated to make source vectors $h_j(1 \leq j \leq J)$ based on Equation (5):

$$h_j = \left[\overrightarrow{h_j}; \overleftarrow{h_j}\right] \tag{5}$$

The decoder takes source vectors as input and returns target words. It starts with the initial hidden vector h_J (concatenated source vector at the end), and generates target words in a recurrent manner using its hidden state and an output context.

The conditional output probability of a target language word y_i is defined as follows:

$$P(y_i|y_{<i}, x) = \mathit{softmax}\left(f(d_i, y_{i-1}, c_i)\right) \tag{6}$$

where f is a non-linear function and d_i is the hidden state of the decoder at step i:

$$d_i = g(d_{i-1}, y_{i-1}, c_i) \tag{7}$$

where g is a non-linear function taking its previous state vector with the previous output word as inputs to update its state vector. c_i is a context vector to retrieve source inputs in the form of a weighted sum of the source vectors h_j, first taking as input the hidden state d_i at the top layer of a stacking LSTM (Hochreiter and Schmidhuber, 1997). The goal is to derive a context vector c_i that captures relevant source information to help predict the current target word y_i.

While these models differ in how the context vector c_i is derived, they share the same subsequent steps. c_i is calculated as follows:

$$c_i = \sum_{j=1}^{J} \alpha_{t,j} h_j \tag{8}$$

where h_j is the annotation of source word x_j and $\alpha_{t,j}$ is a weight for the j^{th} source vector at time step t to generate y_i:

$$\alpha_{t,j} = \frac{\mathit{exp}\left(\mathit{score}\left(d_i, h_j\right)\right)}{\sum_{j'=1}^{J} \mathit{exp}\left(\mathit{score}\left(d_i, h_{j'}\right)\right)} \tag{9}$$

The score function above may be defined in a variety of ways as discussed by Luong et al. (2015).

In this paper, we denote all the parameters to be optimized in the neural network as Θ and denote C as the dataset that contains source-target sentence pairs for the training phase. Hence, the learning objective is to seek the optimal parameters Θ^*:

$$\Theta^* = \arg\max_{\Theta} \sum_{(x,y)\in C} \sum_{(i=1)}^{I} \log P(y_t|y_{<t}, x; \Theta) \tag{10}$$

4 Method Description

Round-tripping involves two related translation tasks: the outbound-trip (source-target direction) and the inbound-trip (target-source direction). The defining traits of these forward and backward tasks are that they form a closed loop and both produce informative feedback that enables simultaneous training of the TMs.

We assume availability of: (1) monolingual datasets (C_X and C_Y) for the source and target languages; and (2) two weak TMs (emergent from training on initial small bilingual corpora) that bidirectionally translate sentences from source and target languages. The goal of the round-tripping approach is to augment the accuracy of the two TMs by employing the two monolingual datasets instead of a bilingual text.

We start by translating a sample sentence in one of the monolingual datasets, as the outbound-trip (forward) translation to the target language. This step generates more bilingual sentence pairs between the source and target languages. We then translate the resulting sentence pairs backward through the inbound-trip translation to the original language. This step finds high-quality sentences throughout the entirety of the generated sentence pairs. Evaluating the results of this round-tripping approach will provide an indication of the quality of the two TMs, and will enable their enhancement, accordingly. This process is iterated for several rounds until both TMs converge.

We define K_X as the number of sentences in C_X and K_Y as the number of sentences in C_Y. We take $P(.|S;\Theta_{XY})$ and $P(.|S;\Theta_{YX})$ to be two neural TMs in which Θ_{XY} and Θ_{YX} are supposed as their parameters. We also assume the existence of two LMs for languages X and Y, trained in advance either by using other resources or using the monolingual data (C_X and C_Y). Each LM takes a sentence as input and produces a real number, based on target-language fluency (LM correctness) together translation accuracy (TM correctness). This number represents the confidence of the translation quality of the sentence in its own language.

We start with a sentence in C_X and denote S_{sample} as a translation output sample. This step has a score as follows:

$$R_1 = LM_Y(S_{sample}) \qquad (11)$$

The R_1 score indicates the well-formedness of the output sentence in language Y.

Given the translation output S_{sample}, we employ the log probability value of s recovered from the S_{sample} as the score of the construction:

$$R_2 = \log P(S|S_{sample};\Theta_{YX}) \qquad (12)$$

We then adopt the LM score and construction score as the total reward score:

$$R_{total} = \alpha R_1 + (1-\alpha)R_2 \qquad (13)$$

where α is an input hyper-parameter.

The total reward score is considered a function of S, S_{sample}, Θ_{XY} and Θ_{YX}. To maximize this score, we optimize the parameters in the TMs utilizing Stochastic Gradient Descent (SGD) (Sutton et al., 2000). According to the forward TM, we sample the s_{sample} and then compute the gradient of the expected score ($E[R_{total}]$), where E is taken from S_{sample}:

$$\nabla_{\Theta_{XY}} E[R_{total}] = E[R_{total} \nabla_{\Theta_{XY}} \log P(S_{sample}|S;\Theta_{XY})] \qquad (14)$$

$$\nabla_{\Theta_{YX}} E[R_{total}] = E[(1-\alpha) \nabla_{\Theta_{YX}} \log P(S|S_{sample};\Theta_{YX})] \qquad (15)$$

Algorithm 1 shows the round-tripping procedure.

Algorithm 1 Round-trip training for NMT

```
Input: Monolingual dataset in source and target languages ($C_X$ and $C_Y$), initial translation
       models in outbound and inbound trips ($\Theta_{XY}$ and $\Theta_{YX}$), language models in source
       and target languages ($LM_X$ and $LM_Y$), trade-off parameter between 0 and 1 ($\alpha$),
       beam search size ($N$), learning rates ($\gamma_{1,t}$ and $\gamma_{2,t}$).
 1: repeat:
 2:   $t = t + 1$.
 3:   Sample sentences $S_X$ and $S_Y$ from $C_X$ and $C_Y$ respectively.
 4:   // Update model starting from language $X$.
      Set $S = S_X$.
 5:   // Generate top-$N$ translations using $\Theta_{XY}$.
      Generate sentences $S_{sample,1}, ..., S_{sample,N}$.
 6:   for $n = 1, ..., N$ do
 7:     // Set LM score for $n^{th}$ sampled sentence.
        $R_{1,n} = LM_Y(S_{sample,n})$.
 8:     // Set TM score for $n^{th}$ sampled sentence.
        $R_{2,n} = logP(S|S_{sample,n};\Theta_{YX})$.
 9:     // Set total score of $n^{th}$ sampled sentence.
        $R_n = \alpha R_{1,n} + (1-\alpha)R_{2,n}$.
10:   end for
11:   // SDG computing for $\Theta_{XY}$.
      $\nabla_{\Theta_{XY}} \hat{E}[R_{total}] = \frac{1}{N}\sum_{n=1}^{N} [R_n \nabla_{\Theta_{XY}} \log P(S_{sample,n}|S;\Theta_{XY})]$.
12:   // SDG computing for $\Theta_{YX}$.
      $\nabla_{\Theta_{YX}} \hat{E}[R_{total}] = \frac{1}{N}\sum_{n=1}^{N} [(1-\alpha)\nabla_{\Theta_{YX}} \log P(S|S_{sample,n};\Theta_{YX})]$.
13:   // Model update.
      $\Theta_{XY} \leftarrow \Theta_{XY} + \gamma_{1,t}\nabla_{\Theta_{XY}} \hat{E}[R_{total}]$.
14:   // Model update.
      $\Theta_{YX} \leftarrow \Theta_{YX} + \gamma_{2,t}\nabla_{\Theta_{YX}} \hat{E}[R_{total}]$.
15:   // Update model starting from language $Y$.
      Set $S = S_Y$.
16:   Go through lines $5 - 14$ symmetrically.
17: until convergence.
```

To achieve reasonable translations we use beam search. We generate N-best sample translations and use the averaged value on the beam search results to estimate the true gradient value.[1]

[1] We used beam sizes 500 and 1000.

5 Experiments and Results

We apply the round-trip training approach to bilingual Persian-Spanish translation, and evaluate the results. We used the Persian-Spanish small bilingual corpora from the *Tanzil* corpus (Tiedemann, 2012),[2] which contains about 50K parallel sentence pairs. We also used 5K and 10K parallel sentences extracted from the *OpenSubtitles2018* collection (Tiedemann, 2012),[3] as the validation and test datasets, respectively. Finally, we utilized 70K parallel sentences from the *KDE4* corpus (Tiedemann, 2012),[4] as the monolingual data.

We implemented the DyNet-based model architecture (Mi et al., 2016) on top of *Mantis* (Cohn et al., 2016) which is an implementation of the attention sequence-to-sequence (Seq-to-Seq) NMT. For each language, we constructed a vocabulary with the most common 50K words in the parallel corpora, and OOV words were replace with a special token <*UNK*>. For monolingual corpora, sentences containing at least one OOV word were removed. Additionally, sentences with more than 80 words were removed from the training set.[5] The encoders and decoders make use of Long Short-Term Memory (LSTM) with 500 embedding dimensions, 500 hidden dimensions. For training, we used the SGD algorithm as the optimizer. The batch size was set as 64 with 20 batches pre-fetched and sorted by sentence lengths.

Below we compare the system based on the optimized round-trip training (round-tripping) through two translation systems; the first one is the standard NMT system (baseline), and the second one is the system that generates pseudo bilingual sentence pairs from monolingual corpora to assist the training step (self-training). For the pseudo-NMT we used the trained NMT model to generate pseudo bilingual sentence pairs from monolingual text, removed sentences with more than 80 words (as above), merged the generated data with the original parallel training data, and then trained the model for testing. Each of the translation systems was trained on a single GPU until their performances stopped improving on the validation set. This approach required an LM for each language. We trained an RNN-based LM (Mikolov et al., 2010) for each language using its corresponding monolingual corpus. The LM was then fixed and the log-likelihood of a received message was utilized for scoring the TM.

To start the round-trip training approach, the systems are initialized using warm-start TMs trained from initial small bilingual data. The goal is to see whether the round-tripping augments the baseline accuracy. We retrain the baseline systems by enlarging the initial small bilingual corpus: we add the optimized generated bilingual sentences to the initial parallel text. The new enlarged translation system contains both the initial and optimized generated bilingual text. For each translation task, we train the round-trip training approach.

We employ Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2001) (using *multi-bleu.perl* script from Moses) as the evaluation metric. BLEU is calculated for individual translated segments by comparing them with a data set of reference translations. The scores of each segment, ranging between 0 and 100, are averaged over the entire evaluation dataset to yield an estimate of the overall translation quality (higher is better).

The baseline systems for Persian-Spanish are first trained, while our round-trip method conducts joint training. We summarize the overall performances in Table 1:

NMT systems	**Pe-Es**	**Es-Pe**
baseline	31.12	29.56
self-train	29.29	27.36
round-trip	34.91	33.43

Table 1: BLEU scores for Persian-Spanish translation task and vice-versa.

As seen in Table 1, the round-tripping systems outperform the others in both translation directions. In Persian to Spanish translation, the round-tripping system outperforms the baseline by about 3.87 BLEU points and also outperforms the self-training system by about 6.07 BLEU points. In the back translation from Spanish to Persian, the improvement of the round-tripping outperforms both the baseline and self-training by about 3.79 and 5.62 BLEU points, respectively.

These results demonstrate the effectiveness of the round-trip training approach. The baseline systems outperform the self-training ones in all cases

[2] http://opus.nlpl.eu/Tanzil.php

[3] http://opus.nlpl.eu/OpenSubtitles-v2018.php

[4] http://opus.nlpl.eu/KDE4-v2.php

[5] The average sentence length is 47; an upper bound of 80 ensured exclusion of non-sentential and other spurious material.

because of the noise in the generated bilingual sentences used by self-training. Upon further examination, this result might have been expected given that the aim of round-trip training is to optimize the generated bilingual sentences by selecting the high-quality sentences to get better performance over self-training systems. When the size of bilingual corpus is small, the round-tripping makes a larger improvement. This outcome is an indication that round-trip training approach makes effective use of monolingual data.

Table 2 shows the performance of the baseline alongside of the enlarged translation systems, where the latter leverages the training text of the baseline and the round-tripping systems as well.

NMT systems	Pe-Es	Es-Pe
baseline	31.12	29.56
enlarged	34.21	33.03

Table 2: BLEU scores comparing the baseline and enlarged NMT systems for Persian-Spanish translation task and vice-versa.

As seen in Table 2, the BLEU scores of the enlarged NMT systems are better than the baseline ones in both translation directions. The enlarged system in the Persian-Spanish direction outperforms the baseline by about 3.47 BLEU points, and outperforms the baseline by about 3.09 BLEU points in the back translation. The improvements show that the optimized round-trip training system is promising for tackling the training data scarcity and it also helps to enhance translation quality.

6 Conclusions and Future Work

In this paper, we applied a round-tripping approach based on a retraining scenario to tackle training data scarcity in NMT systems. An exciting finding of this work is that it is possible to learn translations directly from monolingual data of the two languages. We employed a low-resource language pair and verified the hypothesis that, regardless of the amount of training resources, this approach outperforms the baseline. The results demonstrate that round-trip training is promising and better utilizes the monolingual data.

Many Artificial Intelligence (AI) tasks are naturally in dual form. Some examples are: (1) speech recognition paired with text-to-speech; (2) image captioning paired with image generation; and (3) question answering paired with question generation. Thus, a possible future direction would be to design and test the round-tripping approach for more tasks beyond MT. We note that round-tripping is not restricted to two tasks only. Indeed, the key idea is to form a closed loop so feedback signals are extracted by comparing the original input data with the final output data. Therefore, if more than two associated tasks form a closed loop, this approach can applied in each task for improvement of the overall model, even in the face of unlabeled data.

Acknowledgments

We would like to express our sincere gratitude to Dr. Michael W. Mislove (Tulane University, USA), Dr. Javier Serrano (Universitat Autònoma de Barcelona, Spain) and Dr. Gholamreza Haffari (Monash University, Australia) for all their support. We also would like to acknowledge the financial support received from the School of Science and Engineering, Tulane University (USA).

References

Benyamin Ahmadnia, Gholamreza Haffari, and Javier Serrano. 2018. Statistical machine translation for bilingually low-resource scenarios: A round-tripping approach. In *Proceedings of the 3rd IEEE International Conference on Machine Learning and Natural Language Processing*. pages 261–265.

Benyamin Ahmadnia, Gholamreza Haffari, and Javier Serrano. 2019. Round-trip training approach for bilingually low-resource statistical machine translation systems. *International Journal of Artificial Intelligence* 17(1):167–185.

Benyamin Ahmadnia and Javier Serrano. 2017. Employing pivot language technique through statistical and neural machine translation frameworks: The case of under-resourced persian-spanish language pair. *International Journal on Natural Language Computing* 6(5):37–47.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 858–867.

Trevor Cohn, Cong Duy Vu Huang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza

Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*. pages 876–885.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR* abs/1503.03535.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Proceedings of the 30th Conference on Neural Information Processing Systems*.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. pages 18–24.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* .

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. pages 11–19.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. pages 152–159.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*. pages 2283–2288.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*. pages 1045–1048.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. pages 311–318.

Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. 1998. Neural network-based face detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 20(1):23–38.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.

Holger Schwenk. 2008. Investigations on large-scale lightly-supervised training for statistical machine translation. In *Proceedings of IWSLT*. pages 182–189.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of Association for Computational Linguistics*.

Richard S. Sutton, David A. Mcallester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*. volume 12, pages 1057–1063.

Aram Ter-Sarkisov, Holger Schwenk, Loïc Barrault, and Fethi Bougares. 2015. Incremental adaptation strategies for neural network language models. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. pages 48–56.

Jørg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2008. On using monolingual corpora in statistical machine translation. *Journal of Machine Translation* .

Enhancing Phrase-Based Statistical Machine Translation by Learning Phrase Representations Using Long Short-Term Memory Network

Benyamin Ahmadnia[1] and **Bonnie J. Dorr**[2]

[1]Department of Computer Science, Tulane University, New Orleans, LA, USA
[2]Institute for Human and Machine Cognition (IHMC), Ocala, FL, USA
ahmadnia@tulane.edu , bdorr@ihmc.us

Abstract

Phrases play a key role in Machine Translation (MT). In this paper, we apply a Long Short-Term Memory (LSTM) model over conventional Phrase-Based Statistical MT (PBSMT). The core idea is to use an LSTM encoder-decoder to score the phrase table generated by the PBSMT decoder. Given a source sequence, the encoder and decoder are jointly trained in order to maximize the conditional probability of a target sequence. Analytically, the performance of a PBSMT system is enhanced by using the conditional probabilities of phrase pairs computed by an LSTM encoder-decoder as an additional feature in the existing log-linear model. We compare the performance of the phrase tables in the PBSMT to the performance of the proposed LSTM and observe its positive impact on translation quality. We construct a PBSMT model using the Moses decoder and enrich the Language Model (LM) utilizing an external dataset. We then rank the phrase tables using an LSTM-based encoder-decoder. This method produces a gain of up to 3.14 BLEU score on the test set.

1 Introduction

The three most essential components of a Statistical Machine Translation (SMT) system are: (1) Translation Model (TM); (2) Language Model (LM); and (3) Reordering Model (RM). Among these models, the RM plays an important role in Phrase-Based SMT (PBSMT) (Marcu and Wong, 2002; Koehn et al., 2003), and it still remains a major focus of intense study (Kanouchi et al., 2016; Du and Way, 2017; Chen et al., 2019).

The RM is required since different languages exercise different syntactic ordering. For instance, adjectives in English precede the noun, while they typically follow the noun in Spanish (*the cloudy sky* versus *el cielo nublado*); in Persian the verb precedes the subject, and in Chinese the verb comes last. As a result, source language phrases cannot be translated and placed in the same order in the generated translation in the target language, but phrase movements have to be considered. This is the role of the RM. Estimating the exact distance of movement for each phrase is too sparse; therefore, instead, the lexicalized RM (Koehn, 2009) estimates phrase movements using only a few reordering types, such as a monotonous order, where the order is preserved, or a swap, when the order of two consecutive source phrases is inverted when their translations are placed in the target side.

Neural Machine Translation (NMT) has been receiving significant attention due to its impressive translation performance (Bahdanau et al., 2015). NMT differs from SMT in its adoption of a large neural network to perform the entire translation process in one shot, for which an encoder-decoder architecture is widely used. In this approach, a source sentence is encoded into a continuous vector representation. Subsequently, the decoder uses that representation to generate the corresponding target translation.

NMT's word-by-word translation generation strategy makes it difficult to translate phrases. This is a significant MT challenge as the meaning of a phrase is not always deducible from the meanings of its individual words or parts. Unfortunately, current NMT systems are word-based or character-based where phrases are not considered as translation units. By contrast, phrases are more effective than words as translation units in SMT. Indeed, leveraging phrases has had a significant impact on translation quality (Wang et al., 2017).

Proceedings of Recent Advances in Natural Language Processing, pages 25–32,
Varna, Bulgaria, Sep 2–4 2019.

Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) are a class of artificial neural network that has recently resurfaced in the field of MT (Schwenk, 2012). Unlike feed-forward networks, RNNs leverage recurrent connections that enable the network to refer to prior states and, thus, to process arbitrary sequences of input. The cornerstone of RNNs is their ability to connect previous information to the present task. For example, given a LM that predicts the next word based on previous words, no further context is needed to predict the last word in *the clouds are in the sky*.

When the gap between the relevant information and the place that it is needed is small, RNNs learn the next word from past information. But there are cases where more context is needed, e.g., in the prediction of the last word in *I grew up in Spain and I speak fluent Spanish*. The word *Spain* suggests that the last word is probably the name of a language, but to narrow down that language, access to a larger context is needed. It is entirely possible for the gap between the relevant information and the point where it is needed to become indefinitely large. Unfortunately, as that gap grows, RNNs are increasingly unable to learn to connect the information.

Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) are an extension of RNNs, capable of learning such long-term dependencies. RNNs are a chain of repeating modules of neural network and, in their simplest form, the repeating module has a single layer. LSTMs also have this chain-like structure, but the repeating modules have four interacting layers.

This paper presents a PBSMT model based on the Moses decoder (Koehn et al., 2007) with a LM that is enriched by an external dataset. Scoring of the phrase table generated by Moses is achieved through a LSTM encoder-decoder, and the result is then evaluated in an English-to-Spanish translation task. Specifically, the model is trained to learn the translation probabilities between English phrases and their corresponding Spanish ones. The trained model is then used as a part of a classical PBSMT system, with each phrase pair scored in the phrase table. Our evaluation proves that this approach enhances the translation performance. Although Moses itself is able to score phrases as a part of the coding process, our approach includes the scoring of phrases using the LSTM-based encoder-decoder, thus yielding improvements in quality. Sentences with the highest scores are selected as the translation output.

The rest of this paper is organized as follows: Section 2 discusses the previous related work. Sections 3 and 4 describe our PBSMT framework and LSTM encoder-decoder integration, respectively. Section 5 presents the key elements of our approach, bringing together PBSMT and LSTM encoder-decoder for phrase scoring. The experimental results are covered in Section 6. Finally, Section 7 presents conclusions and future work.

2 Related Work

Recently, various neural network models have been applied to MT. However, few approaches have made effective use of neural networks to enhance the translation quality of SMT.

Sundermeyer et al. (2014) designed a neural TM that uses LSTM-based RNNs and Bidirectional RNNs, wherein the target word is conditioned not only on the history but also on the future source context. The result was a fully formed source sentence for predicting target words.

Feed-forward neural LMs, first proposed by Bengio et al. (2003), were a breakthrough in language modeling. Mikolov et al. (2011) proposed the use of recurrent neural network in language modeling, thus enabling a much longer context history for predicting the next word. Experimental results showed that the RNN-based LM significantly outperforms the standard feed-forward LM.

Schwenk (2012) proposed a feed-forward neural network to score phrase pairs. They employed a feed-forward neural network with fixed-size phrasal inputs consisting of seven words, and with zero padding for shorter phrases. The system also had fixed-size phrasal output consisting of seven words. Similarly, Devlin et al. (2014) utilized a feed-forward neural network to generate translations, but they simultaneously predicted one word in a target phrase. The use of feed-forward neural networks demands the use of fixed-size phrases to work properly.

Zou et al. (2013) also proposed bilingual learning of word and phrase embeddings, which were used to compute the distance between phrase pairs. The result was an additional annotation to score the phrase pairs of an SMT system.

Chandar et al. (2014) trained a feed-forward neural networks to learn the mapping of an input phrase to its corresponding output phrase using a bag-of-words approach. This is closely related to the model proposed by Schwenk (2012), except that their input representation of a phrase was a Bag-Of-Words (BOW). A similar encoder-decoder approach that used two RNNs was proposed by Socher et al. (2011), but their model was restricted to a monolingual setting.

More recently, an encoder-decoder model using an RNN was proposed by Auli et al. (2013), where the decoder was conditioned on a representation of either a source sentence or a source context. Kalchbrenner and Blunsom (2013) proposed a similar model that uses the concept of an encoder and decoder. They used an n-gram LM for the encoder part and a combination of inverse LM and an RNN for the decoder part. The evaluation of their model was based on rescoring the K-best list of the phrases from the SMT phrase table.

3 From SMT to PBMST

Our enhancement to SMT takes a noisy channel model as a starting point, where translation is modeled by decoding a source text, thereby eliminating the noise (e.g., adjusting lexical and syntactic divergences) to uncover the intended translation. However, as in our prior work (Ahmadnia et al., 2017), we adopt a more general, log-linear variant to accommodate an unlimited number of features and to provide a more general framework for controlling each feature's influence on the overall output. Standard probabilities are scaled to their logarithmic counterparts that are then added together, rather than multiplying, following standard logarithmic rules. The log-linear model is derived via direct modelling of the posterior probability $P(y_1^I|x_1^J)$:

$$\hat{y} = \arg\max_{y_1^I} P(y_1^I|x_1^J) \tag{1}$$

where x is a source sentence.

The PBSMT model is an example of the noisy channel approach, where the translation hypothesis y is presented as the target sentence (given x as a source sentence), and the log-linear combination of feature functions is maximized:

$$\hat{y} = \arg\max_{y_1^I} \left\{ \sum_{m=1}^{M} \lambda_m h_m(x_1^J, y_1^I) \right\} \tag{2}$$

In the log-linear model of Equation (2), λ_m corresponds to the weighting coefficients of the log-linear combination. Feature functions $h_m(x_1^J, y_1^I)$ correspond to a logarithmic scaling of the probabilities of each model. The translation process involves segmenting the source sentence into source phrases x, each of which is translated into a target phrase y, and reordering these target phrases to yield the target sentence $\hat{y}$. This model is considered superior in comparison to the noisy-channel model because of the ability to adjust the importance of individual features, thus controlling each feature's influence on the overall output.

In the PBSMT model, the TM is factored into the translation probabilities of matching phrases in the source and target sentences (Ahmadnia and Serrano, 2015). These are considered additional features in the log-linear model and are weighted accordingly to maximize the performance as measured by Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2001). The neural LM Bengio et al. (2003) has become a community standard for SMT system development, i.e., neural networks have been used to rescore translation hypotheses (k-best lists). Recently, however, there has been an emerging interest in training neural networks to score the translated phrase pairs using a source-sentence representation as an additional input (Zou et al., 2013). We adopt this approach for our own PBSMT enrichment, as further detailed below.

4 Integration of LSTM Encoder-Decoder

Following Ahmadnia et al. (2018), we enhance NMT performance by estimating the conditional probability $P(y_i^I|x_j^J)$ where $(x_j, ..., x_J)$ is a source sequence and $(y_i, ..., y_I)$ is its corresponding target sequence whose length I may differ from J. The LSTM computes this conditional probability by first obtaining the fixed-dimensional representation ν of the source sequence given by the last hidden state of the LSTM, and then computing the probability of the target sequence with a standard LSTM as the neural LM formulation whose initial hidden state is set to the representation ν of the source sequence. This is specified as follows:

$$P(y_i^I|x_j^J) = \prod_{i=1}^{I} P(y_i|\nu, y_1^{i-1}) \tag{3}$$

where the $P(y_i|\nu, y_1^{i-1})$ distribution is represented with a softmax over all words in the vocabulary.

To compose the model, both the encoder and decoder are implemented employing RNNs, i.e., the encoder converts source words into a sequence of vectors, and the decoder generates target words one-by-one based on the conditional probability shown in Equation (3). Specifically, the encoder takes a sequence of source words as inputs and returns forward hidden vectors $\overrightarrow{h_j}(1 \leq j \leq J)$ of the forward-RNN:

$$\overrightarrow{h_j} = f(\overrightarrow{h_{j-1}}, x) \tag{4}$$

Similarly, backward hidden vectors $\overleftarrow{h_j}(1 \leq j \leq J)$ of the backward-RNN are obtained, in reverse order.

$$\overleftarrow{h_j} = f(\overleftarrow{h_{j-1}}, x) \tag{5}$$

These forward and backward vectors are concatenated to make source vectors $h_j(1 \leq j \leq J)$ based on Equations (4) and (5):

$$h_j = \left[\overrightarrow{h_j}; \overleftarrow{h_j}\right] \tag{6}$$

The decoder takes source vectors as inputs and returns target words, starting with the initial hidden vector h_J.[1] Target words are generated in a recurrent manner using the decoder's hidden state and an output context. The conditional output probability of a target language word y_i is defined as follows:

$$P_\theta(y_t|y_{<t}, X) = softmax\ (W_s\tilde{h}_t) \tag{7}$$

where W_s is a parameter matrix in the output layer and $\tilde{h}_t$ is a vector:

$$\tilde{h}_t = tanh(W_c[c_t; h_t]) \tag{8}$$

where W_c is a parameter matrix and $h_t = g(h_{t-1}, y_{t-1})$. Here, g is an RNN function that takes its previous state vector and previous output word as input and updates its state vector. c_t is a context vector to retrieve source inputs in the form of a weighted sum of the source vectors h_j, first taking as input the hidden state h_t at the top layer of a stacking LSTM.

The goal of the approach above is to derive a context vector c_t that captures relevant source information, thus enabling the prediction of the current target word y_t. While a variety of models may be used to derive a range of different context vectors c_t, the same subsequent steps are taken. Equation (8) defines our choice for c_t:

$$c_t = \sum_{j=1}^{S} \alpha_{ij} h_j \tag{9}$$

where α_{ij} is a weight for the j^{th} source vector at time step t to generate y_i:

$$\alpha_{ij} = \frac{exp\ (score\ (h_t, h_j))}{\sum_{j'=1}^{J} exp\ (score\ (h_t, h_{j'}))} \tag{10}$$

The score function above is calculated as follows:[2]

$$score(h_t, h_j) = h_t^T h_j \tag{11}$$

Given training data with K bilingual sentences, we train the model by maximizing the log-likelihood as follows:

$$L(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{J} \log P(y_i^k | y_{<i}^k, x^k) \tag{12}$$

5 Phrase Scoring by LSTM

The centerpiece of our PBSMT enhancements is the inclusion of two stages: (1) training of a LSTM encoder-decoder on a phrase table; and (2) subsequent use of training output scores as additional features in the log-linear model when tuning the SMT decoder.

During LSTM encoder-decoder training, the frequencies of each phrase pair in the original corpora are ignored. This measure is taken to reduce the computational expense of randomly selecting phrase pairs from a large phrase table according to the frequencies. Additionally, this measure ensures that the LSTM encoder-decoder does not learn to rank each phrase pair according to its frequency of occurrence.

Regarding the latter point, one reason behind this choice is that the existing translation probability in the phrase table already reflects the frequency of phrase pair occurrence in the original corpus. With a fixed capacity of the LSTM encoder-decoder, we need to ensure that most of the capacity of the model is focused on learning linguistic regularities, i.e., distinguishing between translations, or learning the manifold[3] of translations. Once the LSTM encoder-decoder is trained, we add a new score for each phrase pair to the existing phrase table. This allows the new scores to enter into the existing tuning algorithm with minimal additional overhead in computation.

An alternative to what is described above is the replacement of the existing phrase table with the

[1] Concatenated source vector at the end.

[2] The decoder puts more attention (weights) on source vectors close to the state vector.

[3] Region of probability concentration.

LSTM encoder-decoder. In such an approach, the problem would be recast in the form of the following implementation: given a source phrase, the LSTM encoder-decoder generates a list of target phrases (Schwenk, 2012). However, in this alternative, an expensive sampling procedure must be performed repeatedly. In our approach, the only phrase pairs that are rescored are those in the phrase table.

6 Experiments and Results

Numerous large resources are available for building an English-Spanish SMT system, many of which have become community standards, used in translation tasks in annual workshops and conferences on SMT hosted by ACL, NAACL, EACL, and EMNLP (SMT 2006-2015 and WMT 2016-2019). Bilingual datasets include Europarl, News-Commentary, UN, and two crawled corpora.[4] For our purposes, we have trained the Spanish LM using about 700M words of crawled newspaper material.[5]

We select a subset of 350M words for language modeling as well as a subset of 300M words for training the LSTM encoder-decoder. We use the test set newstest2011 and newstest2012 for data selection and weight tuning with Minimum Error rate Training (MERT) (Och, 2003), and newstest2013 as our test set. Each set has more than 70K words and a single reference translation.

For training the neural networks, including our LSTM encoder-decoder, we limited the source and target vocabulary to the most frequent 10K words for both English and Spanish. This covers approximately 90% of the dataset. All the Out-Of-Vocabulary (OOV) words were mapped to a special token (<*UNK*>).

The baseline PBSMT system is built on top of Moses (Koehn et al., 2007) with default settings. Moses is an SMT decoder that enables automatic training of TMs for any language pair using a large parallel corpus. Once the model is trained, an efficient search algorithm finds the highest probability translation among the exponential number of candidates. Training the Moses decoder yields a phrase model as well as a TM which, together, support translation between source and target languages. Moses scores each phrase in the phrase table with respect to a given source sentence and produces the best-scored phrases as output.

For the training phase of SMT, we apply the following steps:

- Tokenization: Insert spaces and punctuation between words.
- True-casing: Convert initial words in each sentence to their most probable casing, to reduce data sparsity.
- Cleaning: Remove both long and empty sentences, as they may cause misalignment issues within the training pipeline.

The LM is built with the target language (in our case-study, Spanish is the target language) to ensure fluent and well-formed output. KenLM (Heafield, 2011), which comes bundled with the Moses toolkit, is used for building our LM. Also to train the TM, GIZA++ (Och and Ney, 2003) is used for word alignment. Finally, the phrases are extracted and scored as well. The generated phrase table is later used to translate test sentences that are compared to the results of the LSTM encoder-decoder.

The following steps are applied to build the NMT system with the LSTM Encoder-Decoder:

- Vocabulary building: Generate vocabulary corpus for both source and target sides.
- Corpus shuffle: Shuffle the vocabulary corpus of both source and target languages.
- Dictionary building: Create dictionary by leveraging an alignment file to replace the <*UNK*> words.

The files mentioned above are fed to the LSTM encoder for the training phase of the NMT system. The number of epochs is set to 1000. LSTM cells are used with the Adam optimizer (Kingma and Ba, 2014), also 0.001 and 512 are as error rate and batch size, respectively. The validation split of 0.1 is used for the training as well. After the training step, the LSTM decoder will be used to translate given sentences.

In order to analyze the improvement of the performance of LSTM encoder-decoder over the SMT system analyzing the scores of the phrase pairs, we did the same as (Cho et al., 2014); selecting those phrase pairs that are scored higher by the LSTM encoder-decoder compared to the

[4] These two corpora are quite noisy.

[5] Word counts refer to Spanish words, after tokenization.

SMT system with respect to a given source sentence. The scoring of the phrases provided by the LSTM encoder-decoder is similar to the scoring of the phrases provided by the phrase table of the SMT system as the quality of the translation is approximately the same.

We employ BLEU (Papineni et al., 2001) as the evaluation metric. BLEU is calculated for individual translated segments by comparing them with a dataset of reference translations. The scores, between 0 and 100 are averaged over the whole evaluation dataset to reach an estimate of the translation overall quality. Table 1 shows the results on both development set as well as test set.

Model	**BLEU-dev**	**BLEU-test**
PBSMT	30.84	28.51
NMT-baseline	31.95	30.53
LSTM	33.49	31.65

Table 1: BLEU scores computed on the development and test sets using PBSMT, NMT-baseline and LSTM systems.

LSTM encoder-decoder scores indicate in overall improvement in translation performance in terms of BLEU scores. As seen in Table 1, our LSTM encoder-decoder outperforms the NMT-baseline by 1.12 BLEU points while it outperforms the PBSMT by 3.14 BLEU points. Our LSTM encoder-decoder is able to score a pair of sequences (in terms of a conditional probability) or to generate a target sequence given a source sequence.

We evaluated the proposed model with the SMT task, using the LSTM encoder-decoder to score each phrase pair in the phrase table. Qualitatively, we showed that this model captures linguistic regularities in the phrase pairs and also that the LSTM encoder-decoder proposes well-formed target phrases. We also found that the LSTM encoder-decoder contribution is orthogonal to the existing use of neural networks in the SMT system. We conclude that further performance improvements are likely if we were to use the LSTM encoder-decoder and the neural LM together.

7 Conclusions and Future Work

In this paper, we described a PBSMT implementation within which the phrase table is generated by using an LSTM encoder-decoder, and sentences with the highest scores are selected as output. We compared the resulting translation quality of the LSTM against PBSMT and a NMT baseline, and demonstrated a BLEU score increase of up to 3.14 and 1.12, respectively. We also noticed that SMT works well for long sentences while NMT works well for short sentences.

Since NMT systems usually have to apply a certain-sized vocabulary to avoid time-consuming training and decoding, such systems suffer from OOV issues. Furthermore, NMT lacks a mechanism to guarantee/control translation of all the source words and favors short translations, resulting in fluent but inadequate translations.

Issues outlined above are fodder for future work. For example, the incorporation of SMT features into the NMT model within the log-linear framework may be a future next step to address the training/decoding issue above.

Acknowledgments

We would like to express our sincere gratitude to Dr. Michael W. Mislove (Tulane University, USA) for all his support. We also would like to acknowledge the financial support received from the School of Science and Engineering, Tulane University (USA). In addition, we are grateful to the RANLP anonymous reviewers for all their helpful comments and discussions.

References

Benyamin Ahmadnia, Parisa Kordjamshidi, and Gholamreza Haffari. 2018. Neural machine translation advised by statistical machine translation: The case of farsi-spanish bilingually low-resource scenario. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications*. pages 1209–1213.

Benyamin Ahmadnia and Javier Serrano. 2015. Hierarchical phrase-based translation model vs. classical phrase-based translation model for spanish-english statistical machine translation system. In *Proceedings of the 31st Conference of the Spanish Society for Natural Language Processing*.

Benyamin Ahmadnia, Javier Serrano, and Gholamreza Haffari. 2017. Persian-spanish low-resource statistical machine translation through english as pivot language. In *Proceedings of Recent Advances in Natural Language Processing*. pages 24–30.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. pages 1044–1054.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Machine Learning Research* 3(19):1137–1155.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of Advances in Neural Information Processing Systems*. pages 1853–1861.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2019. Neural machine translation with reordering embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. pages 1787–1799.

Kyunghyun Cho, Bart Van merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. pages 1724–1734.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 1370–1380.

Jinhua Du and Andy Way. 2017. Pre-reordering for neural machine translation: Helpful or harmful? *Prague Bull. Math. Linguistics* 108:171–182.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. pages 1700–1709.

Shin Kanouchi, Katsuhito Sudoh, and Mamoru Komachi. 2016. Neural reordering model considering phrase translation and word alignment for phrase-based translation. In *Proceedings of the 3rd Workshop on Asian Translation 2016*. pages 94–103.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn. 2009. *Statistical Machine Translation*. Cambridge University Press.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, and Richard Zens. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the Association for Computational Linguistics*. pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. pages 48–54.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. pages 133–139.

Tomas Mikolov, Stefan Kombrink, Luks Burget, Jan Cernock, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*. pages 5528–5531.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. pages 160–167.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. pages 311–318.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of the 24th International Conference on Computational Linguistics*. pages 1071–1080.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of Advances in Neural Information Processing Systems*. pages 801–809.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 14–25.

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017. Translating phrases in neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1421–1431.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*. pages 1393–1398.

Automatic Propbank Generation for Turkish

Koray Ak
Department of Computer Engineering
Işık University
İstanbul, Turkey
koray.ak@isik.edu.tr

Olcay Taner Yıldız
Department of Computer Engineering
Işık University
İstanbul, Turkey
olcaytaner@isikun.edu.tr

Abstract

Semantic role labeling (SRL) is an important task for understanding natural languages, where the objective is to analyse propositions expressed by the verb and to identify each word that bears a semantic role. It provides an extensive dataset to enhance NLP applications such as information retrieval, machine translation, information extraction, and question answering. However, creating SRL models are difficult. Even in some languages, it is infeasible to create SRL models that have predicate-argument structure due to lack of linguistic resources. In this paper, we present our method to create an automatic Turkish PropBank by exploiting parallel data from the translated sentences of English PropBank. Experiments show that our method gives promising results.

1 Introduction

Semantic role labeling (SRL) is a well defined task that identifies semantic roles of the words in a sentence. Event characteristics and participants are simply identified by answering "Who did what to whom" questions. Having this semantic information facilitates NLP applications such as machine translation, information extraction, and question answering. After the development of statistical machine learning methods in the area of computational linguistics, learning complex linguistic knowledge has became feasible for NLP applications. Recent semantic resources specifically for SRL which provides input for developing statistical approaches are FrameNet (Fillmore et al., 2004), PropBank (Kingsbury and Palmer, 2002) (2003), (2005), (Bonial et al., 2014) and NomBank (2004). These resources enables us to understand language structure by providing a stable semantic representation.

Among these resources PropBank is a commonly used semantic resource which includes predicate - argument structure by stating the roles that each predicate can take along with the annotated corpora. It has been applied to more than 15 different languages. However, manually creating such semantic resource is labor-intensive, time-consuming and most importantly requires a professional linguistic perspective. Also limited linguistic data further blocks generating PropBank-like resources.

Various studies such as Zhuang and Zong (2010), Van der Plas et al. (2011) (2014), Kozhevnikov and Titov (2013), Akbik et al. (2015), which transfer semantic information using parallel corpus, are presented to cope with these problems. In this way, semantic information projected from a resource-rich language (English) to a language with inadequate resources and PropBank of the target language is automatically generated. Here the assumption is translated parallel sentences generally share same semantic information. Word and constituent based alignment techniques are widely used to construct mapping between source and target languages for annotation projection. Previous studies report translation divergences and language specific differences affect the quality of the projection. Filtering projections using learning methods is suggested to increase precision. In this paper, we present our study to create automatic Turkish PropBank using parallel sentences from English PropBank.

This paper is organized as follows: we first give brief information about English and Turkish PropBanks in Section 2. In Section 3, Studies for the automatic proposition bank generation are discussed. In the next section proposed methods are presented. First, we explain the annotation projection using parallel sentence trees. Then, we propose methods for aligning parallel sentence phrases not aligned with tree structure. Finally, in

Section 5, we conclude with the results.

2 PropBank

2.1 English PropBank

PropBank is the bank of propositions where predicate-argument information of the corpora is annotated and semantic roles or arguments that each verb can take are posited. It is constituted on the Penn Treebank (Marcus et al., 1993) Wall Street Journal [WSJ]. The primary goal is to label syntactic elements in a sentence with specific argument roles to standardize labels for the similar arguments such as *the window* in John broke *the window* and *the window* broke. PropBank uses conceptual labels for arguments from Arg0 to Arg5. Only Arg0 and Arg1 indicate the same roles across different verbs where Arg0 means agent or causer and Arg1 is the patient or theme. The rest of the argument roles can vary across different verbs. They can be instrument, start point, end point, beneficiary, or attribute.

Moreover, PropBank uses ArgM's as modifier labels where the role is not specific to the verb group and generalizes over the corpora such as location, temporal, purpose, or cause etc. arguments. The first version of English PropBank, named as The Original PropBank, is constructed for only verbal predicates whereas the latest version includes all syntactic realizations of event and state semantics by focusing different expressions in form of nouns, adjectives and multi-word expressions to represent complete event relations within and across sentences.

2.2 PropBank Studies for Turkish

There have been different attempts to construct Turkish PropBank in the literature. Şahin (2016a; 2016b), Şahin and Adalı (2017) report semantic role annotation of arguments in the Turkish dependency treebank. They construct PropBank by using ITU-METU-Sabancı Treebank (IMST). In these studies, frame files of Turkish PropBank are constructed and extended by utilizing crowdsourcing. 20,060 semantic roles are annotated in 5,635 sentences. The size of the resource is stated as a drawback in the study. Recently, Ak et al. (2018) construct another Turkish Proposition Bank using translated sentences of English PropBank. So far, 9,560 of 17,000 translated sentences are annotated with semantic roles. Also, framesets are created for 1,330 verbs and 1,914 verb senses. These studies constitute a base for Turkish proposition bank, but their size is limited and construction of these proposition banks consumed a lot of time.

3 Automatic PropBank Generation Studies

PropBanks are also generated automatically for resource-scarce languages by using parallel corpus. In this section, proposition bank studies for automatic generation are presented. Zhuang and Zong (2010) proposed performing SRL on parallel corpus of different languages and merging the result via a joint inference model can improve SRL results for both input languages. In the study an English and Chinese parallel corpus is used. First each predicate is processed by monolingual SRL systems separately for producing argument candidates. After the candidates formed, a Joint Inference model selects the candidate that is reasonable to the both languages. Also, a log-linear model is formulated to evaluate the consistency. This approach increased F1 scores 1.52 and 1.74 respectively for Chinese and English.

Van der Plas et al. (2011) presents cross-lingual semantic transfer from English to French. English syntactic-semantic annotations were transferred using word alignments to French language. French semantic annotations gathered from the first step were then trained with a French joint syntactic-semantic parser along with the French syntactic annotations trained separately. Joint syntactic-semantic parser is used for learning the relation between semantic and syntactic structure of the target language and reduces the errors arising from the first step. This approach reaches 4% lower than the upper bound for predicates and 9% for arguments.

Kozhevnikov (2013) shows SRL model transfer from one language to another can be achieved by using shared feature representation. Shared feature representation for language pairs is constructed based on syntactic and lexical information. Afterwards, a semantic role labeling model is trained for source language and then used for the target language. As a result SRL model of the target language is generated. Process only requires a source language model and parallel data to construct target SRL model. Approach is applied for English, French, Czech and Chinese languages.

In the next study, Van der Plas (2014) improves the labeling results with respect to the previous work

(Van der Plas et al., 2011) by building separate models for arguments and predicates. Also, problems of transferring semantic annotations using parallel corpus is examined in the paper. Token-to-token basis annotation transfer, translation shifts, and alignment errors in the previous work is replaced with a global approach that aggregates information at corpus level. Instead of using English semantic annotations of roles and predicate together with French PoS tags to generate French semantic annotations, English annotations of predicates and roles used separately to generate one predicate and one role semantic annotations separately.

Akbik *et al.* propose a two stage approach (Akbik et al., 2015). In the first stage only filtered semantic annotation is projected. Since high confidence semantic labels projected, resulting target semantic labels will be high in precision and low in recall. In the next stage, completed target language sentences sampled and a classifier is trained to add new labels to boost recall and preserve precision. Proposed system is applied on 7 different languages from 3 different language family. These languages are Chinese, Arabic, French, German, Hindi, Russian, and Spanish.

4 Methods

Among the studies for Turkish proposition bank, Ak et al. (2018) is constructed on parallel English - Turkish sentences from the Original English PropBank. We have used the corpus provided in this study to automatically generate proposition bank.

4.1 Automatic Turkish PropBank Using Parallel Sentence Trees

Penn Treebank structure offers advantages for building fully tagged data set in accordance with syntactic labels, morphological labels and parallel sentences. We used this structure to add English PropBank labels for each word in the corpus. In this manner, we exploited this parallel dataset to transfer English PropBank annotations to an automatic Turkish PropBank.

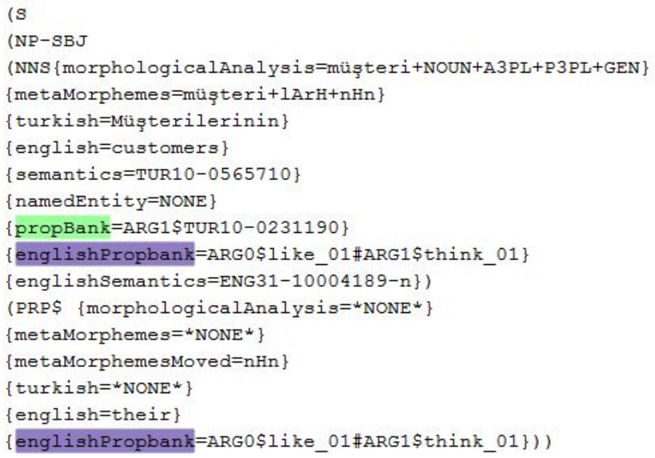
4.1.1 English PropBank Labels

Original English PropBank corpus (Palmer et al., 2004) is accessible through Linguistic Data Consortium (LDC). This resource is the initial version of the English PropBank and it only includes the relations with verbal predicates. In the newer versions adjective and noun relations are also annotated. Since we compare projection results with manually annotated corpus (Ak et al., 2018) which only contains verbal relations, we use the initial version of the English PropBank. We downloaded this dataset and imported annotations for the selected sentences. After this step 6,060 sentences among 9,558 were enhanced with the English annotations. Below in Figure 1, a sample sentence is presented. English annotations are inserted inside "englishPropbank" tags right after Turkish annotations which reside in "propbank" tags. Some of the words have only English annotation, because there is no word translated in the Turkish sentence for this node. As an example, "their" in Figure 1 has annotations in the englishPropbank tag but there is no equivalent translation in Turkish, presented as "*NONE*", so propbank tag does not exist. English tags have predicate information that annotation belongs to. "Müşterilerinin" (customers) in the same example has "ARG0$like_01#ARG1$think_01" in the englishPropbank tag which means there exists at least two words whose root is in verb form. Here the word is annotated with respect to "like" and "think" separately. We have separated multiple annotations with "#" sign and in each annotation predicate label and role is distinguished by "$" sign. In the Turkish annotation, WordNet id of the predicate was used instead of predicate label.

4.1.2 Transfering Annotations to Automatic Turkish PropBank Using Parallel Sentences

After importing English annotations, it is necessary to determine predicate(s) of the Turkish sentences. Morphological structures of the words are examined to detect predicate candidates. Words were morphologically and semantically analyzed in translated Penn TreeBank. We have used "morphologicalAnalysis" tag to check the morphological structure of the words. In Figure 1, sample morphological structure is displayed.

The word which has a verb root and verb according to last inflectional group is treated as the predicate of the sentence. Once we found a word suitable for these conditions, we gathered English PropBank annotation. If it is also labeled as predicate in English proposition bank, we got the predicate label, e.g. like_01, to find annotations with respect to this predicate. We searched for the found



```
(S
(NP-SBJ
(NNS{morphologicalAnalysis=müşteri+NOUN+A3PL+P3PL+GEN}
{metaMorphemes=müşteri+lArH+nHn}
{turkish=Müşterilerinin}
{english=customers}
{semantics=TUR10-0565710}
{namedEntity=NONE}
{propBank=ARG1$TUR10-0231190}
{englishPropbank=ARG0$like_01#ARG1$think_01}
{englishSemantics=ENG31-10004189-n})
(PRP$ {morphologicalAnalysis=*NONE*}
{metaMorphemes=*NONE*}
{metaMorphemesMoved=nHn}
{turkish=*NONE*}
{english=their}
{englishPropbank=ARG0$like_01#ARG1$think_01}))
```

Figure 1: Part of a sentence tree : English PropBank annotations reside in “englishPropBank” tags.

predicate label in the annotations and transfered annotations matching with the predicate label. If we could not find a predicate in Turkish sentence or the corresponding English label did not contain Predicate role annotation, we skipped to the next predicate candidate.

During the transfer, a mapping was needed due to the difference between English and Turkish (Ak et al., 2018) argument labeling. English PropBank corpus has “-” sign in ArgM’s like ARGM-TMP and also some of the arguments from Arg1 to Arg5 are labeled with the prepositions such as ARG1-AT, ARG2-BY etc. We processed these differences and then transferred labels into the “propbank” tags. After analyzing Turkish sentences we found out some sentences have more than one predicate, so we continued to search for another predicate in the sentence and ran the same procedure for each predicate candidate.

4.1.3 Experiments

Annotations gathered from the English sentence were compared with the Turkish hand-annotated proposition bank (Ak et al., 2018). Comparisons were done at the word level by checking the annotations for each corpus. Among the 6,060 sentences enhanced with English PropBank roles, 848 sentences did not have a predicate in Turkish proposition bank. Therefore, in 5,212 sentences, 44,779 word annotations were compared. 31,813 annotations were transferred from English to Turkish. Results of the comparison are presented in Table 1. 19,373 words annotated with PropBank roles correctly . 6,441 annotations are incorrect, PropBank tags are different in both corpus. 5,999 annotations are undetermined, valid PropBank labels transferred from English annotations but no annotation exists in hand annotated proposition bank. Annotations to be compared is not valid so we did not include this set in the evaluation.

Transferred		Untransferred	
Correct Incorrect Undetermined	19,373 6,441 5,999	Not H.A. H.A.	8,837 4,129
Total	31,813	**Total**	12,966

Table 1: Results of the comparison between automatic proposition bank and hand annotated (H.A.) proposition bank.

When we remove undetermined 5,999 words in the comparison; 19,373 annotations from 25,814 annotations are correct, which gives us ~75% accuracy for transferred and comparable set. These 5,999 annotations may be hand-annotated and recompared for validity of the transferred annotations.

In Table 2, we present occurrences of erroneous annotation transfers. Only top ten occurrences are presented. Arg0-Arg1 transfers are the most occurred incorrect transfers 1,843 among 6,441 incorrect annotations. Second most occurred error is in Arg1-Arg2 labels. Errors in Arg0-Arg1 and Arg1-Arg2 labels forms ~44% of the transfer errors.

On the other hand, when we look at the all word results, 12,966 roles were not transferred. If we take these untransferred instances as incorrect; 19,373 annotations out of 38,780 annotation are true and the accuracy drops to ~50%. However, 8,837 of untransferred annotation are not an-

Different Arguments	# of Occurrence
ARG0-ARG1	1,843
ARG1-ARG2	961
ARG2-ARGMEXT	462
ARG1-PREDICATE	255
ARG0-ARG2	229
ARG4-ARGMEXT	226
ARG1-ARGMPNC	220
ARG1-ARGMMNR	186
ARG1-ARGMTMP	160
ARG1-ARGMLOC	148

Table 2: Counts of different argument annotations between transferred annotations and hand annotations.

notated in the hand-annotated corpus. Only 4,129 are valid PropBank arguments. In this respect, if we count only valid arguments for untransferred annotations, accuracy is ~65%.

4.2 Automatic Turkish PropBank Using Parallel Sentence Phrases

In the previous method, annotation projection using parallel sentence trees is discussed. However, finding such a resource in a special format is difficult especially if you are working with a resource-scarce language. Most of the time creating a formatted parallel resource like tree structured sentences complicates translation procedure. In this section, automatic generation with translated sentences without tree structure will be examined.

4.2.1 Phrase Sentence Structure

For the phrase sentences, English sentences re-translated without tree structure. Prior the annotation projection, linguists in the team annotated phrase sentences and populated “propbank” and “shallowParse” tags so that we check the correctness after the annotation transfer. 6,511 sentences among 9,557 phrase sentences have predicate according to hand annotations for newly translated sentences. However, only 5,259 sentences have English PropBank annotation, so we take this set to transfer annotations. As you remember, the same number in the previous section was 5,212. Here translation and annotation differences change the processed sentence count.

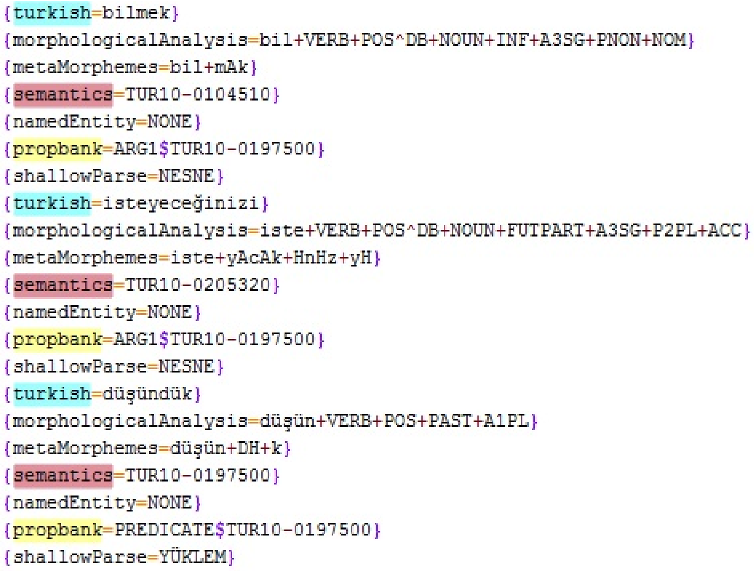
Tag structure of Penn Treebank is preserved to simplify morphologic and semantic analysis requirements during the annotation transfer. In Figure 2, sample phrase sentence can be seen. Unlikely Figure 1, syntactic tags which indicate tree structure are not included. We used original tree formatted English sentence to extract English propbank annotations. However, since the target sentence do not have tree structure definition we used other word alignment methods to determine annotation projection.

4.2.2 Semantic Alignment Using WordNet

In order to transfer annotations, first we tried to match predicates of English sentence and Turkish translation. Again we utilize “morphological-Analysis” tags to determine predicate candidates in the phrase sentence. Words which have a verb root and verb according to last inflectional group is treated as the predicate candidates of the sentence. Once we found all the words ensuring these conditions, we gathered all English PropBank annotation labels which are tagged as “Predicate” in ‘englishPropbank” tag. To align predicates in different languages, we tried to exploit WordNet’s (Ehsani et al., 2018) interlingual mapping capabilities. For each predicate in English sentence we find Turkish translation by searching English synset id in the WordNet. English synset id is located in englishSemantics tags as in the sample in Figure 1. If there exists any translation in the WordNet, we take Turkish synset id and search it in the predicate candidates found for phrase sentence. Whenever translation found, we align predicates and try to transfer annotation with respect to aligned English label. For annotation transfer of other arguments we again align words using WordNet’s interlingual mapping. An example WordNet record is presented in Figure 3.

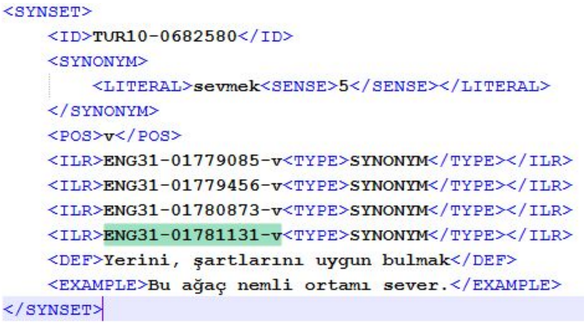
First results gathered with only WordNet mapping were very low. True annotation count is 2,195 among 29,168 annotations tagged manually which yields 7.53%. However, transferred false annotation count is only 342. System heavily relies on semantic annotations for both English and Turkish words where some of the words failed to have semantic annotation. We look deeper into dataset provided by Ak et al. (2018), 11,006 English words do not have semantic annotation so we failed to match these words with Turkish counterparts.

Some words are not annotated semantically such as, proper nouns, time, date, numbers, ordinal numbers, percentiles, fractional numbers, number intervals, and reel numbers. Most of these words are same in Turkish translation so we matched English and Turkish words by string match. For example if a sentence contains proper



```
{turkish=bilmek}
{morphologicalAnalysis=bil+VERB+POS^DB+NOUN+INF+A3SG+PNON+NOM}
{metaMorphemes=bil+mAk}
{semantics=TUR10-0104510}
{namedEntity=NONE}
{propbank=ARG1$TUR10-0197500}
{shallowParse=NESNE}
{turkish=isteyeceğinizi}
{morphologicalAnalysis=iste+VERB+POS^DB+NOUN+FUTPART+A3SG+P2PL+ACC}
{metaMorphemes=iste+yAcAk+HnHz+yH}
{semantics=TUR10-0205320}
{namedEntity=NONE}
{propbank=ARG1$TUR10-0197500}
{shallowParse=NESNE}
{turkish=düşündük}
{morphologicalAnalysis=düşün+VERB+POS+PAST+A1PL}
{metaMorphemes=düşün+DH+k}
{semantics=TUR10-0197500}
{namedEntity=NONE}
{propbank=PREDICATE$TUR10-0197500}
{shallowParse=YÜKLEM}
```

Figure 2: Part of a phrase sentence : Translated words in Turkish tags. Helper tags gives additional information for each word.



```
<SYNSET>
    <ID>TUR10-0682580</ID>
    <SYNONYM>
        <LITERAL>sevmek<SENSE>5</SENSE></LITERAL>
    </SYNONYM>
    <POS>v</POS>
    <ILR>ENG31-01779085-v<TYPE>SYNONYM</TYPE></ILR>
    <ILR>ENG31-01779456-v<TYPE>SYNONYM</TYPE></ILR>
    <ILR>ENG31-01780873-v<TYPE>SYNONYM</TYPE></ILR>
    <ILR>ENG31-01781131-v<TYPE>SYNONYM</TYPE></ILR>
    <DEF>Yerini, şartlarını uygun bulmak</DEF>
    <EXAMPLE>Bu ağaç nemli ortamı sever.</EXAMPLE>
</SYNSET>
```

Figure 3: Sample WordNet record found by searching “ENG31-01781131-v”, English synset id, from the sentence in Figure 1.

noun “Dow Jones”, the same string also exists in the Turkish translation too. However, it may take additional suffixes, so we only check whether English words starts with Turkish root word. Also, translational differences are encountered like decimal separator in English is “.” where some Turkish translations “,” is used. We replace this differences by looking whether the first morphological tag is “NUM”. After these tunings, we rerun the procedure and get 2,680 true and 531 false annotations which increases true annotations to 9.19%. Another problem is erroneous semantic annotations. If English and Turkish semantic annotation is not right, alignment is not possible. Even in the best scenario where both word is annotated, if WordNet mapping is incomplete, an alignment can not be established.

As an alternative we decided to reinforce annotation transfer by using constituent boundaries identified with shallowParse tags by our linguist team mates. Example of shallowParse tags can be seen in Figure 2. Prior to the annotation transfer, phrase sentences are annotated for constituent boundaries which can be used to group argument roles in the sentence. After transferring annotations with respect to semantic annotations, we run another method over phrase sentences which calculates maximum argument types for each constituent and tags any untagged word with the calculated max argument role within the constituent boundary. This procedure further enhance true annotations to 4,255 but also increase false annotations to 1,202. After constituent boundary calculation, correct annotation transfer percent is increased to ~14.59%. In Figure 4 annotation of the sentence 7076.train is presented. Untagged words in “Özne” and “Zarf Tümleci” constituent boundaries are tagged with the found argument role within the boundary. Note that, we did not use the constituent types but we use boundaries of the constituents.

4.2.3 Word Alignment Using IBM Alignment Models

Word alignment through semantic relation requires fair semantic annotation for both languages and also sufficient semantic mapping between languages. We search different word alignment methods between English and Turkish sentences. IBM

(1) [The less-rigorous Senate version]$_{\text{ARG0}}$ [would]$_{\text{ARGM-MOD}}$ [defer]$_{\text{Predicate}}$ [the deductibility]$_{\text{ARG1}}$ for [roughly five years.]$_{\text{ARG2-for}}$

(2) [Daha az sıkı türden bir Senato versiyonu]$_{\text{Özne - Subject}}$ [aşağı yukarı beş yıl için]$_{\text{Zarf Tümleci - Adverbial Clause}}$ [düşülebilirliği]$_{\text{Nesne - Object}}$ [ertelerdi.]$_{\text{Yüklem - Predicate}}$

(3) [Daha az sıkı türden bir]$_{\text{NONE}}$ [Senato]$_{\text{ARG0}}$ [versiyonu]$_{\text{NONE}}$ [aşağı yukarı]$_{\text{ARG2}}$ [beş]$_{\text{NONE}}$ [yıl]$_{\text{ARG2}}$ [için düşülebilirliği]$_{\text{NONE}}$ [ertelerdi.]$_{\text{PREDICATE}}$

(4) [Daha az sıkı türden bir Senato versiyonu]$_{\text{ARG0}}$ [aşağı yukarı beş yıl için]$_{\text{ARG2}}$ [düşülebilirliği]$_{\text{NONE}}$ [ertelerdi.]$_{\text{PREDICATE}}$

Figure 4: Annotation reinforced with respect to constituent boundaries: (1) English sentence (2) constituent boundaries identified with shallow-Parse tags for sentence in 7076.train, (3) Argument roles for the same sentence after annotation transfer, (4) Argument roles for the same sentence after reinforce method.

English Word	Turkish Word	Probability
Reserve	Reserve	0.72270917
Reserve	Rezerv	0.15328414
Reserve	mevduat	0.03056293
Reserve	Bankası'nın	0.02731664
Reserve	kuruluşlarındaki	0.01375332
Reserve	komisyonları	0.01375332
Reserve	Bankasının	0.00611259
Reserve	kuruluşlarında	0.00458444
Reserve	komisyon	0.00458444
Reserve	Federe	0.00458444

Table 3: Word alignment probabilities for English word "Reserve" calculated by IBM Model 1.

English Word	Turkish Word	Probability
Reserve	Reserve	0.67700755
Reserve	Rezerv	0.14360766
Reserve	Federe	0.06154614
Reserve	Bankası	0.05265972
Reserve	tasarruf	0.03072182
Reserve	kuruluşlarına	0.02117394
Reserve	üzerindeki	0.01111856
Reserve	bu	0.00212005
Reserve	kurumlarına	0.00004452
Reserve	Merkez	0.00000002

Table 4: Word alignment probabilities for English word "Reserve" calculated by IBM Model 2.

alignment models offer solution to our word alignment problem. IBM Models are mainly used for statistical machine translation to train a translation model and an alignment model. IBM Model 1 (Brown et al., 1993) is the primary word alignment model offered by IBM. It is widely used for solving word alignments while working with parallel corpora. It is a generative probabilistic model that calculates probabilities for each word alignment from source sentence to target sentence. It takes a corpus of paired sentences from two languages as training data. These paired sentences are possible translation of the sentences from source language to target. With this training corpus, parameters of the model estimated using EM (expectation maximization). IBM Model 2 has an additional model for alignment and introduce alignment distortion parameters. We decided to use IBM model 1 & 2 to establish word alignments instead of WordNet's interligual mapping. We input sentence pairs and gather alignment probabilities for each English word to Turkish equivalent. 244,024 word pairs are taken as output where for each English word, 10 most probable Turkish words are listed. Alignment probabilities for word "Reserve" is presented in Table 3 and 4 for IBM Model 1 and 2 respectively.

After gathering alignment data, we transfer annotations to phrase sentences from English PropBank labels in the tree structured sentences. All words tagged with "PREDICATE" tag in English sentence are stored into a map which includes predicate label from the "englishPropbank" tag *e.g.* "like_01" and English word from the "english" tag *e.g.* "like". Then we search alignments for each found English predicate. Here we observed that aligned Turkish words may not occur in the phrase sentence as they found in the alignment table. Words may include additional suffixes, so we use Finite State Machine(FSM) morphological analyzer available in our NLP Toolkit of Ak et al. (2018) to extract roots of the aligned Turkish words. Since we have several possible morphological parse for each aligned word, we created an array for possible roots. In parallel, we found predicate candidates from the phrase sentence as we stated in the previous methods. Then we tried to match aligned words and possible roots with the found predicate candidates. If there exists a predicate candidate that matches with the aligned word or one of its roots in the array, we tagged the candidate as "PREDICATE" and update map as predicate label and synset id of Turkish predicate.

After finishing predicate discovery, we transfer annotations for found predicates. To do that we look for the annotations with respect to the predicate labels in the map. For each record in map we took the predicate label and corresponding Turkish synset id. When we found an annotation with this predicate label, first we extract the argument and try to find aligned word for the processed English word. For the alignment again we find the most probable word from the table and use FSM morphological analyzer to extract possible roots. Then for each word we search Turkish sentence to match words with aligned word or possible roots extracted. If matched Turkish words do not have argument annotation, we transfer argument with the synset id found in the map record.

As we discuss in the previous annotation transfer procedure 4.2.2, some of the English words such as proper nouns, time, date, numbers, ordinal numbers, percentiles, fractional numbers, number intervals, and reel numbers stay same or take additional suffixes in Turkish translation. So we include the same method used for matching these words. In a case words are not aligned with the information from alignment table, and a valid annotation present in English word, we search exact string match or any word starts with the root of English word in the Turkish sentence.

We run our procedure with IBM Model 1 & 2 separately. We add reinforce step previously used in Section 4.2.2. Unlikely previous attempts, after examining language structure we decided to add rules to tag any untagged words after annotation transfer. We observed argument types affect noun inflections, for some argument types the last word in constituent boundary is taking certain suffixes. So first we find untagged word and select the last word in its constituent boundary. Since we run reinforce step beforehand, only untagged constituents exists in the sentence. In this respect, we set the following rules to determine argument annotation for untransfered words;

- For nouns and proper nouns:
  - Have no suffix then ARG0
  - Last morpheme tag is "ACCUSATIVE" (-(y)H, -nH) or "DATIVE" (-(y)A, -nA) then ARG1
  - Last morpheme tag is "LOCATIVE" (-DA, -nDA) or "ABLATIVE" (-DAn, -nDAn) then ARGMLOC
  - Last morpheme tag is "INSTRUMENTAL" (-(y)lA) then ARG2
- For all word types
  - Morphological parse contains date, time then ARGMTMP
  - Morphological parse contains cardinal number, fraction, percent, range, real number, ordinal number then ARGMEXT

We use these rules to tag any untagged word. After applying these rules annotation transfer result is as shown in Table 5 and 6. Results show that rules applied slightly change the correct annotations. For model 1 rules output much more correct annotation than the incorrect ones whereas in model 2 the number of correct and incorrect annotations gathered are nearly same. However, precision for model 1 is improved to 59.44% and for model 2 precision become 59.86%.

IBM Model 1 + Reinforce + Rules			
Transferred		**Untransferred**	
Correct Incorrect Undetermined	17,340 9,664 14,384	Not H.A. H.A.	1,151 2,170
Total	41,388	**Total**	3,321

Table 5: Results for IBM Model 1 alignment.

IBM Model 2 + Reinforce + Rules			
Transfered		**Untransfered**	
Correct Incorrect Undetermined	17,464 9,635 14,457	Not H.A. H.A.	1,078 2,075
Total	41,556	**Total**	3,153

Table 6: Results for IBM Model 2 alignment.

5 Conclusion

We proposed methods to generate automatic Turkish proposition bank by transferring cross-language semantic information. Using the parallelism with English proposition bank gives us an opportunity to create a proposition bank in a short time with less effort. We currently have 64% accuracy with the hand-annotated proposition bank (Ak et al., 2018) for parallel sentence trees. When we consider only transferred annotations, accuracy is rising to ~75%. We also present annotation projection to phrase sentences using WordNet and IBM alignment models. WordNet alignment heavily relies on semantic annotations, correct annotations transferred after this method is ~14.59%. However, 4,255 correct argument roles are transferred among 5,457 arguments which means 79% of the transferred roles are correct. To increase annotation transfer for phrase sentences, we have also proposed alignment with IBM Model 1 and 2. Both models yields ~60% correct annotations. Annotations transferred with these methods can provide a basis for proposition bank creation in resource-scarce languages. Annotations may then be checked quickly by the annotators and proposition bank reach the final state.

References

Meyers A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 24–31.

K. Ak, O. T. Yıldız, V. Esgel, and C. Toprak. 2018. Construction of a Turkish proposition bank. *Turkish Journal of Electrical Engineering and Computer Science* 26:570 – 581.

A. Akbik, L. Chiticariu, M. Danilevsky, Y. Li, S. Vaithyanathan, and H. Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *ACL (1)*. The Association for Computer Linguistics, pages 397–407.

C. Bonial, J. Bonn, K. Conger, J. D. Hwang, and M. Palmer. 2014. Propbank: Semantics of new predicate types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.* 19(2):263–311.

G. G. Şahin. 2016a. Framing of verbs for turkish propbank. In *TurCLing 2016 in conj. with 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2016)*.

G. G. Şahin. 2016b. Verb sense annotation for turkish propbank via crowdsourcing. In *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2016)*.

G. G. Şahin and E. Adalı. 2017. Annotation of semantic roles for the Turkish proposition bank. *Language Resources and Evaluation* .

R. Ehsani, E. Solak, and O. T. Yıldız. 2018. Constructing a wordnet for Turkish using manual and automatic annotation. *ACM Transactions on Asian Low-Resource Language Information Processing* 17(3).

C. J. Fillmore, J. Ruppenhofer, and Collin F. Baker. 2004. *FrameNet and Representing the Link between Semantic and Syntactic Relations*, Institute of Linguistics, Academia Sinica, Taipei, pages 19–62. Language and Linguistics Monographs Series B.

P. Kingsbury and M. Palmer. 2002. From treebank to propbank. In *LREC*. European Language Resources Association.

P. Kingsbury and M. Palmer. 2003. Propbank: The next level of treebank. In *Proceedings of Treebanks and Lexical Theories*. Växjö, Sweden.

M. Kozhevnikov and I. Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *ACL (1)*. The Association for Computer Linguistics, pages 1190–1200.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.* 31(1):71–106.

M. Palmer, P. Kingsbury, O. Babko-Malaya, S. Cotton, and B. Snyder. 2004. Proposition bank i. Philadelphia: Linguistic Data Consortium. LDC2004T14.

L. Van der Plas, M. Apidianaki, and C. Chen. 2014. Global methods for cross-lingual semantic role and predicate labelling. In *COLING*. ACL, pages 1279–1290.

L. Van der Plas, P. Merlo, and J. Henderson. 2011. Scaling up automatic cross-lingual semantic role annotation. In *ACL (Short Papers)*. The Association for Computer Linguistics, pages 299–304.

T. Zhuang and C. Zong. 2010. Joint inference for bilingual semantic role labeling. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '10, pages 304–314.

Multilingual Sentence-Level Bias Detection in Wikipedia

Desislava Aleksandrova [1,2] **François Lareau** [1] **Pierre-André Ménard** [2]
[1] OLST, Université de Montréal, first.lastname@umontreal.ca
[2] Computer Research Institute of Montreal, first.lastname@crim.ca

Abstract

We propose a multilingual method for the extraction of biased sentences from Wikipedia, and use it to create corpora in Bulgarian, French and English. Sifting through the revision history of the articles that at some point had been considered biased and later corrected, we retrieve the last tagged and the first untagged revisions as the before/after snapshots of what was deemed a violation of Wikipedia's neutral point of view policy. We extract the sentences that were removed or rewritten in that edit. The approach yields sufficient data even in the case of relatively small Wikipedias, such as the Bulgarian one, where 62k articles produced 5k biased sentences. We evaluate our method by manually annotating 520 sentences for Bulgarian and French, and 744 for English. We assess the level of noise and analyze its sources. Finally, we exploit the data with well-known classification methods to detect biased sentences. Code and datasets are hosted at https://github.com/crim-ca/wiki-bias.

1 Introduction

Our goal is to automatically detect neutral point of view (NPOV) violations at the sentence level with a procedure replicable in multiple languages. Sentence-level bias detection is a type of sentiment analysis, closely related to subjectivity detection (Riloff and Wiebe, 2003; Wiebe and Riloff, 2005; Wilson and Raaijmakers, 2008; Murray and Carenini, 2009; Lin et al., 2011; Al Khatib et al., 2012), where an opinion is considered subjective, and a fact, objective. Yet, as far as bias in writing is concerned, both subjective opinions and objective fact reporting (cf. §5) may, in some cases, be sources of impartiality. The importance of the context is one of the main difficulties in detecting bias at the sentence level. Some types of point-of-view bias are equally challenging for humans to detect. Partisanship in editorials, for example, tends to go unnoticed when in line with the reader's own ideas and beliefs (Yano et al., 2010). A further complication arises from the ambiguity of the term bias, which stands for a lack of fairness or neutrality in realms as varied as human cognition (Tversky and Kahneman, 1974), society (Ross et al., 1977), media (Entman, 2007), internet (Baeza-Yates, 2018; Pitoura et al., 2018) or statistical models and algorithms (O'Neil, 2016; Shadowen, 2019), to name a few. With so many different types of bias and their varying definitions, it is not trivial to set the scope of a bias-detection study.

The majority of the work on this task is performed on news articles (Hirning et al., 2017; Baly et al., 2018; Bellows, 2018) and political blogs (Yano et al., 2010; Iyyer et al., 2014) rather than Wikipedia, because of the relative scarcity of examples an encyclopedia provides. Yet, unlike alternative data sources, Wikipedia comes with a definition of bias outlined in its content policy for neutrality of point of view (NPOV). The core guidelines in NPOV are to: (1) avoid stating opinions as facts, (2) avoid stating seriously contested assertions as facts, (3) avoid stating facts as opinions, (4) prefer nonjudgemental language, and (5) indicate the relative prominence of opposing views. In addition, Wikipedia provides lists of bias-inducing words to avoid,[1] such as positively loaded language (puffery) in the form of *peacock* words (e.g., *best, great, iconic*); unsupported attributions, or *weasel words* (e.g., *some people say,*

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch

it is believed, science says); uncertainty markers, known as *hedges* (e.g., *very, much, a bit, often, approximately*), editorializing (e.g., *without a doubt, arguably, however*) and more. When an article is considered biased, an editor can flag it by adding a tag such as {{POV}} to its source, which displays a disputed neutrality warning banner on the page. These explicit guidelines (and the editors who apply them) help reduce biased language in Wikipedia over time through a continuous process of collaborative content revision (Pavalanathan et al., 2018). Still, new instances of bias are introduced just as often as old ones are overlooked because of humans' inherent difficulty with subtle expressions of point-of-view partiality. Recasens et al. (2013) showed that when presented with a biased sentence from Wikipedia, annotators manage to correctly identify the loaded word in only 37% of the cases.

2 Related Work

Bias detection approaches vary primarily in terms of corpora, vectorization methods, and classification algorithms. We present a review of the related literature along this division.

2.1 Corpora

Among those who tackle NPOV violations in Wikipedia, some rely on available datasets (Vincze, 2013), others perform manual annotation (Hube and Fetahu, 2018; Ganter and Strube, 2009; Herzig et al., 2011; Al Khatib et al., 2012), still others attempt to automatically extract labeled examples (Ganter and Strube, 2009; Recasens et al., 2013; Hube and Fetahu, 2018). Our approach is in line with the latter.

Using existing corpora, while being the cheapest method, predetermines which types of bias will be explored and in which languages. Vincze (2013) uses WikiWeasel, the Wikipedia subset of the CoNLL-2010 Shared Task corpora (Farkas et al., 2010) to study discourse-level uncertainty by manually annotating linguistic cues for three overt manifestations of bias: weasel, hedge and peacock words. Ganter and Strube (2009) focus on detecting hedges in a corpus of 1000 extracted sentences tagged with {{weasel}}, Bhosale et al. (2013) try to detect promotional content, while Kuang and Davison (2016) train their model on the English corpus of Recasens et al. (2013).

Manual annotation ensures higher quality but is too costly for large multilingual datasets. Hube and Fetahu (2018) learn to detect bias in Wikipedia on a manually annotated corpus of sentences from the inherently biased Conservapedia, with a precision of 0.74. When tested on an unlabeled dataset extracted from Wikipedia however, the classifier obtains a precision of 0.66 for the sentences classified with a certainty over 0.8.

Recasens et al. (2013) first propose a heuristic to automatically build a labeled corpus with biased sentences. Out of all revisions of NPOV-tagged articles, they identify the bias-driven edits based on the comments the editors left at commit. Although reliable, this method yields a fairly small set of examples for English (2,235 sentences) and none for smaller Wikipedias, first because of its dependence on revision comments (which are optional), and second, because it limits the examples to bias-driven edits containing five or fewer words.

2.2 Vectorization

As for data vectorization, previous work on bias detection relies either on features from pre-trained language models, custom feature-engineering or both. Bellows (2018) finds no significant difference in performance for classifiers trained on Word2vec, GloVe, or fastText representations. Several studies (Recasens et al., 2013; Ganter and Strube, 2009; Hube and Fetahu, 2018) employ multiple lexical, contextual, and linguistic features which, while boosting performance, remain dependant on handcrafted word lists, specialized lexical resources such as SentiWordNet (Baccianella et al., 2010), subjClue (Gitari et al., 2015), etc., and grammatical parsers that often cover only English. Yano et al. (2010) combine word vector representations from GloVe (as semantic features), 32 boolean lexicon-based features from Recasens et al. (2013) and document vector representations (as contextual features) to distinguish between different uses of the same word. They find that when training a logistic regression classifier, the semantic features alone perform better than both the contextual and the combination of the two.

2.3 Classification Algorithms

Also performing bias classification at the sentence level, Vincze (2013) detects sentences containing weasel, hedge or peacock words from the WikiWeasel corpus with a precision of 0.74, recall of 0.69 and F_1 of 0.71, by using a dictionary lookup approach. Bellows (2018) reports an accuracy of

0.68 on a corpus of 2,143 biased sentences from news articles, vectorized using tf-idf and classified with a Mutlinomial Naive Bayes, and an accuracy of 0.77 for a CNN and 0.78 with a RNN. Finally, Hube and Fetahu (2018) achieve an F_1 measure of 0.70 using Random Forest on 686 manually annotated sentences from Conservapedia.

3 Dataset Description

We propose a procedure to semi-automatically derive a labeled corpus of biased sentences from a Wikipedia dump in any language, which, for this paper, we applied to the April 2019 dumps[2] for Bulgarian, French and English.

3.1 Tagset Curation

First, we manually compile a list of NPOV-related tags for each of the target languages using the names of relevant Wikipedia maintenance templates[3] (`{{POV}}`, `{{NPOV}}`, `{{neutral point of view}}`, `{{peacock}}`, etc.).

Most tags, however, vary in spelling, not only based on the context (e.g., inline or at the beginning of an article), but also because of the open and collaborative nature of Wikipedia. Table 1 shows the sixteen most frequent "weasel" tag variations, only five of which (in bold) are documented on Wikipedia. While the official tag is the most frequently used, the unofficial variations account for almost 35% of the most frequent ways to tag a page containing weasel words.

While it may be effortless for human editors to interpret the meaning of these variations, it is not trivial to automatically identify all NPOV-related ones. Simply extracting all the tags starting with the official form of "weasel" yields unrelated tags such as "weasel, back-striped" (an animal) or "weasel, ben" (a punk singer). For that reason, we automatically compiled exhaustive tag frequency lists in each language, and then manually selected the relevant variations of each.

3.2 Revision Extraction

We look for occurrences of the selected tags across all revisions of each page, going forward from the oldest one. When a biased revision is found, we follow its evolution until the POV tag disappears, at which point we assume the problematic content has been either rewritten or edited out. Next, we extract the tag together with the pair of adjacent revisions, where the older one is tagged as biased and the newer one is not. We opted for this diachronic retrieval method, rather than relying on the repertoire of articles in Wikipedia's "NPOV dispute" section (Herzig et al., 2011; Recasens et al., 2013) since the latter only features currently tagged articles, while our method digs NPOV violations from revision histories.

Tag	Count	Ratio
weasel	201,092	0.5748
weasel-inline	89,352	0.2554
weasel words	21,755	0.0622
weasel word	16,991	0.0486
weasel section	3,954	0.0113
weasel-section	3,743	0.0107
weasel inline	2,631	0.0075
weaselinline	2,213	0.0063
weasel-words	2,176	0.0062
weasel-word	2,102	0.0060
weaselword	1,967	0.0056
weasel-name	956	0.0027
weaselwords	503	0.0014
weasel_section	225	0.0007
weasel_words	124	0.0004
weasel_word	80	0.0002

Table 1: "Weasel" tag variation in English

3.3 Processing and Filtering

Each of these revision pairs undergoes a cleaning process using regular expressions to strip as much of the Wikipedia markup, links, and page references as possible, while preserving visible text and essential punctuation. At this point, we proceed to tokenize the text and split it into sentences using the rule-based tokenizer and sentencizer methods of spaCy (Honnibal and Montani, 2017), whose 2.1.3 version supports 51 languages. Finally, we replace all numbers with a special token (`numtkn`), strip all remaining punctuation, and convert everything to lowercase.

Our algorithm also extracts revision pairs where the second member was the subject of a redirect or vandalism, which we filter out. We then compare the revisions to obtain the lists of deleted and inserted sentences for each pair. In about 20% of the cases, the difference consists in simply deleting the NPOV tag, which we believe is an artifact of editorial wars (Sumi et al., 2011; Yasseri et al.,

[2] https://dumps.wikimedia.org

[3] For English, see https://en.wikipedia.org/wiki/Category:Neutrality_templates

2012), given the contentiousness of most NPOV-flagged topics. Another 20% of the revision differences we set aside are punctuation or case-related.

We further clear the dataset from outliers (mostly acts of vandalism) by removing those with more than 400 edited sentences. Finally, we exclude revision pairs with minor differences (character-based Levenshtein distance of 1), which are spelling corrections rather than bias resolution. Table 2 gives the number of initial, final and excluded revisions per language.

Revision pairs	BG	FR	EN
initial number	1,021	46,331	197,953
tag removal	–257	–10,255	–61,397
punct./case	–194	–5,967	–44,345
redir./vandalism	–56	–1,524	–17,154
deletions only	–33	–2,740	–11,331
insertions only	–28	–2,819	–2,938
spelling	–3	–136	–400
outliers	–2	–153	–609
Total pairs	448	22,737	59,779

Table 2: Number of revision pairs per language

To build the final corpora, we take all removed and added sentences (under 300 tokens) from the pre-filtered revisions for the positive and negative classes respectively. We balance the dataset by using unchanged sentences (also treated as negatives), as shown in Table 3.

Sentences	BG	FR	EN
Removed	4,756	105,939	800,191
Added	3,288	72,183	494,993
Unchanged	1,468	33,756	305,198
Total	9,512	211,878	1,600,382

Table 3: Number of sentences per language

4 Dataset Evaluation

Once we have collected the tagged/untagged revision pairs for each language (as per §3.2), we evaluate their potential for automatic bias detection. Our intuition is that the sentences that were removed together with the NPOV tag in the same edit likely contain some form of bias. Insertions, on the other hand, come with little guarantee of neutrality, so we focus on the removed sentences.

4.1 Protocol

For each language, we distribute the tagged/ untagged revision pairs into four bins, based on the number of sentences that were removed in the edit (bin 1: 1 or 2 sentences removed, bin 2: 3–6, bin 3: 7–15, bin 4: 16 or more; these values were determined empirically to yield balanced bins in terms of revision pairs). Each annotator labeled 296 randomly picked sentences for a given language, distributed equally across the four bins. 72 of these sentences (24%) were shared by all annotators working on the same language, while the remaining 224 were labeled by a single annotator (cf. Table 4), thus allowing us to annotate more sentences while maintaining enough overlap to measure inter-annotator agreement (IAA). The Bulgarian sample was annotated by two native speakers, English by three with near-native proficiency, and French by two natives.

Lang	All	Ann1	Ann2	Ann3	Total
BG	72	224	224	—	520
FR	72	224	224	—	520
EN	72	224	224	224	744

Table 4: Number of sentences per annotator

The annotators were given identical instructions. For each sentence in their sample, they had to say whether it violated any of the NPOV principles stated in §1. The annotators were always presented with the full revision pair, so they had access to the context.

4.2 Dataset Evaluation Results

Since we had three annotators for English, we used Fleiss' κ to measure IAA. Tables 5 and 6 give the rate of positive annotations and IAA per language and per bin. On average, across all languages and bins, the annotators found 48% of positives in their samples, with an overall IAA of 0.41. Leaving out BG bin 4 (the only one with a negative κ), we get an average positive rate of 47% (std = 0.08) and an average κ value of 0.46 (std = 0.14). Our IAA coefficients are consistent with Vincze (2013), who had 200 English Wikipedia articles annotated by two linguists for weasel, peacock and hedge words, with IAA rates of 0.48, 0.45 and 0.46, respectively, and higher than the 0.35 reported by Hube and Fetahu (2018), who crowdsourced the annotation of sentences from Conservapedia into biased and unbiased. Identifying such phenomena is thus not trivial but reasonable agreement can be expected.

Bin	BG	EN	FR	avg	std
1	0.34	0.51	0.47	0.44	0.07
2	0.64	0.45	0.45	0.52	0.09
3	0.63	0.45	0.38	0.48	0.11
4	0.63	0.52	0.34	0.50	0.12
avg	0.56	0.48	0.41	0.48	0.06
std	0.13	0.03	0.05	0.03	0.10

Table 5: Positives in annotations

Bin	BG	EN	FR	avg	std
1	0.32	0.55	0.67	0.51	0.15
2	0.22	0.58	0.44	0.41	0.15
3	0.32	0.31	0.61	0.41	0.14
4	−0.23	0.39	0.68	0.28	0.38
avg	0.16	0.46	0.60	0.41	0.18
std	0.23	0.11	0.10	0.08	0.21

Table 6: Inter-annotator agreement (Fleiss' κ)

About half of the annotated sentences turn out to be neutral. Below, we discuss the sources of the noise we have observed in our dataset (including the added sentences).

4.3 Sources of Noise

We identified two types of noise: pipeline-related and human-related. Pipeline-related noise is either noise introduced at the pre-processing phase (e.g., due to inconsistent sentence segmentation) or noise that remains despite our filtering and cleaning efforts (e.g., NPOV-unrelated edits longer than one character, differences resulting from the introduction of an infobox, differences consisting in changing the spelling of numbers).

Human editor-related noise comes from the data itself and stems from the behaviour of Wikipedia's editors. It includes edits which introduce bias (often intentionally, as in (1) below), vandalism, corrections of factual mistakes unrelated to bias, replacing bias with another bias (cf. (2)), and collateral edits, i.e., neutral sentences neighbouring biased ones indirectly targeted by a large-scope edit (cf. (3)). Below are some examples.

(1) a. *(before)* cardinal health inc is a holding company

b. *(after)* cardinal health is a healthcare company **dedicated to making healthcare safer and more productive**

(2) a. *(before)* its support is low only in the cholla province which has for nearly numtkn years supported kim dae jung a well known **leftist** politician born in that province who also served as president of south korea numtkn numtkn

b. *(after)* its support is low only in the jeolla province which has for nearly numtkn years supported kim dae jung a well known **progressive** politician born in that province who also served as president of south korea numtkn numtkn

(3) a. *(before)* from the numtkn th century confucianism was losing its influence on vietnamese society monetary economy began to develop but unfortunately in negative ways

b. *(after)* from the numtkn th century confucianism was losing its influence on vietnamese society and a monetary economy began to develop

5 Expressions of Bias

The manual annotation also highlighted the variety of bias expression. Previously, Recasens et al. (2013) had identified two major classes: epistemological and framing bias (subjective intensifiers and one-sided terms), where they considered the first one to group more implicit expressions such as factive and assertive verbs, entailment and hedges. Based on their work and Wikipedia's Manual of Style, we present biased examples from our corpus[4] and discuss them in terms of the overt/ covert nature of the biased statement, its length (one or more words), and its level of ambiguity.

Subjective intensifiers are mostly expressed through single-word verbal and nominal modifiers (adverbs and adjectives) as in (4) and (5), but may also take the form of superlatives or quantifiers. They explicitly undermine tone neutrality by introducing overstatements and exaggerations (6).

(4) a. *(before)* some prominent liberals including scott reid were **strongly** critical of volpe s response

b. *(after)* some prominent liberals including scott reid **criticized** volpe s response

(5) *(before)* he is **truly** one of the greatest americans

(6) a. *(before)* this is an **absurd** statement because the cavalry of any age is designed first and foremost to run over the enemy and separate them as to make

[4] Examples are taken from the English evaluation subsets, where sentences are in lowercase, stripped of punctuation and numbers are replaced by `numtkn`.

them far more vulnerable to being overwhelmed and overrun

b. *(after)* this is **wrong** because the cavalry of any age is designed first and foremost to run over the enemy and separate them as to make them far more vulnerable to being overwhelmed and overrun

Clichés and jargon tend to be non-ambiguous but introduce low-frequency words in the corpus, as a result of being discouraged by Wikipedia.

(7) *(before)* x force was **concocted** by illustrator rob liefeld who started **penciling** the new mutants comic book in numtkn

Describing or analyzing rather than reporting events is a form of partiality harder to model, as it may not necessarily contain explicitly proscribed vocabulary.

(8) *(before)* he was a former club rugby and an opening batsman in club cricket but did not have the ability to make it all the way to the top level these two sports have become his particular area of expertise however he is very knowledgable on all sports that are played

(9) *(before)* however the most important consequence of the battle was that president lincoln was able to sieze upon the victory claim it as a strategic victory for the north and release his emancipation proclamation

Active voice may be used in cases like (10) to stress the agency of a participant in a situation, alongside a positively loaded support verb.

(10) a. *(before)* the united states department of justice indicted the company but **amway secured an acquittal**

b. *(after)* the united states department of justice indicted the company but **amway were acquitted**

To state a fact as an opinion is to use a weasel word to undermine the fact (11) or hide its source. While previous research shows the success of word-lists in detecting this particular type of bias (Recasens et al., 2013; Ganter and Strube, 2009), Vincze (2013) warns against the ambiguity of many of them. For example, *most* can be a weasel word (*Most agree that...*), a hedge (*most of his time*), a peacock (*the most touristic beach*) or neutral (*He did the most he could.*)

(11) a. *(before)* in the first invasion operation litani in numtkn the israeli military and south lebanon army sla occupied a narrow strip of land **ostensibly** as a security zone

b. *(after)* in the first operation litani in numtkn the israel defense forces and south lebanon army occupied a narrow strip of land **described** as the security zone

To state an opinion as a fact may be done with the use of an adverb (12) or an omission (13).

(12) a. *(before)* **in fact** the need for fast and secure fund transfers is growing and in the next year instant payments will quickly become the new normal for electronic fund transfers

b. *(after)* it is predicted that in the next year instant payments will become the standard for electronic fund transfers

(13) a. *(before)* in numtkn the journal won the praise of fascist leaders

b. *(after)* **there are some authors who retain** that the journal won the praise of fascist leaders

Intentional vagueness or the omission of factual information (14), is arguably the hardest type of bias expression to detect not only for machines, which are expected to recognize the lack of data as an informative feature, but also for humans, since filling factual gaps requires a fair amount of domain-specific knowledge.

(14) a. *(before)* as of numtkn it is the ethnic minority party in romania with representation in the romanian parliament

b. *(after)* as of numtkn it is the ethnic minority party in romania with representation in the romanian parliament **and is part of the governing coalition along with the justice and truth alliance and the conservatives**

6 Classification Experiments

The goal of the experiments is to assess the usefulness of the dataset in a sentence classification task. Our hypothesis is that having similar examples in both the biased and non-biased classes would help to single out discriminative words targeted by the NPOV-related edits.

Each dataset was split into a training set (80%), a development set (10%) on which we tuned the parameters, and a test set (10%) on which we ran a single evaluation with the best parameters.

6.1 Embeddings

We used fastText's classification function (Joulin et al., 2017), which implements a multinomial logistic regression algorithm on top of pretrained word embeddings. It uses word and character level embeddings to predict the class value of an instance. The parameter optimization was done by altering values for epoch (5, 10, 25), learning rate (0.1, 0.01, 0.05), word n-grams (1 to 5), minimum count (1–5), embedding dimensions (100, 300), loss function (softmax, ns, hs), minimum character level n-gram size (2, 3), using pretrained vectors or not, and learning rate update rate (50, 100).

When applying fastText's pretrained vectors,[5] we obtained comparable results for English and French without any significant gain, and with lower performance on Bulgarian. Thus, the final model chosen for its overall best performance across all three languages was trained without the use of an additional language model. The best performing values were then tried out on the test set.

6.2 Bag-of-Words Vectorization

We also experimented with classic bag-of-word vectorization with the stochastic gradient descent (SGD) (LeCun et al., 1998) and logistic regression (Hosmer and Lemeshow, 2000) algorithms. Each algorithm was run with the same settings on all three datasets to get the best average overall performances for precision, recall and F_1 measure. Parameter optimization was done using a grid search. Stop word lists were used for each language, which is the only language-specific aspect of the experiment.

The optimization for SGD ran 72 permutations with the following parameters:

- Bag-of-word n-gram size: unigrams only, unigrams and bigrams, unigrams to trigrams.
- Bag-of-word size: 100, 150, 300, 500, 1,000 and 3,000.
- Use idf reweighting or not.
- α value: 0.01, 0.001.

All the other parameters were set to their default values.[6] For logistic regression, 504 permutations were tested using the following settings:

- Same BOW n-gram size and BOW size and value type as SGD.
- C: 1.0e–3, 1.0e–2, 1.0e–1, 1.0e0, 1.0e+1, 1.0e+2 and 1.0e+3.
- Solver: sag, saga.

Using the training and development sets to run the grid search optimization on all three languages, the average F_1 measure was used to see which parameter values offered the best average performance across the board. The selected values were then used to run the same algorithm once on each language's training and test sets.

7 Results and Discussion

Table 7 shows the results for the experiments detailed in §6 for the SGD, fastText and logistic regression (LR) algorithms. For each performance measure, dataset section, algorithm and language, we provide results with respect to the *biased* class. The highest performance obtained on the test dataset of each language is in bold.

For the LR algorithm, the best performances were obtained using a C value of 0.001 with the saga solver using a unigram model of 100 features without inverse document frequency (idf) reweighting. The best parameters for the SGD used a model of unigrams to trigrams, with an α of 0.001 and idf reweighting. For fastText, the best performing parameter set used the default values[7] and a minimum of 5 occurrences per token.

Overall, the similar results between the development and test sets for each algorithm confirm that they did not overfit. Furthermore, all three measures have relatively low variance across languages, except for recall with SGD, which is considerably lower for Bulgarian (also impacting F_1) than for the other two languages.

We observe that FastText's vectorization and classification methods deliver higher precision upon larger datasets, but SGD and LR assure a much higher recall regardless of the number of examples.

While relatively better, the SGD performance level on the test set leaves room for improvement. This is likely due to the noise level in the sentences labeled as biased, which count many non-biased examples (see §4.2). The results are equally likely affected by the lexical and contextual ambiguity of the biased expressions, as discussed in §5. However, we do observe comparable best performance

[5] Available for 157 languages, pretrained on Common Crawl and Wikipedia (Grave et al., 2018) https://fasttext.cc/docs/en/crawl-vectors.html

[6] Version 0.21.2 of the sklearn toolkit.

[7] For version 0.8.3 of https://github.com/facebookresearch/fastText

Measure	Lang.	Dev-SGD	Test-SGD	Dev-fastText	Test-fastText	Dev-LR	Test-LR
Precision	BG	0.5387	**0.5886**	0.5324	0.5330	0.5182	0.5032
	FR	0.5059	0.5087	0.5533	**0.5520**	0.5151	0.5161
	EN	0.5112	0.5083	0.5656	**0.5634**	0.5230	0.5224
Recall	BG	0.4318	0.5049	0.4752	0.4937	0.6219	**0.6303**
	FR	0.8877	**0.8363**	0.5724	0.5721	0.6751	0.6739
	EN	0.8357	**0.8277**	0.5686	0.5718	0.5344	0.5354
F_1	BG	0.4794	0.5435	0.5022	0.5126	0.5653	**0.5596**
	FR	0.6444	**0.6146**	0.5627	0.5619	0.5844	0.5845
	EN	0.6334	**0.6291**	0.5671	0.5676	0.5286	0.5288

Table 7: Results for each language, dataset and classification method for the biased class

across corpora of varying size and languages from different families.

On the test set, our best overall average F_1 measure ranged between 0.56 and 0.62. This is lower than Vincze (2013)'s 0.71 or Hube and Fetahu (2018)'s 0.70, but our approach uses a large corpus, automatically derived from Wikipedia in any language with minimal language-specific input, applied to sentence-level bias detection, while Vincze (2013) used a monolingual, dictionary-based approach, and Hube and Fetahu (2018) relied on language-specific resources to extract multiple lexical and grammatical features. Our results set the baseline for sentence-level bias detection across the three languages of this corpus. Higher performance for a specific language may be achieved by a reconfiguration of the parameters or by the introduction of additional features.

8 Conclusion and Future Work

We presented a semi-automatic method to extract biased sentences from Wikipedia in Bulgarian, French and English. As this method does not rely on language-specific features, apart from the NPOV tag list and a stop word list, it can be easily applied to Wikipedia archives in other languages. It relies on the tags added by human editors in the articles that they considered biased. We retrieve the last tagged revision and the untagged revision following it and regard them respectively as biased and unbiased. By comparing the revisions, we get the lists of removed and added sentences.

We manually annotated 1,784 of the removed sentences, for all three languages combined, and found that only about half of them were actually biased. An average Fleiss' κ of 0.41 (0.46 if ignoring an outlier), consistent with the literature, indicates that the task is not trivial even for humans.

Using our corpora, we tested three classification algorithms: bag-of-word vectorization with SGD, fastText, and logistic regression.

In future work, we would like to improve the quality of the dataset by addressing issues uncovered during the human evaluation, such as incoherent sentence segmentation, enumerations, minor edits and remaining noise. Another conceivable optimization is to segment the dataset into two or more subsets according to the main forms of bias expression (e.g., explicit vs implicit). It would allow to explore and evaluate different forms of bias separately, which in turn might motivate differential classification techniques. Finally, populating the negative examples class with sentences from Wikipedia's Featured Articles (in line with Bhosale et al. 2013) might help reduce class ambiguity by reinforcing the contrast between neutral encyclopedic tone and expressions of bias.

Acknowledgments

This work has been supported by the Ministère de l'Économie et de l'Innovation du Québec (MEI). We would like to thank the annotators for their help with the quality evaluation process and the anonymous reviewers for their insightful comments.

References

Khalid Al Khatib, Hinrich Schütze, and Cathleen Kantner. 2012. Automatic detection of point of view differences in Wikipedia. In *Proceedings of COLING 2012*. The COLING 2012 Organizing Committee, Mumbai, India, pages 33–50.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA), Valletta, Malta.

Ricardo Baeza-Yates. 2018. Bias on the web. *Communications of the ACM* 61(6):54–61.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. *EMNLP-2018* .

Martha Bellows. 2018. *Exploration of Classifying Sentence Bias in News Articles with Machine Learning Models*. Ph.D. thesis, University of Rhode Island.

Shruti Bhosale, Heath Vinicombe, and Raymond Mooney. 2013. Detecting promotional content in wikipedia. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1851–1857.

Robert M Entman. 2007. Framing bias: Media in the distribution of power. *J. Commun.* 57(1):163–173.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The conll-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*. Association for Computational Linguistics, pages 1–12.

Viola Ganter and Michael Strube. 2009. Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. pages 173–176.

Njagi Dennis Gitari, Zuping Zhang, Hanyurwimfura Damien, and Jun Long. 2015. A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering* 10(4):215–230.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Livnat Herzig, Alex Nunes, and Batia Snir. 2011. An annotation scheme for automated bias detection in wikipedia. In *Proceedings of the 5th Linguistic Annotation Workshop*. Association for Computational Linguistics, Stroudsburg, PA, USA, LAW V '11, pages 47–55.

Nicholas P Hirning, Andy Chen, and Shreya Shankar. 2017. Detecting and identifying bias-heavy sentences in news articles. Technical report, Stanford University.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* .

David W. Hosmer and Stanley Lemeshow. 2000. *Applied logistic regression*. John Wiley and Sons.

Christoph Hube and Besnik Fetahu. 2018. Detecting biased statements in wikipedia. In *Companion Proceedings of the The Web Conference 2018*. International World Wide Web Conferences Steering Committee, pages 1779–1786.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1113–1122.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 427–431.

Sicong Kuang and Brian D Davison. 2016. Semantic and context-aware linguistic model for bias detection. In *Proc. of the Natural Language Processing meets Journalism IJCAI-16 Workshop*. pages 57–62.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. Springer-Verlag, London, UK, UK, pages 9–50.

Chenghua Lin, Yulan He, and Richard Everson. 2011. Sentence subjectivity detection with weakly-supervised learning. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 1153–1161.

Gabriel Murray and Giuseppe Carenini. 2009. Detecting subjectivity in multiparty speech. In *Tenth Annual Conference of the International Speech Communication Association*.

Cathy O'Neil. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown.

Umashanthi Pavalanathan, Xiaochuang Han, and Jacob Eisenstein. 2018. Mind your POV: Convergence of articles and editors towards wikipedia's neutrality norm. *Proc. ACM Hum. -Comput. Interact.* 2(CSCW):137:1–137:23.

Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On measuring bias in online information. *ACM SIGMOD Record* 46(4):16–21.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1650–1659.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*.

Lee D Ross, Teresa M Amabile, and Julia L Steinmetz. 1977. Social roles, social control, and biases in social-perception processes. *J. Pers. Soc. Psychol.* 35(7):485.

Nicole Shadowen. 2019. Ethics and bias in machine learning: A technical study of what makes us "good". In Newton Lee, editor, *The Transhumanism Handbook*, Springer International Publishing, Cham, pages 247–261.

R Sumi, T Yasseri, A Rung, A Kornai, and J Kertesz. 2011. Edit wars in wikipedia. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. ieeexplore.ieee.org, pages 724–727.

A Tversky and D Kahneman. 1974. Judgment under uncertainty: Heuristics and biases. *Science* 185(4157):1124–1131.

Veronika Vincze. 2013. Weasels, hedges and peacocks: Discourse-level uncertainty in wikipedia articles. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. pages 383–391.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *International conference on intelligent text processing and computational linguistics*. Springer, pages 486–497.

Theresa Wilson and Stephan Raaijmakers. 2008. Comparing word, character, and phoneme n-grams for subjective utterance recognition. In *Ninth Annual Conference of the International Speech Communication Association*.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, Stroudsburg, PA, USA, CSLDAMT '10, pages 152–158.

Taha Yasseri, Robert Sumi, András Rung, András Kornai, and János Kertész. 2012. Dynamics of conflicts in wikipedia. *PLoS One* 7(6):e38869.

Supervised Morphological Segmentation Using Rich Annotated Lexicon

Ebrahim Ansari,[†‡] **Zdeněk Žabokrtský,**[†]
Mohammad Mahmoudi,[‡] **Hamid Haghdoost**[‡] **and Jonáš Vidra**[†]
[†] Institute of Formal and Applied Linguistics,
Faculty of Mathematics and Physics, Charles University
[‡] Department of Computer Science and Information Technology,
Institute for Advanced Studies in Basic Sciences
{ansari,m.mahmodi,hamid.h}@iasbs.ac.ir
{zabokrtsky,vidra}@ufal.mff.cuni.cz

Abstract

Morphological segmentation of words is the process of dividing a word into smaller units called morphemes; it is tricky especially when a morphologically rich or polysynthetic language is under question. In this work, we designed and evaluated several Recurrent Neural Network (RNN) based models as well as various other machine learning based approaches for the morphological segmentation task. We trained our models using annotated segmentation lexicons. To evaluate the effect of the training data size on our models, we decided to create a large hand-annotated morphologically segmented corpus of Persian words, which is, to the best of our knowledge, the first and the only segmentation lexicon for the Persian language. In the experimental phase, using the hand-annotated Persian lexicon and two smaller similar lexicons for Czech and Finnish languages, we evaluated the effect of the training data size, different hyperparameters settings as well as different RNN-based models.

1 Introduction

Morphological analysis must be tackled somehow in all natural language processing tasks, such as machine translation, speech recognition, and information retrieval. Morphological segmentation of words is the process of dividing a word into smaller units called morphemes. Morphological segmentation task is harder for those languages which are morphologically rich and complex like Persian, Arabic, Czech, Finnish or Turkish, especially when there are not enough annotated data for those languages. In this paper, we designed and evaluated various supervised setups to perform morphological segmentation using a hand-annotated segmented lexicon for training.

The efficiency of supervised approaches (especially of deep neural network models) is naturally highly dependent on the size of training data. In order to evaluate the effect of the training data size on our segmentation models, we created a rich Persian hand-annotated segmentation lexicon, which is, as far as we know, the first and the only such computer-readable dataset for Persian. Persian (Farsi) is one of the languages of the Indo-European language family within the Indo-Iranian branch and is spoken in Iran, Afghanistan, Tajikistan and some other regions related to ancient Persian. In addition, we evaluated our models on Czech and Finnish, however, the amount of annotated data for them is substantially lower.

Automatic morphological segmentation was firstly introduced by Harris (1970). More recent research on morphological segmentation has been usually focused on unsupervised learning (Goldsmith, 2001; Creutz and Lagus, 2002; Poon et al., 2009; Narasimhan et al., 2015; Cao and Rei, 2016), whose goal is to find the segmentation boundaries using an unlabeled set of word forms (or possibly a corpus too). Probably the most popular unsupervised systems are LINGUISTICA (Goldsmith, 2001) and MORFESSOR, with a number of variants (Creutz and Lagus, 2002; Creutz et al., 2007; Grönroos et al., 2014). Another version of the latter which includes a semi-supervised extension was introduced by (Kohonen et al., 2010). Poon et al. (2009) presented a log-linear model which uses overlapping features for unsupervised morphological segmentation.

In spite of the dominance of the unsupervised systems, as soon as even just a small amount of

Proceedings of Recent Advances in Natural Language Processing, pages 52–61,
Varna, Bulgaria, Sep 2–4 2019.

segmented training data is available, then the entirely unsupervised systems tend not to be competitive. Furthermore, unsupervised segmentation still has considerable weaknesses, including over-segmentation of roots and erroneous segmentation of affixes (Wang et al., 2016). To deal with those limitations, recent works show a growing interest in semi-supervised and supervised approaches (Kohonen et al., 2010; Ruokolainen et al., 2013, 2014; Sirts and Goldwater, 2013; Wang et al., 2016; Kann and Schütze, 2016; Kann et al., 2018; Cotterell and Schütze, 2017; Grönroos et al., 2019) which employ annotated morpheme boundaries in the training phase.

In our work we designed and evaluated various machine learning models and trained them using only the annotated lexicon in a supervised manner. Our models do not leverage the unannotated data nor context information and only use the primary hand-annotated segmentation lexicons.

Experimental results show that our Bi-LSTM model perform slightly better than other models in boundary prediction for our hand-segmented Persian lexicon, while KNN (K-Nearest Neighbors algorithm) performs better when the whole word accuracy is under question.

The paper is organized as follows: Section 2 addresses the related work on morphological segmentation. Section 3 describes the methodology and machine learning models used in this work. Section 4 introduces our hand-segmented Persian lexicon as well as related preprocessing phases. Section 5 presents the experiment results compared to some other baseline systems and finally Section 6 concludes the paper.

2 Related Work

Supervised morphological segmentation, i.e. using a lexicon (or a corpus) with annotated morpheme boundaries in the training phase, has attracted increasing attention in recent years. One of the most recent successful research directions on supervised morphological segmentation is the work of (Ruokolainen et al., 2013), whose authors employ CRF (Conditional Random Fields), a popular discriminative log-linear model to predict morpheme boundaries given their local sub-string contexts instead of learning a morpheme lexicon. (Ruokolainen et al., 2014) extended their work to semi-supervised learning version by exploiting some available unsupervised segmentation techniques into their CRF-based model via a feature set augmentation. (Ruokolainen et al., 2014)

Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) have recently achieved great success in sequence learning tasks, including outstanding results on sequential tasks such as machine translation (Sutskever et al., 2014). Wang et al. (2016) proposed three types of window-based LSTM neural network models named Window LSTM, Multi-window LSTMs and Bidirectional Multi-Window LSTMs, in order to automatically learn sequence structures and predict morphological segmentations of words in a raw text. They used only word boundary information without any need for extra feature engineering in the training phase. The authors compared their models with selected supervised models as well as with an LSTM architecture (Wang et al., 2016), and similarly to the work of Ruokolainen et al. (2013), their architecture is based on the whole text and context information instead of using only the lexicon. Cotterell and Schütze (2017) increased the segmentation accuracy by employing semantic coherence information in their models. They used RNN (Recurrent Neural Network) to design a composition model. They also found that using RNN with dependency vector has the best results on vector approximation (Cotterell and Schütze, 2017).

Recently, using encoder-decoder models Bahdanau et al. (2014) (attention-based models) made some great successes in machine translation systems. Kann and Schütze (2016) used an encoder-decoder model which encodes the input as a sequence of morphological tags of source and targets and feeds the model by sequence of letters of a source form. They select the final answer using a majority voting amongst their five different ensembled RNN encoder-decoder models. Kann and Schütze (2016), proposed a seq2seq (sequence-to-sequence network) architecture for the word segmentation task. They used a bi-directional RNN to encode the input word (i.e. sequence of characters) and concatenated forward and backward hidden states yielded from two GRUs and pass the result vector to decoder part. The decoder is a single GRU which uses segmentation symbols for training. She introduced two multi-task training approaches as well as data augmentations to improve the quality of the presented model. She shows that neural seq2seq models perform on par with or better than other strong baselines for polysynthetic languages in a minimal-resource setting. Their suggested neural seq2seq models constitute the state of the art for morphological segmentation in high-resource settings and for (mostly) European languages (Kann et al., 2018).

The main studied language in our work is Persian, which belongs to morphologically rich languages and which is powerful and versatile in word building. Having many affixes to form new words (over a hundred), and the ability to build affixes and especially prefixes from nouns, the Persian language is considered as an agglutinative language since it also frequently uses derivational agglutination to form new words from nouns, adjectives, and verbal stems. Hesabi (1988) claimed that Persian can derive more than 226 million words (Hesabi, 1988).

To the best of our knowledge, the research on morphology of the Persian language is very limited. Rasooli et al. (2013) claimed that performing morphological segmentation in the pre-processing phase of statistical machine translation could improve the quality of translations for morphology rich and complex languages. Although they segmented very low portion of Persian words (only some Persian verbs), the quality of their machine translation system increases by 1.9 points of BLEU score. Arabsorkhi and Shamsfard (2006) proposed a Minimum Description Length (MDL) based algorithm with some improvements for discovering the morphemes of Persian language through automatic analysis of corpora.

3 Our Machine Learning Models

In this work we decided to evaluate selected machine learning models including those feature-based machine learning approaches in which the task of word segmentation is reformulated as a classification task, as well as various deep-learning (DL for short) neural network models.

Because of huge number of learned parameters in DL, having enough training data is critical. The fact that we decided to create a large hand-annotated dataset for Persian allows evaluating the effect of the training data size on a relatively wide scale, as described in Section 4.

We convert all segmentations into a simple string format in which letters "B" and "L" encode the presence of the boundary letter and the continuation letter, respectively. For example for word "goes", the encoded segmentation is "LLBL", which shows that there is a segmentation boundary in front of the third letter ("e"). While in our model we consider only morphologically segmented lexicon and we do not employ any other information like corpus contexts or lists of unannotated words, this encoding is sufficient and make the specification of boundary location easy.

In the case of presence of a semi-space letter (a feature specific for the Persian written language), the semi-space letter is always considered as a boundary letter. An experiments focused on this feature is described in Subsection 5.2.3, which shows that our models could perform better when this information exists in the annotated lexicon.

3.1 Classification-Based Segmentation Models

In the first setup, we convert the segmentation task (the task of segmenting a word into a sequence morphemes) simply to a set of independent binary decisions capturing the presence or absence of a segmentation boundary in front of each letter in the word. For this task, we use various standard off-the-shelf classifiers available in the Scikit-learn toolkit (Pedregosa et al., 2011).

So far, we provide the classifiers only with features that are extractable from the word alone. More specifically, we use only character-based features. These character-based features include letters and letter sequences (and their combinations) before and after under the character under question, which is subsequently assigned one out of two classes: "B" for boundary characters, and "L" which stands for continuation characters. The main task of these methods is then to train a classification model to classify all characters in the word into those two classes, given binary features based on surrounding characters. For example, for the fifth character of word "hopeless", some of our features could be: "e", "le", and "ope". The classification predictions are performed independently.

3.2 Deep Neural Network Based Models

Besides the classification-based segmentation models, we designed and evaluated five DL models based on GRU, LSTM, Bi-LSTM, seq2seq and Bi-LSTM with the attention mechanism, respectively. The first three models are illustrated in Figures 1 and 2. The presented seq2seq model, is similar to the model described in (Grönroos et al., 2019). The last presented model is an attention based model, which is shown in Figure 3. In this model, we use Bi-LSTM as encoder and LSTM as attention layer, and finally, outputs of encoder and attention layers are added together.

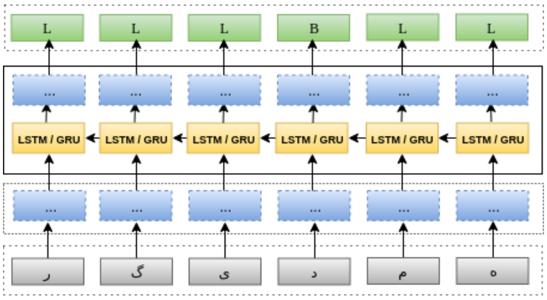


Figure 1: The schema of the LSTM/GRU models used in this experiments.

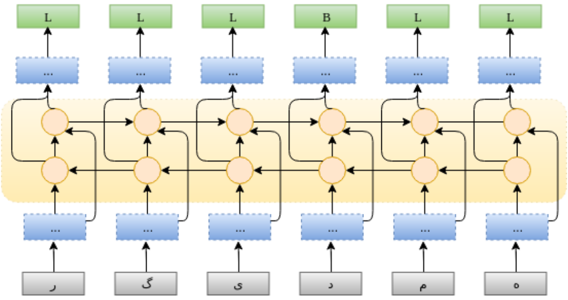


Figure 2: The schema of the Bi-LSTM model used in this experiments.

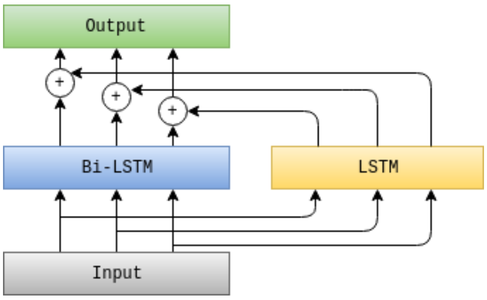


Figure 3: The schema of the Bi-LSTM with the attention mechanism model used in this experiments.

4 Morphological Segmentation Lexicons

In this section, the rich Persian hand-annotated dataset and the existing Finnish datasets from the Morpho-Challenge shared task 2010 (Virpioja et al., 2011) as well as the Czech dataset used in our experiments are described.

4.1 Persian Hand-Annotated Morphological Segmentation Dataset

We extracted our primary word list from three different corpora. The first corpus contains sentences extracted from the Persian Wikipedia (Karimi et al., 2018). The second one is popular Persian mono-lingual corpus BijanKhan (Bijankhan et al., 2011), and the last one is Persian-NER[1] (Poostchi et al., 2018).

For all introduced corpora, using Hazm toolset (Persian preprocessing and tokenization tools)[2] and the stemming tool presented by Taghi-Zadeh et al. (2015), we extracted and normalized all sentences and in the final steps using our rule-based stemmer and a Persian lemma collection, all words are lemmatized and stemmed. Finally all semi-spaces are automatically detected and fixed. Words with more than 10 occurrences in the corpora were selected for manual annotation. We decided to send all 80K words to our 16 annotators in the way that each word is checked and annotated by two independent persons. Annotators decided about the lemma of a word under question, segmentation parts, plurality, ambiguity (whether a word has more than one meaning) or they might delete the word if they think is not a proper Persian word. Moreover, some segmentations predicted by our automatic segmentator with high confidence score were offered to our annotators. We removed almost 30K words which were selected to be deleted by both annotators. And remaining 50K words sent for inter-annotation comparison part. In this step, all disagreements were checked and corrected by the authors of this paper and finally all words were quickly reviewed by two Persian linguists. The whole process took around six weeks. In order to use a hand-annotated lexicon in our work, we extracted the segmentation part from the dataset and converted it to our binary model which is described in Section 3.

The total number of words we used in our Persian dataset is 40K. The dataset is publicly available in the LINDAT/CLARIN repository (Ansari et al., 2019).

[1] https://github.com/HaniehP/PersianNER

[2] https://github.com/sobhe/hazm

4.2 Existing Finnish and Czech Segmentation Datasets

We downloaded the Finnish segmentation dataset from the Morpho-Challenge shared task 2010[3] (Virpioja et al., 2011) and converted them into our binary format. The Finnish dataset contains 2000 segmented words. While comparing to our hand-annotated Persian dataset these datasets are small, we used them to see the efficiency of our presented models when the size of training dataset is limited.

The Czech dataset results from a prototype segmentation annotation of Czech words. A sample of 1000 lemmas were selected randomly from DeriNet, which is a lexical database focus on derivation in Czech (Žabokrtský et al., 2016). The lemmas were manually segmented by two independent annotators, and all annotation differences were resolved subsequently during a third pass through the data. The annotation resulted in 4.6 morphemes per word, partially as a result of the fact that the lemmas were sampled uniformly, regardless of their corpus frequency, and thus the selection is biased towards longer words.

5 Experimental Results

To partition our dataset (Persian, Czech and Finnish) into training, development and test sets a commonly used method is used (Ruokolainen et al., 2013), which involves sorting words according to their frequency and assigning every eighth term starting from the first one into the test set and every eighth term from the second into the development set, while moving the remaining terms into the training set.

In order to evaluate the effect of the training data size, we randomly select the first 1/64, 1/32, 1/16, 1/8, 3/8, 1/4, 1/2, 3/4 and all amount of data from the training set to carry out experiments with different training sizes. In all experiments, we report three evaluation measures: the number of correctly predicted morpheme boundaries (in terms of precision, recall, and f-measure), the percentage of correct binary predictions on all characters, and the percentage of correctly segmented words.

As described in Section 2, some previous works reported accuracy in terms of the number of correct predictions (boundary and word) in a running text, instead of considering unique words sampled from a lexicon. Hence we decided to also report such accuracy in our experiments in addition to our lexicon evaluation. For this new experiment, we selected a part of a mono-lingual text and after removing all presented words in the text from our training lexicon, the remaining segmented words are considered as the training set and finally accuracy of word segmentation of words in test sentences is reported separately.

5.1 Baselines

We used two baseline systems which we selected to compare our models with. The first baseline is an unsupervised version of MORFESSOR, which is introduced and implemented by Creutz et al. (2007). The second baseline is FlatCat (Grönroos et al., 2014), which is a well-known semi-supervised version of MORFESSOR that uses the Hidden Markov Model for segmentation. In addition to the annotated data, semi-supervised MORFESSOR (i.e. FlatCat) uses a set of 100,000 word types following their frequency in the corpus as their unannotated training dataset. For both baselines, the best performing model is selected and compared with our neural network based models.

5.2 Results and Discussion

As described in Section 4, we designed various models for the morphological segmentation task. In the following subsections, different experiments done in this work are reviewed. In all tables, the column entitled by **W%** indicates the proportion of perfectly segmented words. The column entitled by **Ch%** indicated the accuracy of characters which are classified as boundary or non-boundary. Finally, **P%**, **R%**, and **F%** indicate precision, recall and F-measure score respectively for the morpheme boundary detection, naturally excluding the trivial final position characters from our evaluation.

5.2.1 Comparison of Different Models

Table 1 shows the evaluation results of morphological segmentation using our Persian hand-annotated dataset if the whole training data is used. For each model, only results of the best-performing hyperparameter configuration are reported. As is shown in Table 1, our Bi-LSTM model performs slightly better than the rest in boundary prediction, however, the classification models are surprisingly almost on the par with our complex DL model. Considering word accuracy,

[3] http://morpho.aalto.fi/events/morphochallenge2010/datasets.shtml

Model	P% / R% / F%	W%	Ch%
LSTM	90.09 / 87.55 / 88.80	64.10	93.20
GRU	85.43 / 84.50 / 84.96	58.35	91.44
Bi-LSTM	92.50 / 88.65 / **90.53**	66.51	94.37
Seq2Seq	88.04 / 84.04 / 86.09	59.10	91.65
Bi-LSTM with Attention	92.57 / 85.85 / 89.08	65.30	93.52
SVC, Kernel: linear	85.86 / 82.20 / 83.94	73.08	94.45
SVC, Kernel: poly, Degree: 2	89.57 / 85.86 / 87.67	**78.72**	95.72
SVC, Kernel: rbf	89.71 / 84.42 / 86.99	77.61	95.52
SVC, Kernel: poly, Degree: 5	89.77 / 83.91 / 86.74	77.17	95.45
SVC, Kernel: poly, Degree: 3	89.58 / 85.89 / 87.70	78.70	95.73
Logistic Regression, Solver: sag	87.55 / 79.66 / 83.42	72.60	94.39
Logistic Regression, Solver: liblinear	87.55 / 79.60 / 83.42	72.63	94.39
Logistic Regression, Solver: lbfgs	87.49 / 79.78 / 83.46	72.64	94.39
KNeighbors, Neighbors: 5	86.22 / 82.47 / 84.30	73.12	94.56
KNeighbors, Neighbors: 10	86.22 / 82.47 / 84.03	73.12	94.56
KNeighbors, Neighbors: 30	90.23 / 86.69 / 88.42	78.64	**95.73**
Ada Boost, Estimators: 100	83.34 / 64.10 / 72.46	58.21	90.83
Decision Tree	88.25 / 87.05 / 87.65	76.83	95.38
Random Forest, Estimators: 10	89.75 / 84.87 / 87.15	76.08	95.30
Random Forest, Estimators: 100	89.93 / 85.92 / 87.88	77.37	95.54
Bernoulli Naive Bayes	78.38 / 88.31 / 83.05	66.71	93.21
Perceptron MaxIteration: 50	83.98 / 74.45 / 78.93	65.07	92.52
Unsupervised MORFESSOR	69.58 / 81.10 / 74.90	29.01	83.28
Supervised MORFESSOR	82.13 / 92.94 / 87.20	59.56	91.60

Table 1: Result of applying our models on small Persian segmented lexicon. **P%**, **R%**, and **F%** indicate precision, recall and F-measure score respectively. **W%** means the percentage of number of correct predicted words and **Ch%** indicated the the accuracy of characters which are classified in two boundary or non-boundary classes.

classification models are performing better than DL models. A possible explanation for this is that the classification models make use of n-gram features and handle the characteristics of the whole word more efficiently than sequence-based models. Moreover, regarding our experiments, the presented seq2seq model does not perform well. An explanation could be that while there is not any available context information, the used attention mechanism does not have any far parts to make a relation between them. Moreover, our Bi-LSTM with the attention mechanism does not perform better than normal Bi-LSTM either. Finally, Tables 2 and 3 show the results of this experiment on two other languages, Finnish and Czech, for which the sizes of training data are very limited comparing the Persian dataset. As we expected, with so small training data, the classification methods perform better than more complex DL strategies.

Table 4 shows a comparison of our DL models, when different LSTM output sizes and drop-out thresholds are tested. Only two best-performing models (LSTM and Bi-LSTM) are shown.

Model	P% / R% / F%	W%	Ch%
LSTM	99.67 / 29.08 / 44.98	03.58	81.57
GRU	99.99 / 28.01 / 45.01	03.59	81.60
Bi-LSTM	86.96 / 32.82 / 47.66	04.88	81.30
Bi-LSTM with Attention	81.50 / 44.18 / 57.30	05.53	78.26
SVC, Kernel: linear	78.39 / 76.83 / 77.31	38.11	91.16
SVC, Kernel: poly, Degree: 2	89.00 / 77.62 / 82.23	47.55	93.63
SVC, Kernel: rbf	90.06 / 74.83 / 81.74	45.92	93.34
SVC, Kernel: poly, Degree: 5	91.35 / 64.71 / 75.75	35.83	91.75
SVC, Kernel: poly, Degree: 3	89.70 / 76.56 /**82.61**	**46.57**	**93.58**
Logistic Regression, Solver: sag	82.43 / 69.37 / 75.34	31.92	90.95
Logistic Regression, Solver: liblinear	82.43 / 69.37 / 75.34	31.92	90.95
Logistic Regression, Solver: lbfgs	82.43 / 69.37 / 75.34	31.92	90.95
KNeighbors, Neighbors: 5	82.56 / 71.23 / 76.48	33.55	91.27
KNeighbors, Neighbors: 10	82.56 / 71.23 / 76.48	33.55	91.27
KNeighbors, Neighbors: 30	82.56 / 71.23 / 76.48	33.55	91.27
Ada Boost, Estimators: 100	76.45 / 38.48 / 51.19	16.28	85.38
Decision Tree	79.58 / 76.29 / 77.90	39.41	91.38
Random Forest, Estimators: 10	87.41 / 68.44 / 76.77	37.45	91.75
Random Forest, Estimators: 100	88.08 / 72.83 / 79.73	44.29	92.62
Bernoulli Naive Bayes	64.27 / 76.43 / 69.82	26.38	86.84
Perceptron MaxIteration: 50	73.22 / 75.36 / 74.27	31.92	89.60
Unsupervised MORFESSOR	25.85 / 89.87 / 40.15	00.32	30.53
Supervised MORFESSOR	70.48 / 79.67 / 74.79	31.49	87.68

Table 2: Result of applying our models on small Finnish segmented lexicon.

Model	P% / R% / F%	W%	Ch%
LSTM	69.64 / 36.44 / 47.82	04.19	69.77
GRU	74.72 / 27.23 / 39.92	00.59	63.86
Bi-LSTM	68.56 / 48.33 / 56.69	05.38	67.45
Bi-LSTM with Attention	66.62 / 71.16 / 68.81	08.98	72.16
SVC, Kernel: linear	84.28 / 70.84 / 76.98	20.95	83.88
SVC, Kernel: poly, Degree: 2	91.42 / 69.46 / 78.94	31.73	85.90
SVC, Kernel: rbf	91.39 / 67.40 / 77.59	30.53	85.19
SVC, Kernel: poly, Degree: 5	94.03 / 48.71 / 64.18	20.35	79.32
SVC, Kernel: poly, Degree: 3	90.95 / 60.37 / 72.57	25.14	82.64
Logistic Regression, Solver: sag	90.69 / 66.89 / 76.99	25.04	84.80
Logistic Regression, Solver: liblinear	90.69 / 66.89 / 76.99	25.04	84.80
Logistic Regression, Solver: lbfgs	90.69 / 66.89 / 76.99	25.04	84.80
KNeighbors, Neighbors: 5	82.18 / 79.93 / 81.04	28.74	85.77
KNeighbors, Neighbors: 10	87.50 / 76.15 / **81.24**	**29.34**	**86.62**
KNeighbors, Neighbors: 30	82.18 / 79.93 / 81.04	28.74	85.77
Ada Boost, Estimators: 100	88.85 / 57.46 / 69.79	16.16	81.08
Decision Tree	78.46 / 56.26 / 65.53	15.56	77.49
Random Forest, Estimators: 10	91.42 / 65.86 / 76.57	**29.34**	84.67
Random Forest, Estimators: 100	91.76 / 68.78 / 76.82	**29.34**	85.77
Bernoulli Naive Bayes	85.94 / 74.44 / 79.77	26.94	85.64
Perceptron MaxIteration: 50	80.45 / 72.04 / 76.01	19.16	82.71
Unsupervised MORFESSOR	44.28 / 99.33 / 61.25	00.59	44.61
Supervised MORFESSOR	67.12 / 77.43 / 71.91	05.95	73.33

Table 3: Result of applying our models on the Czech segmented lexicon.

Model	Parameters	P% / R% / F%	W%	Ch%
Bi-LSTM	Outstate: 25 Dropout: 0.2	89.44 / 82.80 / 86.00	59.44	91.73
Bi-LSTM	Outstate: 50 Dropout: 0.2	88.79 / 87.89 / 88.34	62.57	92.86
Bi-LSTM	Outstate: 70 Dropout: 0.2	91.39 / 88.85 / 90.10	64.51	93.70
Bi-LSTM	Outstate: 70 Dropout: 0.5	92.50 / 88.65 / **90.53**	**66.51**	**94.37**
LSTM	Outstate: 25 Dropout: 0.2	91.69 / 83.00 / 87.13	62.32	92.45
LSTM	Outstate: 50 Dropout: 0.2	93.09 / 82.29 / 87.36	60.82	92.67
LSTM	Outstate: 70 Dropout: 0.2	90.09 / 87.55 / 88.80	64.10	93.20
LSTM	Outstate: 70 Dropout: 0.5	87.86 / 88.59 / 88.22	62.19	92.72

Table 4: Effect of using different hyper-parameters on LSTM and Bi-LSTM models, two best performing deep neural network models for Persian dataset

As is seen in the tables, the classification models perform well when compared to more complex DL models. One explanation for this evidence is the lack of any external information (other than a segmented lexicon) which limits the number of possible features from the training data. For example there is no information about some previous words, and consequently RNN-based models can not learn any information about distant previous characters in the training phase. Possibly, this also explains the inferior performance of our seq2seq model compared to the Bi-LSTM model implemented for this work.

Finally, Table 5 shows results of selected models when the segmentation is done on all words occurring in a corpus instead of a segmented lexicon. In this experiments we expected those words with more frequency has higher effect on results comparing with less frequent words.

5.2.2 Effect of Training Data Size

In order to evaluate the effect of the training data size on our DL models, different amount of training data are selected from and feed to our models. Figure 4 and Figure 5 demonstrate an experiment in which the baseline line is the results of unsupervised version of MORFESSOR for similar test dataset. Only four best performing feature-based models in addition to two DL-based models are selected to be shown here. As this figure shows, after having more than 10K training instances, increasing the training data further does not have a substantial effect any more.

5.2.3 Semi-Space Feature for Persian Words

An important feature of the Persian and Arabic languages is the existence of semi-space. For example word "کتاب‌ها" (books) is a combination of word "کتاب" and "ها", in which the former is Persian translation of word "book" and the latter is morpheme for a plural form. We can say these semi-space signs segment words into smaller morphemes. However, in formal writing and in all Persian normal corpora, this space is neglected frequently and it could make a lot of problems in Persian and Arabic morphological segmentation task. For example both forms for the previous example, "کتاب‌ها" and "کتابها" , are considered correct in Persian text and have the same meaning. To deal with this problem and in order to improve the quality of our segmentation dataset, we implemented a preprocessor to distinguish this kind of space in Persian words and consequently our hand-annotated dataset contains these semi-spaces correctly. While we wanted to test the effect of having this prior knowledge in the lexicon, we evaluated our models in two different forms. In the first case, we used our hand annotated dataset as is. In the second case, we removed all semi-spaces from the lexicon. Table 6 shows a comparison for deploying our models on these two different datasets and as could be seen in this table, having the accurate dataset which is created by our preprocessing strategy could improve results drastically.

Model	**P% / R% / F%**	**W%**	**Ch%**
LSTM	94.42 / 92.93 / 93.67	78.14	95.13
Bi-LSTM	95.97 / 93.69 / **94.89**	**78.37**	**95.79**
SVC, Kernel: poly, Degree: 3	93.88 / 92.11 / 92.99	89.85	97.02
KNeighbors, Neighbors: 30	94.50 / 92.77 / 93.63	89.91	96.93
Random Forest, Estimators: 100	94.32 / 91.99 / 93.10	88.64	96.66

Table 5: Experiment results when a model is used to predict boundaries of Persian words of a small corpus instead of lexicon words. Only five best performing models are shown.

6 Conclusion

The main task of this work is to evaluate different supervised models to find the best segmentation of a word when only a segmented lexicon without any extra information is available in the training phase. In recent years, recurrent neural networks (RNN) attracted a growing interest in morphological analysis, that is why we decided to design and evaluate various neural network based models (LSTM, Bi-LSTM, GRU, and attention based models) as well as some machine learning classification models including SVM, Random Forest, Logistic Regression and others for our morphological segmentation task. While a critical point in any DL model is the training data size, we decided to create a rich hand annotated Persian lexicon which is the only segmented corpus for Persian words. Using this lexicon we evaluated our presented models as well as the effect of training data size on results. Moreover, we evaluated and tested our models on some limited datasets for Czech and Finnish languages. Experimental results show our Bi-LSTM model performs slightly better in boundary prediction, however the results of classification-based approaches overcome the DL models in percentage of completely correctly segmented words.

Acknowledgments

The research was supported by OP RDE project No. CZ.02.2.69/0.0/0.0/16_027/0008495, International Mobility of Researchers at Charles University, and by grant No. 19-14534S of the Grant Agency of the Czech Republic. It has been using language resources developed, stored, and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the

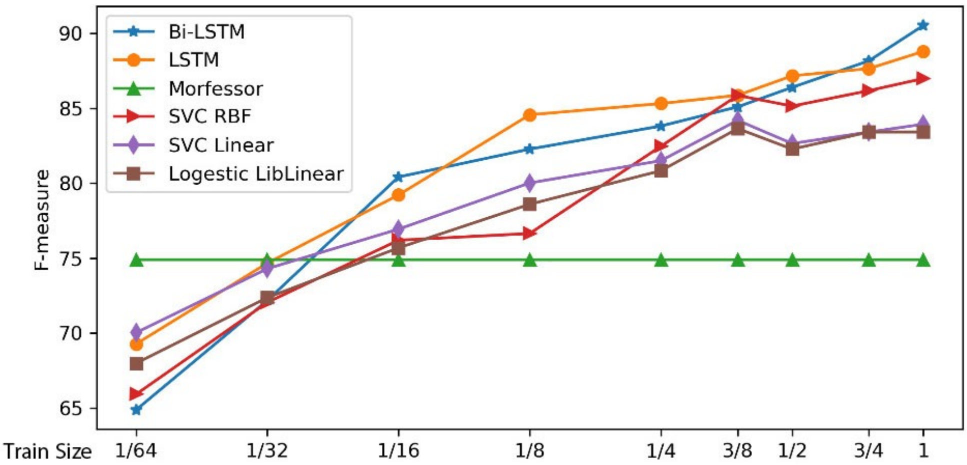


Figure 4: The effect of Persian training data size on boundary detection F-measure.

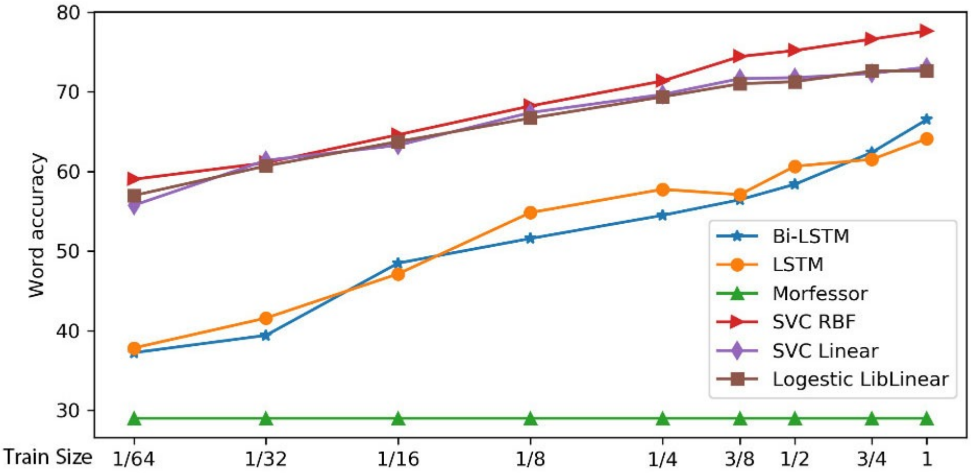


Figure 5: The effect of Persian training data size on whole-word segmentation accuracy.

Model	with semi			without semi		
	P% / R% / F%	W%	Ch%	P% / R% / F%	W%	Ch%
LSTM	90.09 / 87.57 / 88.80	64.10	93.20	91.15 / 74.76 / 82.15	51.42	89.53
Bi-LSTM	92.50 / 88.65 / **90.53**	66.51	94.37	89.19 / 77.18 / 82.75	52.58	89.64
SVC, Kernel: linear	85.86 / 82.20 / 83.94	73.08	94.45	81.67 / 77.75 / 79.66	68.17	92.62
SVC, Kernel: poly, Degree: 2	89.57 / 85.86 / 87.67	**78.72**	95.72	86.52 / 82.96 / 84.71	75.66	94.43
SVC, Kernel: rbf	89.71 / 84.42 / 86.99	77.61	95.52	86.34 / 80.96 / 83.56	74.39	94.08
SVC, Kernel: poly, Degree: 5	89.77 / 83.91 / 86.74	77.17	95.45	86.11 / 80.11 / 83.00	73.00	93.90
SVC, Kernel: poly, Degree: 3	89.58 / 85.89 / 87.70	78.70	**95.73**	86.30 / 83.02 / 84.63	75.30	94.39
Logistic Regression, Solver: sag	87.55 / 79.66 / 83.42	72.60	94.39	83.83 / 75.75 / 79.58	68.61	92.77
Logistic Regression, Solver: liblinear	87.55 / 79.60 / 83.42	72.63	94.39	83.84 / 75.75 / 79.59	68.63	92.78
Logistic Regression, Solver: lbfgs	87.49 / 79.78 / 83.46	72.64	94.39	83.74 / 75.59 / 79.46	68.47	92.73
KNeighbors, Neighbors: 5	82.47 / 86.22 / 84.30	73.12	94.56	82.19 / 76.34 / 79.15	67.36	92.52
KNeighbors, Neighbors: 10	86.22 / 82.47 / 84.30	73.12	94.56	82.19 / 76.34 / 79.15	67.36	95.52
KNeighbors, Neighbors: 30	90.23 / 86.69 / 88.42	78.64	95.73	82.19 / 76.34 / 79.15	67.36	92.52
Ada Boost, Estimators: 100	83.34 / 64.10 / 72.46	58.21	90.83	75.17 / 51.87 / 61.39	52.95	87.87
Decision Tree	88.25 / 87.05 / 87.65	76.83	95.38	88.24 / 86.05 / **87.13**	**75.92**	**95.21**
Random Forest, Estimators: 10	89.75 / 84.87 / 87.15	76.08	95.30	85.04 / 78.17 / 81.46	70.83	93.38
Random Forest, Estimators: 100	89.93 / 85.92 / 87.88	77.37	95.54	85.21 / 79.66 / 82.34	71.95	93.65
Bernoulli Naive Bayes	78.38 / 88.31 / 83.05	66.71	93.21	75.63 / 84.91 / 80.00	62.01	92.01
Perceptron MaxIteration: 50	83.98 / 74.45 / 78.93	65.07	92.52	75.41 / 77.28 / 76.34	62.51	90.05
Unsupervised MORFESSOR	69.58 / 81.10 / 74.90	29.01	83.28	71.16 / 81.88 / 76.14	30.33	83.48
Supervised MORFESSOR	82.13 / 92.94 / 87.20	59.56	91.60	81.60 / 92.24 / 86.60	58.84	90.80

Table 6: The effect of considering semi-space on training data when all training data are used.

Czech Republic (project LM2015071).

References

Ebrahim Ansari, Zdeněk Žabokrtský, Hamid Haghdoost, and Mahshid Nikravesh. 2019. Persian

Morphologically Segmented Lexicon 0.5. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, https://hdl.handle.net/11234/1-3011.

Mohsen Arabsorkhi and Mehrnoush Shamsfard. 2006. Unsupervised Discovery of Persian Morphemes. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters Demonstrations*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '06, pages 175–178. http://dl.acm.org/citation.cfm?id=1608974.1609002.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. Cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. http://arxiv.org/abs/1409.0473.

Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. 2011. Lessons from building a Persian written corpus: Peykare. *Language Resources and Evaluation* 45(2):143–164.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 18–26. https://doi.org/10.18653/v1/W16-1603.

Ryan Cotterell and Hinrich Schütze. 2017. Joint semantic synthesis and morphological analysis of the derived word. *CoRR* abs/1701.00946. http://arxiv.org/abs/1701.00946.

Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pylkkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. 2007. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Trans. Speech Lang. Process.* 5(1):3:1–3:29. https://doi.org/10.1145/1322391.1322394.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*. Association for Computational Linguistics, pages 21–30. https://doi.org/10.3115/1118647.1118650.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Comput. Linguist.* 27(2):153–198. https://doi.org/10.1162/089120101750300490.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2019. North Sámi morphological segmentation with low-resource semi-supervised sequence labeling. In *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*. Association for Computational Linguistics, Tartu, Estonia, pages 15–26. https://www.aclweb.org/anthology/W19-0302.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-Based Method for Unsupervised and Semi-Supervised Learning of Morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 1177–1185. https://www.aclweb.org/anthology/C14-1111.

Zellig S. Harris. 1970. *From Phoneme to Morpheme*, Springer Netherlands, Dordrecht, pages 32–67. https://doi.org/10.1007/978-94-017-6059-1_2.

Mahmoud Hesabi. 1988. *Persian Affixes and Verbs*, volume 1. Javidan.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 47–57. https://doi.org/10.18653/v1/N18-1005.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany, pages 62–70. https://doi.org/10.18653/v1/W16-2010.

Akbar Karimi, Ebrahim Ansari, and Bahram Sadeghi Bigham. 2018. Extracting an English-Persian Parallel Corpus from Comparable Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*.

Oskar Kohonen, Sami Virpioja, Laura Leppänen, and Krista Lagus. 2010. Semi-supervised extensions to Morfessor Baseline.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics* 3:157–167. https://doi.org/10.1162/tacl_a_00130.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '09, pages 209–217. http://dl.acm.org/citation.cfm?id=1620754.1620785.

Hanieh Poostchi, Ehsan Zare Borzeshi, and Massimo Piccardi. 2018. BiLSTM-CRF for Persian Named-Entity Recognition ArmanPersoNERCorpus: the First Entity-Annotated Persian Dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*.

Mohammad Sadegh Rasooli, Ahmed El Kholy, and Nizar Habash. 2013. Orthographic and Morphological Processing for Persian-to-English Statistical Machine Translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 1047–1051. https://www.aclweb.org/anthology/I13-1144.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 29–37. https://www.aclweb.org/anthology/W13-3504.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Association for Computational Linguistics, Gothenburg, Sweden, pages 84–89. https://doi.org/10.3115/v1/E14-4017.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *TACL* 1:255–266.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Hossein Taghi-Zadeh, Mohammad Hadi Sadreddini, Mohammad Hasan Diyanati, and Amir Hossein Rasekh. 2015. A new hybrid stemming method for Persian language. *Digital Scholarship in the Humanities* 32(1):209–221. https://doi.org/10.1093/llc/fqv053.

Sami Virpioja, Ville T. Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. 2011. Empirical comparison of evaluation methods for unsupervised learning of morphology. *TRAITEMENT AUTOMATIQUE DES LANGUES* 52(2):45–90.

Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. 2016. Morphological Segmentation with Window LSTM Neural Networks. In Dale Schuurmans and Michael P. Wellman, editors, *AAAI*. AAAI Press, pages 2842–2848.

Zdeněk Žabokrtský, Magda Ševčíková, Milan Straka, Jonáš Vidra, and Adéla Limburská. 2016. Merging data resources for inflectional and derivational morphology in Czech. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association, Paris, France, pages 1307–1314.

Combining Lexical Substitutes in Neural Word Sense Induction

Nikolay Arefyev[1,2], **Boris Sheludko**[1,2], **and Alexander Panchenko**[3]

[1]Samsung R&D Institute Russia, Moscow, Russia
[2]Lomonosov Moscow State University, Moscow, Russia
[3]Skolkovo Institute of Science and Technology, Moscow, Russia
narefjev@cs.msu.ru, b.sheludko@samsung.com, a.panchenko@skoltech.ru

Abstract

Word Sense Induction (WSI) is the task of grouping of occurrences of an ambiguous word according to their meaning. In this work, we improve the approach to WSI proposed by Amrami and Goldberg (2018) based on clustering of lexical substitutes for an ambiguous word in a particular context obtained from neural language models. Namely, we propose methods for combining information from left and right context and similarity to the ambiguous word, which result in generating more accurate substitutes than the original approach. Our simple yet efficient improvement establishes a new state-of-the-art on WSI datasets for two languages. Besides, we show improvements to the original approach on a lexical substitution dataset.

1 Introduction

Ambiguity, including lexical ambiguity, is one of the fundamental properties of natural languages and is a central challenge for NLP and its applications. Lexical ambiguity is a common situation when a single word has several meanings which can be either closely related (*coffee* as a plant, as a drink, or as beans for preparing that drink) or entirely unrelated (*band* as a musical group or as a strip of material). Consider the word *book* in *book a flight* or *buy a book*. Depending on the expressed meaning, machine translation systems should translate this word differently, search engines should find different information, personal digital assistants should take different actions, etc.

Word sense induction (WSI) is the task of clustering of occurrences of an ambiguous word according to their meaning. For evaluation of WSI systems, text fragments containing ambiguous words are hand-labeled with senses from some sense inventory (a dictionary or a lexical ontology). WSI systems are given text fragments only and should cluster them into some a priori unknown number of clusters (unlike Word Sense Disambiguation systems, which are also given the sense inventory).

Words that can appear instead of an ambiguous word in a particular context, also known as lexical contextual substitutes, are very helpful for WSI because possible substitutes strongly depend on the expressed meaning of the ambiguous word. For instance, for the word *build* possible substitutes are *manufacture*, *make*, *assemble*, *ship*, *export* if it is used in the *manufacturing some goods* sense and *erect*, *rebuild*, *open* for the *constructing a building* sense. Baskaya et al. (2013) proposed generating substitutes using n-gram language models and had shown one of the best results at the SemEval-2013 WSI shared task for English (Jurgens and Klapaftis, 2013). Later Amrami and Goldberg (2018) proposed generating contextual substitutes with a bidirectional neural language model (biLM) ELMo (Peters et al., 2018). With several other improvements, they had achieved new state-of-the-art results on the same dataset. However, their method simply unites substitutes generated independently from probability distributions $P(w_i|w_{i-1}...w_1)$ and $P(w_i|w_{i+1}...w_T)$ estimated by the forward and the backward ELMo LM independently, each given only one-sided context. This results in noisy substitutes when either left or right context is short or non-informative.

The **main contribution** of this paper is an approach that combines the forward and the backward distributions into a single distribution and fuses the similarity to the ambiguous word into the combined distribution. This allows taking into account all information we have about a particular ambiguous word occurrence for better substitutes generation. We compare several methods for combining distributions. Substitutes retrieved from the combined distribution perform much better for WSI achieving the a state-of-the-art on the SemEval 2013 dataset for English as well two datasets for Russian.

2 Related Work

The first methods to word sense induction were proposed already in the late 90s (Pedersen and Bruce, 1997; Schütze, 1998; Lin, 1998) with several competitions being organized to systematically evaluate various methods, including SemEval 2007 task 2 (Agirre and Soroa, 2007), SemEval 2010 task 14 (Manandhar et al., 2010) and SemEval 2013 task 13 (Jurgens and Klapaftis, 2013) for the English language, and RUSSE 2018 (Panchenko et al., 2018) for the Russian language.[1] Navigli (2012) provides a survey of word sense induction and related approaches. Methods for word sense induction can be broadly classified into three groups: context clustering approaches, word (ego-network) clustering, and latent variable models. We discuss these approaches below. Also, note that methods for learning word sense embedding (Camacho-Collados and Pilehvar, 2018) can be used to induce vector representations of senses from text.

2.1 Context/Vector Clustering Methods

This methods from this group represent a word instance by a vector that characterizes its context, where the definition of context can vary greatly. These vectors are subsequently clustered.

Early approaches, such as (Pedersen and Bruce, 1997; Schütze, 1998; Reisinger and Mooney, 2010) used sparse vector representations. Later approaches dense vector representations were adopted, e.g. Arefyev et al. (2018) and Kutuzov (2018) used weighted word embeddings (Mikolov et al., 2013) pre-trained on a large corpus to represent context of an ambiguous target word. Anwar et al. (2019) used contextualized (Peters et al., 2018) and non-contextualized (Mikolov et al., 2013) word embeddings to cluster occurrences of ambiguous occurrences of verbs according to their semantic frames.

The approach presented in this paper is also an instance of vector clustering methods. More specifically, it exploits contextual substitutes for the ambiguous word to differentiate between its senses. Baskaya et al. (2013) proposed using substitute vectors for WSI, and their system *AI-KU* was one of the best-performing systems at SemEval 2013. Alagić et al. (2018) proposed another approach which leverages lexical substitutes for unsupervised word sense induction. They perform clustering of contexts using the affinity propagation algorithm (Dueck and Frey, 2007). The similarity between instances is measured using three different measures based on cosine similarities between pre-trained word embeddings by Mikolov et al. (2013). One measure relies on an average of embeddings of context words. Another one relies on an average of embeddings of lexical substitutes (also combination of both measures is tested). Finally, Amrami and Goldberg (2018) proposed using neural language models and dynamic symmetric patterns establishing a new best result on this dataset. Their approach is described in details in Section 3 as a starting point for our method.

2.2 Word/Graph Clustering Methods

This group of methods cluster word ego-networks consisting of a single node (ego) together with the nodes they are connected to (alters) and all the edges among those alters. Nodes of an ego-network can be words semantically similar to the target word or context features relevant to the target. This line of work starts from the seminal work of (Widdows and Dorow, 2002) who used graph-based methods for unsupervised lexical acquisition. In this work, senses of the word were defined as connected components in a graph which excludes the ego. Véronis (2004), Biemann (2006), and Hope and Keller (2013) further developed this idea by performing clustering of nodes instead of the simple search for connected components. Pelevina et al. (2016) proposed to transform word embeddings to sense embeddings using graph clustering (Biemann, 2006). The obtained sense embeddings were used to solve the WSI task based on similarity computations between the context and the induced sense.

2.3 Latent Variable Methods

Methods from this group, define a generative process of the documents which include word senses as a latent variable and then perform estimation

[1] https://russe.nlpub.org

of the model from unlabeled textual data. For instance, Lau et al. (2013) relies on the Hierarchical Dirichlet Process (HDP) (Teh et al., 2006). Latent topics discovered in the training instances, specific to every word, are interpreted as word senses. Since the HDP is generative, also new instances can be assigned a sense topic. Latent variable model of Bartunov et al. (2016) is a Bayesian extension of Skip-gram (Mikolov et al., 2013) that automatically learns the number of word senses; it relies on the stick-breaking process. Amplayo et al. (2019) propose another graphical model which tackles the sense granularity problem, setting new state-of-the-art results for the SemEval 2010/2013 WSI datasets.

3 Bayesian Fusion of Lexical Substitutes from Bidirectional Language Models

In this section, we describe the method of word sense induction proposed by Amrami and Goldberg (2018), which is based on lexical substitutes generated given left and right context separately and then united together. Then we propose several methods to build a combined distribution incorporating information from left and right context as well as the similarity to the target word for better substitutes generation. For qualitative comparison, Table 1 lists lexical substitutes generated by different methods for several randomly selected sentences from the TWSI dataset (Biemann, 2012). For readability, we select either the top 10 predictions from the combined distributions or the union of the top 5 predictions from the forward and the backward distributions. The actual number of substitutes may be smaller due to duplicates appearing after lemmatization of substitutes.

3.1 Baselines: No Fusion (Union of Substitutes)

We base our approach on the method by Amrami and Goldberg (2018) (named **original** hereafter), which has achieved state-of-the-art results on the SemEval-2013 dataset for English WSI. Suppose c is the target ambiguous word and l, r are its left and right contexts. First, the method employs pretrained forward and backward ELMo LMs (Peters et al., 2018) to estimate probabilities for each word w to be a substitute for c given only the left context $P_{fwd}(w|l)$ or only the right context $P_{bwd}(w|r)$. Second, from the top K most probable words of each distribution L substitutes are sampled. This is done S times resulting in S representatives of the original example consisting of $2L$ substitutes each. Then TF-IDF BoW vectors for all representatives of all examples of a particular ambiguous word are built. Finally, agglomerative clustering is performed on the obtained representations with a fixed number of clusters. To provide more information to the LMs the target word can be included in the context using the technique called dynamic patterns. For example, given the sentence *These apples are sold everywhere* instead of *'These _'* the forward LM receives *'These apples and _'* and instead of *'_ are sold everywhere'* the backward LM receives *'_ and apples are sold everywhere'*. The underscore denotes the position for which the language model predicts possible words.

Thus, lexical substitutes are obtained **independently** from the forward and the backward LM and then **united**. For soft clustering required by the SemEval-2013 dataset, the probability distribution over clusters for each example is estimated from the number of representatives of this example put in each cluster. For the RUSSE (the Russian WSI) datasets we further convert soft clustering into hard clustering by selecting the most probable cluster for each example.

The second baseline (named **base**) simplifies the **original** method by skipping sampling and using $S = 1$ representative consisting of the union of the top K predictions from each LM. While being simpler and deterministic, this modification also delivers better results on RUSSE. Additionally, we have found that baselines with dynamic patterns translated into Russian perform worse than their counterparts without patterns (**original-no-pat** and **base-no-pat**) on RUSSE. This is in line with the ablations study from Amrami and Goldberg (2018) who found that the patterns are useful for verbs and adjectives but almost useless for nouns which the RUSSE datasets consist of. Interestingly, our best models perform better without dynamic patterns on all datasets.

3.2 Fusion at the Level of LM Distributions

During preliminary experiments, we have found that uniting substitutes retrieved from the forward and the backward LM independently results in lots of substitutes not related to the target word sense. For instance, consider the first example in Table 1 where the ambiguous word is the last word of the sentence. The backward LM simply

base-no-pat	base	BComb-LMs	BComb-3
It offers courses at the Undergraduate and Post Graduate levels in various subjects.			
sept, industry, feb, university, discipline, nov, dec, language, field, oct	offer, **course**, teach, subject, style, **topic**, background, size, include, provide	profession, subject, industry, university, discipline, sector, guise, language, field, department	field, occupation, language, discipline, sector, guise, profession, subject, department, industry
Wakeboards with a three - stage rocker push more water in front of the wakeboard, making the ride slower but riders are able to jump higher off the water.			
slightly **trip** perfect become **journey** climb **trek** bit speed	faster landing climb bend rid harder speed walk bike	jump incline slope climb bend **trek**	dive incline climb **trek** slope **journey** crawl
The couple were married on the bride's family estate at Ballyhooly, Cork, Ireland; afterwards the couple set up home at Caddington Hall.			
tree bear **residence** holiday wedding vacation live farm cottage	marry mansion be live castle farm cottage move divorce	honeymoon croft ranch vineyard homestead **residence** farmhouse wedding farm cottage	farm ranch **residence** wedding croft cottage homestead

Table 1: **Examples of generated lexical substitutes**: baselines and our models. Contexts are from the TWSI dataset. Ambiguous word is underlined, substitutes intersecting with human-generated are **bold**. Here **base** is the baseline approach of Amrami and Goldberg (2018) and **base-no-pat** is its simplified version without patterns, while **BComb-LMs** and **BComb-3** are our models described in Section 3.

predicts all words which can appear before dot resulting mostly infrequent abbreviations. Dynamic patterns help a little, but there is still no context available for the backward LM to disambiguate the target word. To solve this problem we propose combining distributions from the forward and the backward LM first and then taking the top K words from this combined distribution. We experiment with the following combinations.

3.2.1 Average (avg)

This straightforward method of fusion of two distributions computes an average of forward and backward distributions (no information about the target word is used):

$$\begin{aligned} P(w|l,r) &= \frac{1}{2}(P(w|l) + P(w|r)) \\ &= \frac{1}{2}(P_{fwd} + P_{bwd}). \end{aligned} \tag{1}$$

3.2.2 Bayesian Combination of LMs (BComb-LMs)

Using Bayes' rule and supposing left and right context are independent given possible substitutes we estimate fused distribution as follows:

$$\begin{aligned} P(w|l,r) &= \frac{P(l,r|w)P(w)}{P(l,r)} \\ &= \frac{P(l|w)P(r|w)P(w)}{P(l,r)} \\ &\propto \frac{P(w|l)P(w|r)}{P(w)}. \end{aligned} \tag{2}$$

The numerator is estimated as $P_{fwd}P_{bwd}$, but pretrained ELMo LMs don't contain frequencies of the words in the vocabulary, so we cannot directly estimate the denominator. Instead we approximate it with Zipf distribution (the vocabulary is sorted by frequency):

$$P(w) \propto \frac{1}{(k + rank(w))^s}, \tag{3}$$

where k and s are hyperparameters: the first is needed to perform adjustment for frequent words while the second defines how quickly word frequency drops as its rank grows.

3.2.3 Three-Way Bayesian Combination (BComb-3)

Substitutes should not only be compatible with context, but also similar to the target word c. Amrami and Goldberg (2018) integrate information about the target word using dynamic patterns, but here we propose a probabilistic approach of fusion of forward and backward distribution with the information about the target word. Namely, we estimate similarity using a scaled dot product of output embeddings from ELMo:

$$P(w|c) \propto \exp(\frac{emb_w^T emb_c}{Temperature}), \quad (4)$$

where $Temperature$ is a hyperparameter which allows scaling this distribution to fit to the LM distributions. Similarly to *BComb-LMs* and supposing the target word is independent from the context given possible substitutes (which can be interpreted as fixing a particular sense of the target):

$$P(w|l, c, r) \propto \frac{P(w|l)P(w|r)P(w|c)}{P^2(w)}. \quad (5)$$

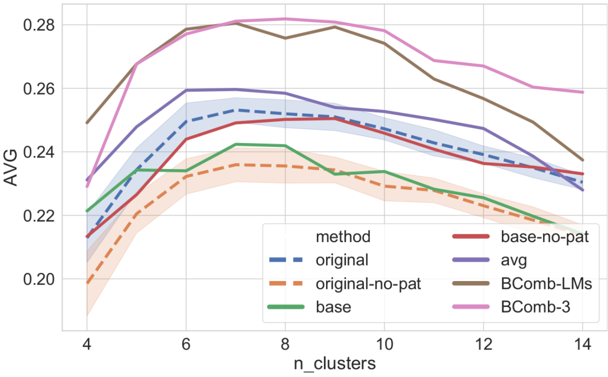


Figure 1: **SemEval 2013 task 13**: geometric average of fNMI and fB^3 with respect to the number of clusters per word. Hyperparameters are selected on the TWSI dataset (Biemann, 2012).

4 Evaluation and Results

To evaluate the quality of our proposed approach, we performed three experiments. Two of them are based on WSI datasets coming from the shared tasks for English (Jurgens and Klapaftis, 2013) and Russian (Panchenko et al., 2018). The last experiment compares substitutes generated by the original and our methods to the human-generated substitutes using a lexical substitution dataset for English (Biemann, 2012).

4.1 Experiment 1: SemEval 2013 WSI Task

4.1.1 Experimental Setup

First, we evaluate our methods on the SemEval-2013 dataset for English WSI (Jurgens and Klapaftis, 2013). The dataset contains contexts for 50 ambiguous words, including 20 nouns, 20 verbs, and 10 adjectives. It provides 20-100 contexts per word, and 4,664 contexts in total, which were gathered from the Open American National Corpus and annotated with senses from WordNet. We used this dataset as the test set and tuned all hyperparameters except for the number of clusters on the TWSI dataset (Biemann, 2012).

Evaluation metrics. Performance is measured with two cluster comparison measures: Fuzzy NMI (fNMI) and Fuzzy B-Cubed (fB^3) as defined in (Jurgens and Klapaftis, 2013).

4.1.2 Discussion of Results

Figure 1 shows a geometric average (AVG) between Fuzzy Normalized Mutual Information (fNMI) and Fuzzy B-Cubed F1 (fB^3) depending on the number of clusters. Following Amrami and Goldberg (2018), Table 2 reports the results for the number of clusters equal to 7 which is the average number of senses in SemEval-2013. BComb-3 shows the best results closely followed by BComb-LMs, while the *avg* combination methods performs worse but still outperforms baseline methods.

4.2 Experiment 2: RUSSE 2018 WSI Task

4.2.1 Experimental Setup

For the Russian language we test our methods on the *active-dict* and the *bts-rnc* datasets from the RUSSE 2018 WSI shared task (Panchenko et al., 2018). These datasets are split into dev and test parts containing non-overlapping ambiguous words. The *bts-rnc* dataset relies on contexts sampled from the Russian National Corpus (RNC)[2] and annotated based on the sense inventory of the Large Explanatory Dictionary of Russian[3]. The dev set contains 30 ambiguous words and 3,491 contexts. The test set contains 51 ambiguous words and 6,556 contexts. The *active-dict* dataset is based on the Active Dictionary of Russian, which is an explanatory dictionary (Apresjan, 2011). For each sense, contexts were extracted from the glosses and examples of this dictionary. The train/development set has 85 ambiguous words and 2,073 contexts. The test set has 168 ambiguous words and 3,729 contexts.

[2] http://ruscorpora.ru/en
[3] http://gramota.ru/slovari/info/bts

Model	fNMI	fB^3	AVG
One sense for all	0.000	**0.623**	0.000
One sense per instance	0.071	0.000	0.000
Best competition results (Jurgens and Klapaftis, 2013)			
AI-KU	0.065	0.390	0.159
Unimelb	0.060	0.483	0.170
Best after-competition results			
(Amrami and Goldberg, 2018)	0.113	0.575	0.254
(Amplayo et al., 2019)	0.096	**0.622**	0.244
This paper			
avg	0.120	0.562	0.260
BComb-LMs	**0.139**	0.566	0.280
BComb-3	0.135	0.586	**0.281**

Table 2: **SemEval 2013 task 13**: comparison to the previous best results. Following Amrami and Goldberg (2018) the number of clusters is 7, other hyperparameters are selected on the TWSI dataset.

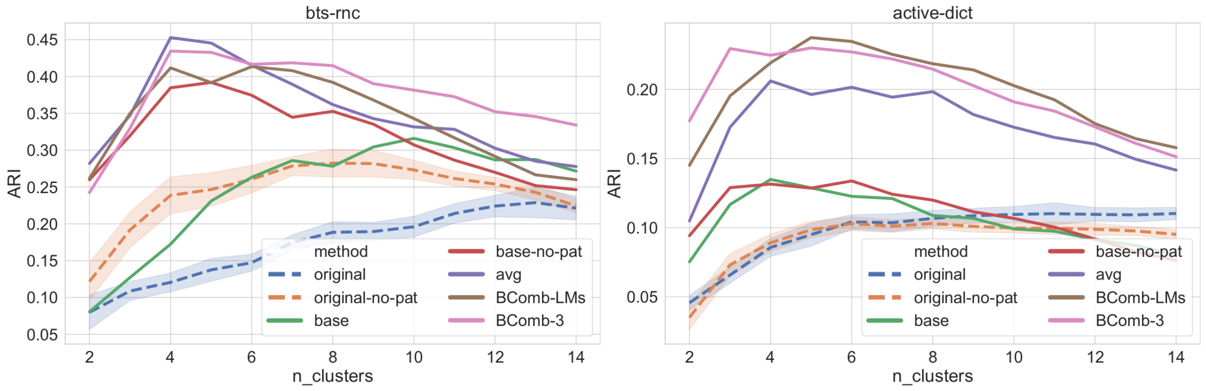


Figure 2: **RUSSE-2018 development sets**: ARI with respect to the number of clusters per word. Hyperparameters are selected on the TWSI dataset.

Evaluation metrics. Performance is measured using Adjusted Rand Index (ARI) (Hubert and Arabie, 1985).

4.2.2 Discussion of Results

Figure 2 shows results on the development set using the same hyperparameters used for SemEval-2013. Despite being selected on an English WSI dataset, they perform surprisingly well. Similarly to SemEval-2013, on *active-dict* BComb methods outperform Avg by a large margin. However, on *bts-rnc* dataset, Avg seems to be the best performing method which we attribute to suboptimal hyperparameters. For our final submissions to the leaderboard reported in Table 3 we selected hyperparameters on the development set corresponding to each dataset and with these hyperparameters BComb methods are indeed better than Avg. We report results for (i) a fixed number of clusters (selected on the development sets) and for (ii) individual number of clusters for each word selected by maximizing the silhouette score of clustering[4]. Using individual number of clusters consistently improves results for all our methods.

4.3 Experiment 3: TWSI Lexical Substitution

4.3.1 Experimental Setup

In the third experiment, we evaluated the quality of lexical substitutes generated by our methods comparing them with human-generated ones from the

[4] https://scikit-learn.org/stable/modules/clustering.html#silhouette-coefficient

Model	bts-rnc Test	active-dict Test
avg	0.355 / 0.436	0.254 / 0.255
BComb-LMs	0.464 / **0.502**	0.304 / 0.331
BComb-3	0.455 / 0.473	0.300 / **0.332**
post compet'n best results	0.348	0.307
competition 1st best result	0.351	0.264
competition 2nd best result	0.281	0.236

Table 3: **RUSSE 2018 test sets**: comparison to the previous best results. The number of clusters is selected on corresponding development sets (like other hyperparameters) / using silhouette score.

TWSI dataset by Biemann (2012). Version 2.0 of the dataset was used in our experiments. The dataset is composed of 1,012 frequent nouns with 2.26 senses per word on average. For these nouns, the dataset provides 145,140 annotated sentences sampled from Wikipedia. Besides, it features a sense inventory, where each sense is represented with a list of words that can substitutes.

Evaluation Metrics Performance is measured using precision and recall among top $K = 10$ lexical substitutions.

4.3.2 Discussion of Results

Table 4 reports the results. One should carefully interpret these results since humans generate precise but not exhaustive lists of substitutes. For instance, for the sentence *Henry David Thoreau wrote the famous phrase, "In wildness is the preservation of the world."* BComb-3 model generates the following substitutes: *dictum, proverb, poem, motto, epitaph, slogan, quote, aphorism, maxim* from which only *slogan* and *maxim* were generated by humans. As one may observe, according to metrics, both base methods with patterns and BComb-3 generate much more human-like substitutes than their counterparts that do not take into account the target word (base-no-pat and BComb-LMs) with BComb-3 being a little better. Examples of generated substitutes are shown in Table 1.

Model	rec.@10	prec.@10
base	0.115	0.035
base-no-pat	0.058	0.020
avg	0.093	0.032
BComb-LMs	0.073	0.025
BComb-3	**0.127**	**0.041**

Table 4: **TWSI lexical substitution**: comparison our method to the baseline model by Amrami and Goldberg (2018) on the dataset of human-generated lexical substitutes.

5 Conclusion

We proposed a new method for neural word sense induction which improves the approach of Amrami and Goldberg (2018). We show that substantially better results can be obtained if the information from the forward and the backward LMs is combined in a more principled way using Bayesian fusion of distributions rather than a simple union of substitutes generated independently from each distribution. More specifically, this work shows that integration of the forward and the backward distributions retrieved from neural LMs and the similarity to the target word results in better-generated substitutes for ambiguous words, which enabled achieving a new state-of-the-art for WSI for two languages.

Acknowledgements

We thank three anonymous reviewers for their helpful comments and suggestions, especially the Reviewer #2. Besides, we are grateful to all colleagues with whom we discussed our ideas with, especially to Artem Grachev, Dima Lipin, and Alex Nevidomsky.

References

Eneko Agirre and Aitor Soroa. 2007. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 7–12.

Domagoj Alagić, Jan Šnajder, and Sebastian Padó. 2018. Leveraging lexical substitutes for unsupervised word sense induction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Reinald Kim Amplayo, Seung won Hwang, and Min Song. 2019. Autosense model for word sense induction. In *AAAI*.

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural biLM and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 4860–4867. https://www.aclweb.org/anthology/D18-1523.

Saba Anwar, Dmitry Ustalov, Nikolay Arefyev, Simone Paolo Ponzetto, Chris Biemann, and Alexander Panchenko. 2019. HHMM at SemEval-2019 task 2: Unsupervised frame induction using contextualized word embeddings. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Minneapolis, Minnesota, USA, pages 125–129. https://www.aclweb.org/anthology/S19-2018.

Valentina Apresjan. 2011. Active dictionary of the russian language: theory and practice. *Meaning-Text Theory* 2011:13–24.

Nikolay Arefyev, Pavel Ermolaev, and Panchenko Alexander. 2018. Russian word sense induction by clustering averaged word embeddings. In *Proceedings of the 24th International Conference on Computational Linguistics and Intellectual Technologies (Dialogue2018)*. RGGU, Moscow, Russia.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Osman Baskaya, Enis Sert, Volkan Cirik, and Deniz Yuret. 2013. AI-KU: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 300–306. https://www.aclweb.org/anthology/S13-2050.

Chris Biemann. 2006. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing*. Association for Computational Linguistics, pages 73–80.

Chris Biemann. 2012. Turk bootstrap word sense inventory 2.0: A large-scale resource for lexical substitution. In *LREC*. pages 4038–4042.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research* 63:743–788.

Delbert Dueck and Brendan J Frey. 2007. Non-metric affinity propagation for unsupervised image categorization. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, pages 1–8.

David Hope and Bill Keller. 2013. UoS: A Graph-Based System for Graded Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA, 1, pages 689–694. http://www.aclweb.org/anthology/S13-2113.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2(1):193–218.

David Jurgens and Ioannis Klapaftis. 2013. SemEval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 290–299. https://www.aclweb.org/anthology/S13-2049.

Andrey Kutuzov. 2018. Russian word sense induction by clustering averaged word embeddings. In *Proceedings of the 24th International Conference on Computational Linguistics and Intellectual Technologies (Dialogue2018)*. RGGU, Moscow, Russia.

Jey Han Lau, Paul Cook, and Timothy Baldwin. 2013. unimelb: Topic Modelling-based Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM): SemEval 2013*. Atlanta, Georgia, USA, volume 2, pages 307–311. http://www.aclweb.org/anthology/S13-2051.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of ICML*. Madison, WI, USA, volume 98, pages 296–304.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 task 14: Word sense induction &disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Uppsala, Sweden, pages 63–68. https://www.aclweb.org/anthology/S10-1011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Process-*

ing Systems 26. Curran Associates, Inc., Lake Tahoe, NV, USA, pages 3111–3119.

Roberto Navigli. 2012. A quick tour of word sense disambiguation, induction and related approaches. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, pages 115–129.

Alexander Panchenko, Anastasiya Lopukhina, Dmitry Ustalov, Konstantin Lopukhin, Nikolay Arefyev, Alexey Leontyev, and Natalia Loukachevitch. 2018. Russe'2018: a shared task on word sense induction for the russian language. *Computational Linguistics and Intellectual Technologies* pages 547–564.

Ted Pedersen and Rebecca Bruce. 1997. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. Providence, RI, pages 197–207.

Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 174–183. https://doi.org/10.18653/v1/W16-1620.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California, pages 109–117. http://www.aclweb.org/anthology/N10-1013.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational linguistics* 24(1):97–123.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association* 101(476):1566–1581. https://doi.org/10.1198/016214506000000302.

Jean Véronis. 2004. Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language* 18(3):223–252.

Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational linguistics*. Taipei, Taiwan, pages 1–7. https://doi.org/10.3115/1072228.1072342.

Detecting Clitics Related Orthographic Errors in Turkish

Uğurcan Arıkan **Onur Güngör** **Suzan Uskudarli**
Department of Computer Engineering, Bogazici University, Istanbul, Turkey
{ugurcan.arikan,onurgu,suzan.uskudarli}@boun.edu.tr

Abstract

For the spell correction task, vocabulary based methods have been replaced with methods that take morphological and grammar rules into account. However, such tools are fairly immature, and, worse, nonexistent for many low resource languages. Checking only if a word is well-formed with respect to the morphological rules of a language may produce false negatives due to the ambiguity resulting from the presence of numerous homophonic words. In this work, we propose an approach to detect and correct the “de/da” clitic errors in Turkish text. Our model is a neural sequence tagger trained with a synthetically constructed dataset consisting of positive and negative samples. The model’s performance with this dataset is presented according to different word embedding configurations. The model achieved an F_1 score of 86.67% on a synthetically constructed dataset. We also compared the model’s performance on a manually curated dataset of challenging samples that proved superior to other spelling correctors with 71% accuracy compared to the second best (Google Docs) with 34% accuracy.

1 Introduction

Misspellings can change the meanings of words and, consequently, of sentences, which can lead to major miscommunication and frustration. This paper focuses on a common spelling error in Turkish, namely the spelling of the “de/da” clitic. Its written form (“de” and “da”) depends on the vowel harmony rule that is based on the last vowel of the word previous to the conjunction. When the final vowel of the prior word is in {e,i,ö,ü} the clitic is written as “de”, otherwise (in {a,ı,o,u}) it is written as “da”. For example, in the sentence “Selin de burada” meaning “Selin is also here”, the last word before the clitic (“de”) is “Selin” whose final vowel is “i”. Thus, the clitic is written as “de”. Whereas, in the sentence “Fatma da burada” meaning “Fatma is also here”, the last word before the clitic (“da”) is “Fatma” whose final vowel is “a”, causing the clitic to be written as “da”.

The “de/da” clitic in Turkish is a conjunction when it is written separately and has the same meaning as "as well", "too", and "also" in English. In addition to being a conjunction, the “de” and “da” homonyms may be used as locative suffixes meaning “at” or “in”. For example, the word “araba” (car) with the suffix “-da” (“arabada”) means “in the car”. Although the “de/da” clitic in the meaning of conjunction must always be written separately, it is commonly confused with the locative suffix "de/da" and incorrectly written concatenated to the previous word.

The misspelling of the "de/da" clitics alter the meaning of a sentence, and possibly render it meaningless. For example, when the clitic in the sentence "Araba da gördüm" is misspelled as "Arabada gördüm", changes the meaning from “I also saw a car” to “I saw it in the car”. This type of misspelling happens to be one of the most pervasive and annoying misspellings in Turkish. One can frequently encounter expressions of criticism and frustration in this regard.

Morphological analysis is not very useful in spelling correction of “de/da” since in most cases new meaningful words form when it is written as a suffix. As such, most of the Turkish spell checkers perform poorly or not at all. The only way to differentiate between them is to take the sentence context into account.

This work proposes a neural sequence tagger model to detect and correct “de/da” errors. The model employs a conditional random field (CRF) for choosing the best prediction based on score vectors that are provided by a multilayered bidirectional LSTM. Words in input sentences are replaced with word embeddings trained with different algorithms. The model is tested with various combinations of these pretrained embeddings on a synthetically constructed dataset, where the best scores were obtained when all three embeddings were used that yielded an F1-Measure of 86.67%. It was also tested on a manually created more challenging dataset.

The main contributions of this work are:

- state-of-the-art spelling corrector that handles the "de/da" misspellings in Turkish,
- a comparative analysis of alternative word embedding models for spell checking Turkish sentences,
- a dataset of Turkish sentences with difficult to detect "de/da" errors, and
- a demo website for spellchecking sentences including "de/da" cases.

The remainder of the paper is organized as follows: Section 2 presents background information needed to follow this work, Section 3 discusses the state-of-the-art and current solutions to spelling corrections in Turkish, Section 4 discusses the model and experiments, Section 5 presents an evaluation of the proposed model, Section 6 reflects on observations and provides insights about the future work, and finally concluding remarks are given in Section 7.

2 Background

2.1 Clitic, Conjunction and Locative Suffix

A clitic is a morpheme that is syntactically independent but phonologically dependent and attached to a host. It has the syntactic characteristics of a word, but depends phonologically on another word or phrase.

A conjunction is a word that syntactically connects other words or larger constituents while also expressing a semantic relationship between them. Some conjunction examples from English include *and*, *or*, *but* and *if*. The clitic "de/da" can be given as an example conjunction in Turkish.

The locative suffix indicates the locative case, which is the grammatical case that conveys a location. In Turkish, the locative case is specified by the suffix "-de/-da".

Our model focuses on the Turkish clitic "de/da" that means "also, as well, too" and must always be written separately. It is commonly confused with the locative suffix "de/da" that means "at" or "in" as explained in Section 1.

2.2 The CoNLL Sentence Representation

In 2003 a data format was introduced for the *CoNLL-2003 shared task: Language-independent named entity recognition* (Kim Sang and De Meulder, 2003). In this format, each word is on a separate line with an empty line after each sentence. The first item of a line is a word, the second is a part-of-speech (POS) tag, the third is a syntactic chunk tag, and the fourth is the named entity tag. To represent sequences of meaningful words, the chunks and entities use B-TYPE to indicate the beginning and I-TYPE to indicate being *inside* the phrase. The TYPE refers to the type of the entity (i.e., person). Numerous datasets for NLP tasks utilize this format for interoperability. A word with tag "O" (outside) is considered as not being a part of a phrase. The CoNLL format is often used for publishing datasets. We use a variant of this format for representing correct and incorrect sentence samples as detailed in Section 4.1.

2.3 Word Embeddings

Word embeddings are the vector representations of different sets of words. They are one of the most widely utilized methods used for language representation. Word embeddings are capable of capturing the semantic and syntactic similarity between words. In this work the word embeddings that are used are GloVe (Pennington et al., 2014), fastText (Grave et al., 2018) and Word2Vec (Mikolov et al., 2013).

Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) is an unsupervised learning algorithm to acquire word vectors form words. It works on word to word global co-occurrence matrices and is successful in capturing semantic information. It combines global matrix factorization and local context window methods to create word embeddings.

FastText (Grave et al., 2018) is an open-source, lightweight library for very fast text classification introduced by Facebook in 2016. FastText is proposed as an extension of Word2Vec that trains models given labeled texts, performs predictions, and evaluates models. It is a hierarchical classifier where labels are represented in a binary tree that

Benim	ben+Pron+Pers+A1sg+Pnon+Gen
de	de+Conj
aklım	akıl+Noun+A3sg+P1sg+Nom
sen**de**	sen+Pron+Pers+A2sg+Pnon+Loc
kaldı	kal+Verb+Pos+Past+A3sg

Table 1: The morphological analysis of a Turkish sentence (My mind also remains with you) with both the clitic and the affix forms of “de”.

facilities much faster model training without loss of accuracy. FastText breaks words into n-grams creating sub-words that are fed to the model to obtain the embeddings of each word. The tri-grams of the word *selam* are *sel*, *ela*, and *lam*. In this way information about patterns within words are captured, which enables out of vocabulary words to be processed.

Word2Vec models generate word embeddings with a two-layer neural network that creates a set of feature vectors for words in a corpus.

2.4 Turkish Language

Turkish is an agglutinative language, where complex words are derived by stringing together morphemes. In agglutinative languages a sequence of affixes are attached to the end of the words. Table 1 shows the morphological analysis of the sentence (using the ITU NLP pipeline (Eryiğit, 2014)): “Benim de aklım sende kaldı.”, which roughly translates to “My mind remains with you too” (a manner of expressing that one’s thoughts are with someone). More literally it translates to “Also, my mind has remained with you.” This sentence includes both forms of “de”, which are shown in bold. The “de” following “Benim” refers to ”also”. The affix “de” within “sende” is locative and means at you (in English this is expressed as with you).

The morphological analysis of Turkish sentences can get very complex. It is rather difficult for non native speakers to learn the ordering of affixes and to distinguish among the clitics. Even native speakers may have trouble distinguishing the intended meaning and will need to clarify the context. These complexities present significant challenges to building language supporting tools for Turkish. Although, machine learning approaches show promise.

3 Related Work

Zemberek is a collection of natural language processing tools for Turkish and is capable of various tasks including morphological analysis, tokenization and sentence boundary detection and basic spell checker. It is also used as the spell checker for LibreOffice. However, it is not capable of detecting the misspelling of the clitic "de/da" as it does not make a semantic analysis on the sentence.(Akın and Akın, 2007)

ITU Turkish Natural Language Processing Pipeline can make syntactic and morphological analysis of raw Turkish sentences, although it is not capable of making a semantic analysis and thus fails to classify and correct spellings of Turkish "de/da" clitic (Eryiğit, 2014).

The spelling correctors for Turkish do not satisfactorily correct misspellings of the “de/da” clitic as they are limited to the morphological analysis of words which is insufficient for accurately classify them. Google, Microsoft Office, and LibreOffice all have different spell checkers for Turkish but none of them present satisfactory results in the case of handling the “de/da” clitics in Turkish. Their accuracy is significantly lower compared to our model as will be detailed in Section 5.

4 Experiments and Results

4.1 Data

To train the model, sentences with both correct and incorrect spellings of the clitic “de/da” are required. For this purpose, incorrect sentences have been generated from the correct sentences from a corpus consisting of approximately 75 million Turkish sentences extracted from various websites, novels and news sites (Yildiz et al., 2016). Since the corpus was extracted from novels and news sites, the sentences are assumed to include only a few or no orthographic errors. Thus, the spellings of the “de/da” cases are considered to be correct when written separately, attached as a locative suffix, or used as a conjunction. Note that some words simply end with “de/da” and these suffixes are not due to locative morphemes (i.e., ‘ziyade’ meaning plentiful). However, such cases are few and considered negligible.

To generate incorrectly spelled forms of “de/da” samples, two simple actions are performed: (1) append the separately written “de/da” to its preceding word and (2) separate the “de/da” suffixes

	Train	Dev	Test
Sentences	15,203	3,729	2,070
Tokens	383,066	94,232	51,226

Table 2: The number of sentences and tokens for the training, development, and test dataset used in training our models.

from the words that contain them. For example, for the sentence “Kedi de gördüm” (meaning “I also saw a cat”), the sentence “Kedide gördüm” (meaning “I saw it at the cat”) is generated by concatenation. Both are syntactically correct sentences but have very different meanings. The sentence “Evde kalıyorum” meaning “I am staying at home” which uses the locative suffix “de/da” correctly, the sentence “Ev de kalıyorum” is generated. The resulting sentence is an incorrectly separated “de/da”, which translates to “I am staying also home”, which doesn’t make sense.

The generated sentences are tagged in a manner like the CoNLL NER tags (Section 2.2). We tag incorrectly spelled terms with “B-ERR” and all others with “O” (other), such as:

Correct sentence	Incorrect sentence
`Onlar O`	`Onlarda B-ERR`
`da O`	`'Sende O`
`'Sende O`	`kalsın O`
`kalsın O`	`, savcılığa O`
`, savcılığa O`	`verirsin O`
`verirsin O`	`'O`
`'O`	`dediler O`
`dediler O`	`. O`
`. O`	

The dataset consisting of sentences whose words are tagged with “B-ERR” and “O” are divided into training, development, and test sets (Table 2).

In addition to the this synthetically constructed dataset, a dataset consisting of 100 Turkish sentences with misspelled forms of “de/da” is formed manually. The sentences in this second dataset is created so that they are syntactically correct but semantically challenging to understand[1].

4.2 Model

A multilayered bidirectional LSTM and CRF based model (Akbik et al., 2018) that uses pretrained embeddings was considered suitable for our problem since it achieved the state-of-the-art results for named entity recognition, part-of-speech tagging and chunking tasks.

4.3 Experimental Setup

The initial task was to train the model with Turkish word embeddings. For this task, GloVe was used with the dimension size of 300 and window size of 15. The pretrained word vectors for Turkish were obtained from the model trained on Common Crawl and Wikipedia using fastText (Grave et al., 2018). The pretrained Word2Vec vectors are for Turkish with dimension size 300 (Güngör and Yıldız, 2017). The models were trained using Continuous Bag of Words (CBOW), with position-weights, dimension size of 300, character n-grams of length 5, and a window size of 5 and 10 negatives.

Parameter optimization was performed to achieve the best F$_1$ scores. During hyperparameter optimization, the training was performed for 10 epochs using fastText embeddings for all possible configurations for the following criteria:

- batch size: [8, 16, 32, 64]
- RNN layer size: [1, 2, 3, 4]
- learning rate: [0.05, 0.1, 0.15, 0.2]
- hidden size: [16, 32, 64, 128, 256]

The hyperparameters with the highest F$_1$ score are: batch size=16, RNN layer size=2, learning rate=0.2, and hidden size=256. These parameter values were used to train models with different word embedding configurations for 150 epochs. All models were trained on a PC with GPU GeForce RTX2080 with 32 GB RAM. A single training took approximately 10 hours to complete.

5 Results and Evaluation

A total of seven different models were trained with the optimal parameters. The embedding types used were GloVe (Pennington et al., 2014), fastText (Grave et al., 2018) and Word2Vec. These embeddings were also combined by concatenating them to form a new embedding with a higher number of dimensions. Furthermore, two baseline models were used for comparison purposes. Baseline model baseline$_1$ considers only the separately written “de/da” as correct, falsely classifying the correctly spelled locative suffix “de/da” as a misspelling. Baseline model baseline$_2$ considers only

[1] Both this and the synthetic dataset is shared at https://github.com/derlem/kanarya

Model			BL		P	R	F_1
G	fT	W	1	2	(%)	(%)	(%)
			+		10.60	25.67	15.00
				+	59.89	74.32	66.33
+					87.09	81.53	84.22
	+				87.05	79.73	83.23
		+			87.67	79.50	83.39
+	+				90.55	81.98	86.05
+		+			89.79	81.83	85.63
	+	+			87.59	80.03	83.64
+	+	+			**91.56**	**82.28**	**86.67**

Table 3: A comparison of the results our model trained with various combinations of the Glove (G), fastText (fT) and Word2Vec(W) methods on a synthetically constructed dataset against two baseline models (BL-1 & BL-2). P, R and $\mathbf{F_1}$ refer to the precision, recall, and $\mathbf{F_1}$ measures.

the suffix form of "de/da" to be correct, falsely classifying the correctly spelled "de/da" conjunction as a misspelling. The results of these models are shown in Table 3.

Figure 1 shows some of the challenging sentences that where spelling errors and were correctly identified using our best model. The erroneous words are shown with a red bounding box. In these examples, the second sentence correctly identifies "çokta" as an error. In Turkish, when "-de/da" is to follow a work that ends with of the letters "p, ç, t, k, s, ş, h, f", "-de/da" becomes "-te/-ta". However, as a grammatical term it is referred to as "de/da" and is the more common case.

Finally, we examined the performance of various configurations of our model with other well-known spellcheckers for the 100 manually curated challenging sentences. Table 4 shows that our models performed significantly better than others. The best model utilizes the Word2Vec embeddings with an accuracy of 71% while the second best accuracy was achieved by Google Docs with 34%.

6 Discussion and Future Work

The work presented in this paper created a state-of-the-art model that achieved a much higher accuracy in detecting the "de/da" misspellings in Turkish when compared to existing spell correctors. Our model currently only addresses the misspelling of the "de/da" clitic. Further work is needed to integrate this work with morphological analysis to yield a more complete spell checker for Turkish.

Maça iyide başlamıştık aslında ama olmadı.

Bu adam öyle aslında çokta kötü bir adam değil.

Ya ders calış yada çık dışarıda oyna.

Sonunda bizde derin öğrenmeye geçtik.

Kalemleri ve kitabı ev de kalmış.

Belkide Galatasaray'ı şampiyonluktan ettik.

Onuda yaptığında gidebilirsin.

Figure 1: The errors caught by our model with the best configuration on challenging sentences.

Recently, much success is being reported regarding the use of BERT (Devlin et al., 2019), which we are currently working on to obtain word embeddings, which we expect to further increase the performance of our model.

The proposed model can be integrated with various platforms, ranging from text editors to social media to messaging platforms. The "de/da" distinctions can be especially difficult for foreigners who are attempting to learn Turkish as a second language. Such spellcheckers could be very useful in assisting learning. We are also working on developing and API and a demo service that make this work more accessible. The scope of access will be limited by the resources we are able to acquire.

7 Conclusions

We developed a deep learning model to detect orthographic errors caused by the misspelling of the clitic "de/da" in Turkish. This model uses various word embeddings to train a model for the named entity recognition task for this clitic. The best model achieved an F_1 score of 86.67% on a synthetically constructed dataset. To our knowledge, this is the state-of-the-art result for spelling correction for the misspellings of "de/da" clitics in Turkish. These results are very encouraging. We intend to extend the model with a similar case as well as make all the resources related to this work accessible as open source.

Ours			Others				Acc
G	fT	W	ITU	G	W	📄	(%)
+							55
	+						64
		+					71
+	+						66
+		+					67
	+	+					69
+	+	+					65
				+			34
					+		29
			+				0
						+	0

Table 4: Results of spell checking of semantically challenging sentences. G, fT, and W refer to Glove, fastText and Word2Vec respectively. ITU is the ITU NLP Pipeline for Turkish, and the icons G, W, and 📄 the spellcheckers of Google Docs, Microsoft Office, and LibreOffice.

Acknowledgements

This work is supported by BAP (the Bogazici University Research Fund) under Grant No. 13083 and the Turkish Ministry of Development under the TAM Project, number DPT2007K120610.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*. pages 1638–1649.

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek: An open source NLP framework for Turkic languages. *Structure* 10:1–5.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. pages 4171–4186.

Gülşen Eryiğit. 2014. ITU Turkish NLP web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 1–4.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Onur Güngör and Eray Yıldız. 2017. Linguistic features in Turkish word representations. In *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE, pages 1–4.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. USA, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Eray Yildiz, Caglar Tirkaz, H Bahadir Sahin, Mustafa Tolga Eren, and Omer Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *30th AAAI Conference on Artificial Intelligence*.

Benchmark Dataset for Propaganda Detection in Czech Newspaper Texts

Vít Baisa and **Ondřej Herman** and **Aleš Horák**
Natural Language Processing Centre
Masaryk University, Faculty of Informatics
Botanická 68a, Brno
{xbaisa,xherman1,hales}@fi.muni.cz

Abstract

Propaganda of various pressure groups ranging from big economies to ideological blocks is often presented in a form of objective newspaper texts. However, the real objectivity is here shaded with the support of imbalanced views and distorted attitudes by means of various manipulative stylistic techniques.

In the project of Manipulative Propaganda Techniques in the Age of Internet, a new resource for automatic analysis of stylistic mechanisms for influencing the readers' opinion is developed. In its current version, the resource consists of 7,494 newspaper articles from four selected Czech digital news servers annotated for the presence of specific manipulative techniques.

In this paper, we present the current state of the annotations and describe the structure of the dataset in detail. We also offer an evaluation of bag-of-words classification algorithms for the annotated manipulative techniques.

1 Introduction

State and pressure groups propaganda is a very well studied phenomenon from the sociological point of view (Herman and Chomsky, 2012; Zhang, 2013; Paul and Matthews, 2016). With the spread of digital media, the influence of propaganda news grows rapidly (Helmus et al., 2018) and the consequences of public opinion manipulation reach new levels (Woolley and Howard, 2017).

The main way of self-protection against such propaganda influence lies in careful verification of the presented information sources. Nevertheless, psycholinguistic evidence (Fazio et al., 2015) shows that a prevailing opinion often outweighs even direct knowledge. Computational tools that could warn against possible manipulation in the text can thus offer an invaluable help even to an informed reader.

In the following text, we are presenting the first results of a research project aimed at automatic analysis of the *style* of a newspaper text to identify a presence of specific manipulative techniques. In the first phase, a specific tool for expert annotations of selected news from 4 Czech internet media sites was developed (Baisa et al., 2017). This tool has now been used to obtain 7,494 annotated articles with detailed manipulative techniques annotations of texts expressing e.g. blaming, demonizing, relativizing, labelling, or fear mongering. The following Section 2 provides detailed information about the dataset characteristics and content. In Section 3, an evaluation of 10 classification techniques and their results with the benchmark dataset is presented.

2 The Benchmark Dataset

The Propaganda benchmark dataset currently contains data from two successive years. The first part is based on two sets of articles from 2016. The newspaper texts were extracted from four newspaper media domains[1] which were previously scrutinized by annotators as possible sources of pro-Russian propaganda. The downloaded cleaned data were merged with the annotation data stored separately in a SPSS[2] format (converted with the GNU PSPP tool[3]) which is used widely in Social science research. The result is a corpus with metadata (structure attributes) available for full-text

[1] sputnik.cz, parlamentnilisty.cz, ac24.cz and www.svetkolemnas.info.

[2] https://www.ibm.com/products/spss-statistics

[3] https://www.gnu.org/software/pspp/

Proceedings of Recent Advances in Natural Language Processing, pages 77–83,
Varna, Bulgaria, Sep 2–4 2019.

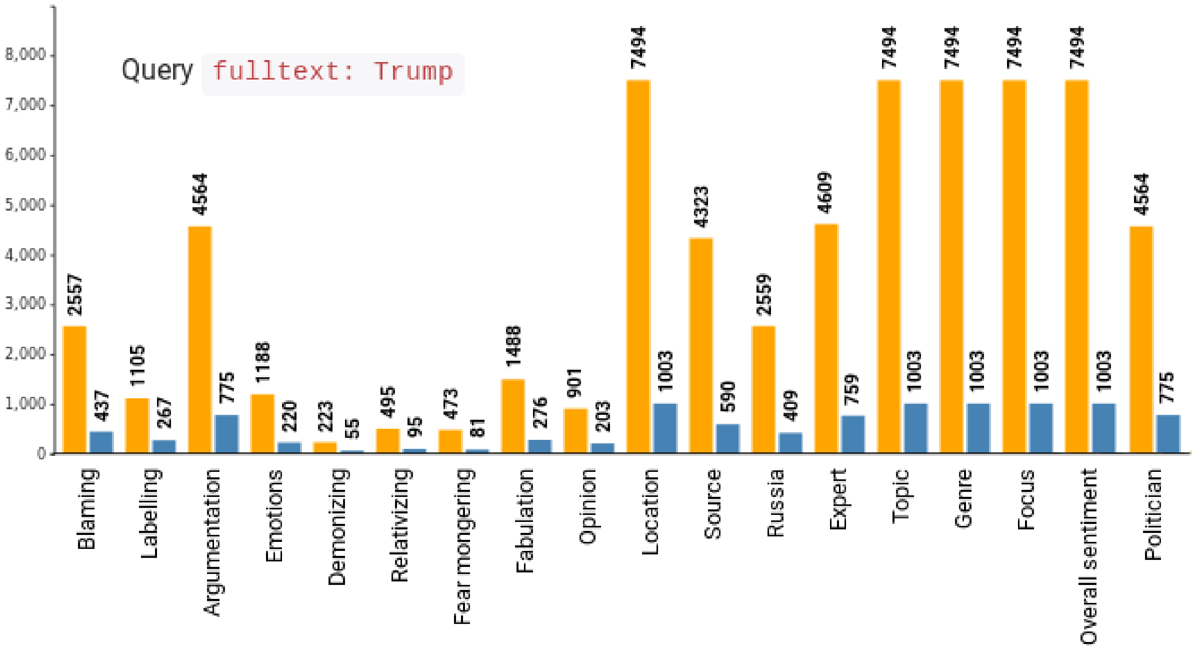


Figure 1: Numbers of articles with significant attribute values (not null, neutral or missing) in the whole collection of 7,494 documents. The first (yellow) columns show numbers for the whole collection and the second (blue) columns show an example of a filtered subset of articles containing the word "Trump".

search in the Sketch Engine corpus manager (Kilgarriff et al., 2014). As far as we know, this is the first corpus of propaganda text annotated for detailed ensemble of manipulative techniques. The full document texts were thus supplemented with the following attributes (see Figure 1 for representation of particular attributes in the dataset):

a) **Blaming**: does the text accuse someone of something?

b) **Labelling**: the text uses specific labels – short and impactful phrases or words – to describe a person or a group.

c) **Argumentation**: does the text present facts or arguments (logical, emotional, etc.) to support the main claim?

d) **Emotions**: What is the main emotion the text is trying to evoke in the reader? Anger, hate, fear.

e) **Demonizing**: is the “enemy” and/or his/her goals or interests presented in the text as being evil?

f) **Relativizing**: are the presented actions of a person, group or party being relativized?

g) **Fear mongering**: is the text trying to appeal to fear, uncertainty or other threat?

h) **Fabulation**: does the text contain unsubstantiated, overstated or otherwise incorrect claims?

i) **Opinion**: does the author of the text present his or hers personal opinion?

j) **Location**: what is the main location the text talks about?

k) **Source**: is the text presented as being based on a specific source?

l) **Russia**: is the topic related to Russia?

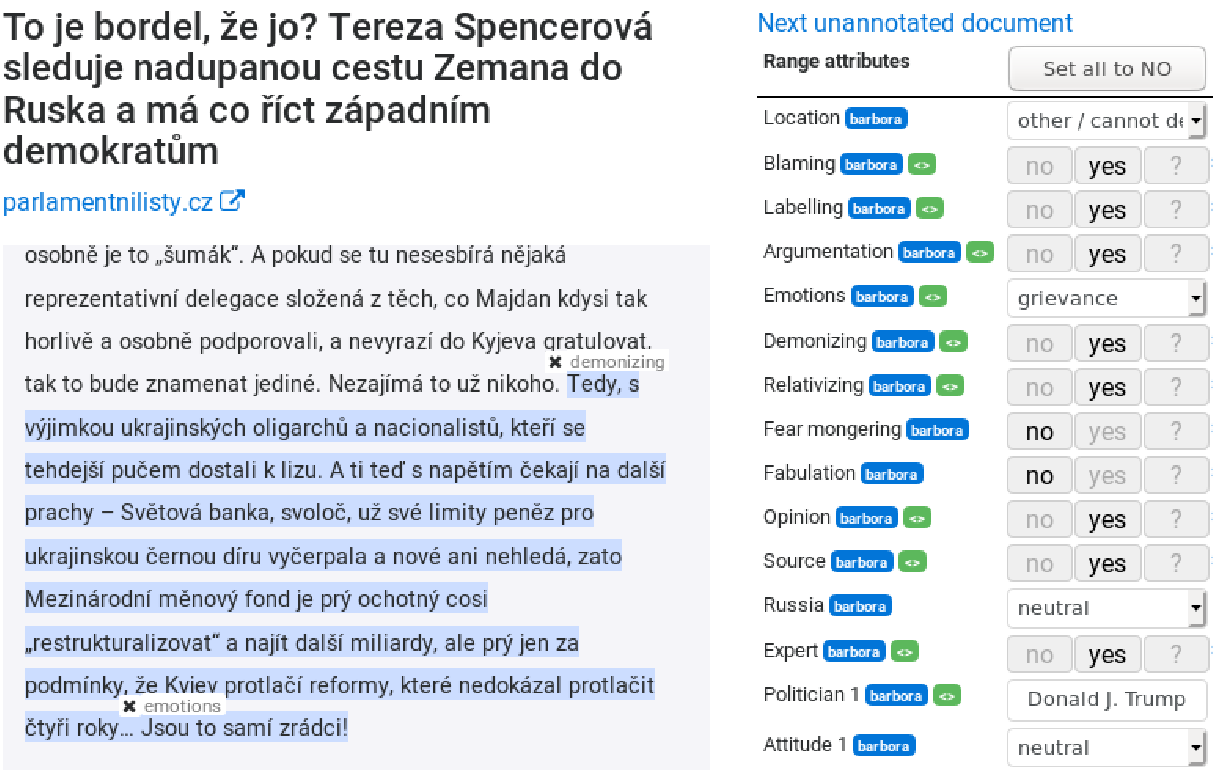
m) **Expert**: is the text or opinion in the text presented as being supported by an expert?

n) **Attitude to a politician**: neutral, negative, positive for up to 3 mentioned politicians.

o) **Topic**: migrant crisis, domestic politics, etc.

p) **Genre**: report, interview, or commentary.

q) **Focus**: foreign, domestic, can’t be determined.



To je bordel, že jo? Tereza Spencerová sleduje nadupanou cestu Zemana do Ruska a má co říct západním demokratům

parlamentnilisty.cz

osobně je to „šumák". A pokud se tu nesesbírá nějaká reprezentativní delegace složená z těch, co Majdan kdysi tak horlivě a osobně podporovali, a nevyrazí do Kyjeva gratulovat, tak to bude znamenat jediné. Nezajímá to už nikoho. Tedy, s výjimkou ukrajinských oligarchů a nacionalistů, kteří se tehdejší pučem dostali k lizu. A ti teď s napětím čekají na další prachy – Světová banka, svoloč, už své limity peněz pro ukrajinskou černou díru vyčerpala a nové ani nehledá, zato Mezinárodní měnový fond je prý ochotný cosi „restrukturalizovat" a najít další miliardy, ale prý jen za podmínky, že Kviev protlačí reformy, které nedokázal protlačit čtyři roky... Jsou to samí zrádci!

Next unannotated document

Range attributes	Set all to NO
Location barbora	other / cannot de
Blaming barbora <>	no [✓ yes] ?
Labelling barbora <>	no [✓ yes] ?
Argumentation barbora <>	no [✓ yes] ?
Emotions barbora <>	grievance
Demonizing barbora <>	no [✓ yes] ?
Relativizing barbora <>	no [✓ yes] ?
Fear mongering barbora	[✓ no] yes ?
Fabulation barbora	[✓ no] yes ?
Opinion barbora <>	no [✓ yes] ?
Source barbora <>	no [✓ yes] ?
Russia barbora	neutral
Expert barbora <>	no [✓ yes] ?
Politician 1 barbora <>	Donald J. Trump
Attitude 1 barbora	neutral

Figure 2: An example of (a part of) an annotated article with ranges showing *demonizing* and *grievance* as a value of the *emotions* attribute.

r) **Overall sentiment**: neutral, negative, or positive.

The second part, articles from the same domains published in 2017, has undergone a fine-grained annotation using a specific data processing and annotating tool (Baisa et al., 2017), which requires the annotators not only to specify the respective attribute values but also enrich them with particular phrase examples. The annotators were asked to amend each significant attribute value (not null, neutral or missing) by marking a particular block (or blocks) of text that offer the evidence of the value. The attributes are split into two groups. The attributes a) to n), denoted as *range attributes*, are bound to a sequence of words from the text, the attributes o) to r), i.e. the document attributes, are related to the article as a whole. An example of annotated range attributes can be seen in Figure 2. Unfortunately, due to the complexity of the annotation process, there was only one annotator per document and the inter-annotator agreement could not be decided.

The text of the articles has been extracted from the media server web pages, then tokenized using *unitok* (Michelfeit et al., 2014) and morphologically annotated using *majka* (Šmerk, 2009) and *desamb* (Šmerk, 2010). The dataset thus allows complicated full-text search in the articles. The size of the data (sub)sets is in Table 1.

Table 1: Text statistics of the two parts of the benchmark dataset.

	2016	2017	Total
Tokens	2,774,178	930,304	3,704,482
Words	2,331,116	781,725	3,112,842
Sentences	144,097	49,140	193,237
Paragraphs	50,554	17,264	67,818
Documents	5,500	1,994	7,494

3 Dataset Evaluation

We have performed the dataset evaluation to express the baseline accuracy of assigning the labels automatically using 10 machine learning classifiers. The classifiers were trained with the 20,000 most frequent lemmata present in the corpus, with the text transformed to a numerical vector format using bag-of-words using TF-IDF weighting.

Table 2: Classifier Accuracy

	Blaming	Labelling	Argumentation	Emotions	Demonizing	Relativizing	Fear mongering	Fabulation	Opinion	Location	Source	Russia	Expert	Topic	Genre	Focus	Overall sentiment	Server
dummy	.59	.79	.69	.81	.96	.93	.91	.74	.86	.41	.60	.70	.74	.32	.89	.53	.75	.63
bernoulli_nb	.67	.78	.59	.74	.87	.85	.84	.75	.84	.56	.63	.73	.63	.53	.91	.72	.72	.80
multinomial_nb	.67	.79	.70	.81	.96	.93	.91	.74	.86	.52	.60	.71	.74	.54	.89	.86	.75	.72
nearest_centroid	.66	.71	.62	.63	.74	.80	.75	.71	.75	.58	.60	.55	.67	.56	.80	.66	.65	.73
passive_aggressive	.70	.79	.72	.77	.96	.94	**.92**	.78	.84	.74	.67	.79	.80	.69	.95	.85	.73	.92
random_forest	.69	.81	.74	.81	.96	.93	.92	.77	.87	.67	.68	.80	.80	.63	.92	.85	.76	.88
ridge	**.72**	**.82**	**.75**	**.81**	.96	.94	.92	**.79**	.89	.75	**.70**	.80	.81	.71	.96	**.87**	**.78**	.91
sgd_elasticnet	.71	.82	.73	.81	.96	**.94**	.92	.78	.89	.76	.70	.82	.80	.71	.96	.87	.77	.93
sgd_l1	.70	.81	.72	.81	.96	.94	.92	.78	.89	.76	.70	**.82**	.81	.70	.96	.87	.77	**.94**
sgd_l2	.70	.82	.73	.81	**.96**	.94	.92	.78	**.89**	**.76**	.70	.81	.80	.71	.96	.87	.77	.92

3.1 Selected Classifiers

For the evaluation, we have chosen a representative subset of classification techniques, which are often employed in bag-of-words tasks for attribute value estimation. The resulting set of classifiers includes:

- `dummy`: a baseline, classifies every instance as the majority class present in the input data.
- `passive_aggressive`: an efficient Perceptron-like classifier (Crammer et al., 2006).
- Two Naive Bayes variants: `bernoulli_nb` assumes that the data is Bernoulli distributed, while `multinomial_nb` assumes a Multinomial distribution (McCallum et al., 1998).
- Three different Support Vector Machine classifiers trained using stochastic gradient descent: `sgd_l1` with L1 regularization, `sgd_l2` with L2 regularization and `sgd_elasticnet` with Elasticnet regularization (Zhang, 2004).
- `ridge` is a regularized linear regression based classifier (Rifkin and Lippert, 2007).
- `random_forest`: An ensemble of decision tree classifiers is built on samples drawn from the training set. The resulting class during the classification is obtained by taking the most common class as assigned by each of the decision trees (Breiman, 2001).
- `nearest_centroid`: computes a per-class mean of examples during training, the classification then assigns class according to the closest mean (McIntyre and Blashfield, 1980).

Table 3: Examples of word sentiment data used in the experiment.

Czech	English	Positive	Negative
neschopný	incapable	0	0.75
čistý	clean	0.5	0
poměrný	proportional	0.25	0.5
hojný	abundant	0.125	0
přijatelný	acceptable	0.625	0
závadný	harmful	0	0.375
přístupný	accessible	0.625	0
zastrčený	inserted	0.125	0
úslužný	obliging	0.75	0

3.2 Evaluation Strategy

The final accuracy scores have been obtained by stratified 3-fold cross validation to evaluate the performance of the classifiers. In the 3-fold cross validation, documents were first grouped by their classes. Each of these classes was then divided into 3 parts. The training set for the investigated classifier then consists of two parts of all groups and the test set consists of the remaining parts of all groups. There are three different ways to select which of the parts will go into the training and the evaluation sets. Each classifier has been evaluated three times, once with each of these ways or folds. The resulting score was computed as the average of the three scores obtained for each of the folds.

Table 4: Classifier prediction accuracy sorted by the weighted F1-score which takes into account imbalanced attribute classes. The resulting accuracy is compared to the baseline accuracy of the majority class.

	best classifier	weighted F1	accuracy	baseline	difference
Demonizing	sgd_l2	.85	.96	.96	.00
Genre	sgd_elasticnet	.84	.96	.89	.07
Server	sgd_l1	.83	.94	.63	.31
Relativizing	sgd_elasticnet	.82	.94	.93	.01
Fear mongering	passive_aggressive	.81	.92	.91	.01
Opinion	sgd_l2	.79	.89	.86	.03
Focus	ridge	.77	.87	.53	.34
Labelling	ridge	.73	.82	.79	.03
Expert	ridge	.73	.81	.74	.07
Russia	sgd_l1	.71	.82	.70	.12
Emotions	ridge	.70	.81	.81	.00
Fabulation	ridge	.70	.79	.74	.04
Overall sentiment	ridge	.70	.78	.75	.04
Location	sgd_l2	.68	.76	.41	.36
Argumentation	ridge	.65	.75	.69	.06
Blaming	ridge	.65	.72	.59	.13
Topic	sgd_elasticnet	.64	.71	.32	.39
Source	ridge	.63	.70	.60	.10

3.3 Evaluation Metrics

Each trained classifier predicts the class for a document based on its text. By comparing the results to the dataset gold standard data, each of the classifier was evaluated by means of its attribute-related accuracy, precision, recall, and F1 score. The accuracy results are summarized in Table 2 and compared with the `dummy` baseline accuracy in Table 4.

3.4 Correlations of Attributes and Sentiment Coefficients

The set of article attributes contains several items which express sentiment values, either to the article as a whole or to a mentioned politician. We have evaluated the possibility of using the article sentiment analysis to predict the corresponding attribute values for the texts.

The paragraph sentiment analysis results were explicitly expressed as an average score of positivity and negativity of particular words. A list of 6,261 words was prepared as projections of SentiWordNet (Baccianella et al., 2010) scores via the Czech WordNet (Rambousek et al., 2018; Horák et al., 2008) database, see Table 3 for examples. Each paragraph received an average value of *only positive* words, *only negative* words and of their *average score* computed as a difference between word positivity and negativity. The overall document scores were then computed as a maximum positive paragraph score, maximum negative paragraph score and maximum and minimum of the average word score for each paragraph.

Each of the resulting document sentiment scores were evaluated for a correlation[4] with positive and negative values of the selected attributes annotated in the data. The results are presented in Table 5. None of the attributes has proven really strong correlation, but several attributes partly correlate with the maximum negative sentiment of the document. Interestingly, there is no correlation in case of the *emotions* attribute.

4 Conclusion and Future Directions

We have introduced a new benchmark dataset for propaganda manipulative techniques detection in Czech newspaper texts. The dataset contains 7,494 documents annotated for the presence of eight manipulative techniques and 10 document attributes relevant for propaganda detection. The

[4] Computed as Spearman's correlation coefficient with statistical significance.

Table 5: Correlations of selected attributes and document sentiment analysis scores. The † symbol denotes statistically significant values ($p < 0.05$) of Spearman's correlation coefficient.

Attribute	max positive		max negative		max average		min average	
blaming	0.18	†	**0.23**	†	0.17	†	-0.23	†
demonizing	0.11	†	**0.13**	†	0.11	†	-0.12	†
fear mongering	0.16	†	**0.18**	†	0.16	†	-0.18	†
emotions compassion	0.02		-0.00		0.03		-0.00	
emotions fear	-0.07	†	0.02		-0.07	†	-0.02	
emotions hate	0.06	†	0.04		0.06		-0.04	
emotions grievance	-0.00		-0.05		-0.00		0.05	
overall sentiment	0.16	†	**0.18**	†	0.16	†	-0.18	†
attitude1	0.04	†	**0.04**	†	0.04	†	-0.04	†
attitude2	0.10	†	**0.15**	†	0.09	†	-0.16	†
attitude3	0.13	†	**0.13**	†	0.11	†	-0.13	†
attitude avg	0.13	†	**0.14**	†	0.11	†	-0.15	†

dataset is currently being expanded with the third part of documents from 2018 and it is planned to be released for public access after this expansion.

We have evaluated the current data with 10 current classification techniques. Regularized linear regression and Support vector machines are able to classify the data with the best accuracies, even though the manipulative techniques need to employ extra features to significantly improve over the baseline.

In the currently running experiments, we are preparing new evaluation of the dataset using detailed stylometric features and distributed semantic representations of the texts.

Acknowledgments.

This project was partially supported by the Grant Agency of Masaryk University within the project MUNI/G/0872/2016.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of LREC 2010*. pages 2200–2204.

Vít Baisa, Ondřej Herman, and Aleš Horák. 2017. Manipulative Propaganda Techniques. In *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2017*. pages 111–118.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7(Mar):551–585.

Lisa K Fazio, Nadia M Brashier, B Keith Payne, and Elizabeth J Marsh. 2015. Knowledge does not protect against illusory truth. *Journal of Experimental Psychology* 144(5):993–1002.

Todd C Helmus, Elizabeth Bodine-Baron, Andrew Radin, Madeline Magnuson, Joshua Mendelsohn, William Marcellino, Andriy Bega, and Zev Winkelman. 2018. *Russian Social Media Influence: Understanding Russian Propaganda in Eastern Europe*. Rand Corporation.

Edward Herman and Noam Chomsky. 2012. A propaganda model. *Media and cultural studies: Keyworks* pages 204–230. Reproduced from Manufacturing Content, 1988.

Aleš Horák, Karel Pala, and Adam Rambousek. 2008. The Global WordNet Grid Software Design. In *Proceedings of the Fourth Global WordNet Conference, University of Szegéd*. pages 194–199.

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. The Sketch Engine: ten years on. *Lexicography* 1(1):7–36.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. pages 41–48.

Robert M McIntyre and Roger K Blashfield. 1980. A nearest-centroid technique for evaluating the minimum-variance clustering procedure. *Multivariate Behavioral Research* 15(2):225–238.

Jan Michelfeit, Jan Pomikálek, and Vít Suchomel. 2014. Text tokenisation using unitok. In Aleš Horák and Pavel Rychlý, editors, *RASLAN 2014*. Tribun EU, Brno, Czech Republic, pages 71–75.

Christopher Paul and Miriam Matthews. 2016. The Russian ”firehose of falsehood” propaganda model. *Rand Corporation* pages 2–7.

Adam Rambousek, Aleš Horák, and Karel Pala. 2018. Sustainable long-term WordNet development and maintenance: Case study of the Czech WordNet. *Cognitive Studies/Études cognitives* (18).

Ryan M Rifkin and Ross A Lippert. 2007. Notes on regularized least squares. *Computer Science and Artificial Intelligence Laboratory Technical Reports* https://dspace.mit.edu/handle/1721.1/37318.

Pavel Šmerk. 2009. Fast Morphological Analysis of Czech. In Petr Sojka and Aleš Horák, editors, *Third Workshop on Recent Advances in Slavonic Natural Language Processing*. Masaryk University, pages 13–16.

Pavel Šmerk. 2010. *K počítačové morfologické analýze češtiny (in Czech, Towards Computational Morphological Analysis of Czech)*. Ph.D. thesis, Faculty of Informatics, Masaryk University.

Samuel C Woolley and Philip N Howard. 2017. Computational propaganda worldwide: Executive summary. *Working Paper* 2017(11).

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, page 116.

Jianqing Zhang. 2013. *The Propaganda Model and the Media System in China*. Dartmouth College.

Diachronic Analysis of Entities by Exploiting Wikipedia Page revisions

Pierpaolo Basile
University of Bari Aldo Moro
Dept. of Computer Science
Bari, Italy
pierpaolo.basile@uniba.it

Annalina Caputo
ADAPT Centre
Trinity College Dublin
Dublin, Ireland
annalina.caputo@adaptcentre.ie

Seamus Lawless
ADAPT Centre
Trinity College Dublin
Dublin, Ireland
seamus.lawless@adaptcentre.ie

Giovanni Semeraro
University of Bari Aldo Moro
Dept. of Computer Science
Bari, Italy
giovanni.semeraro@uniba.it

Abstract

In the last few years, the increasing availability of large corpora spanning several time periods has opened new opportunities for the diachronic analysis of language. This type of analysis can bring to the light not only linguistic phenomena related to the shift of word meanings over time, but it can also be used to study the impact that societal and cultural trends have on this language change. This paper introduces a new resource for performing the diachronic analysis of named entities built upon Wikipedia page revisions. This resource enables the analysis over time of changes in the relations between entities (concepts), surface forms (words), and the contexts surrounding entities and surface forms, by analysing the whole history of Wikipedia internal links. We provide some useful use cases that prove the impact of this resource on diachronic studies and delineate some possible future usage.

1 Introduction

The availability of large corpora spanning different time periods has encouraged researchers to analyse language from a diachronic perspective. Language is dynamic and detecting significant linguistic shifts in the meaning and usage of words is a crucial task for both social and cultural studies and for Natural Language Processing applications. Recent work focusing on the automatic detection of the semantic shift of words has adopted diachronic (or dynamic) word embeddings (Kim et al., 2014; Hamilton et al., 2016b; Kulkarni et al., 2015). This type of work represents words as vectors in a *semantic* space where the proximity between word vectors indicate the existence of a semantic relationship between the terms involved. The diachronic analysis is then performed by building a different semantic space for each period of investigation and aligning vectors belonging to different spaces in order to make them comparable. The variations in the similarity between the word vectors in two different spaces marks possible changes in the context of appearance of that word. This is used as a proxy indicator of change, either cultural, social or semantic, associated with the occurrence of that specific word. This kind of work has generated a variety of resources for the diachronic analysis of word meanings, covering different time periods, languages, and genres.

While the broader area of automatic detection of semantic shift of words is gaining momentum, only little effort has focused on the more specific problem of analysing the semantic shift of named entities. This problem has a huge impact on the correct identification of entities in context, with repercussions on many natural language processing problems, such as entity linking and search, aspect-based sentiment analysis and event detection (Kanhabua and Nørvåg, 2010b; Tahmasebi et al., 2012; Georgescu et al., 2013).

Generally, an entity has a clear referent and what evolves is the context in which it appears or the surface form used to refer to it. In this work, we build a resource that tracks how the surface forms used to link an entity change over time by taking into account the revisions of Wikipedia pages. In doing so, we also extract time-dependent contexts of each mention of a link in Wikipedia

Proceedings of Recent Advances in Natural Language Processing, pages 84–91,
Varna, Bulgaria, Sep 2–4 2019.

pages. The Wikipedia page history, sometimes called revision history or edit history, tracks the order in which changes were made to any editable Wikipedia page. We believe that this corpus can help researchers to design approaches for tracking entities usage over time. This resource can be functional to promote new research for dynamic embeddings of named entities. We propose some preliminary case studies for proving the potentiality of this resource.

The paper is structured as follows: Section 2 reviews the state of the art, while Section 3 describes the methodological aspects of our approach. Section 4 shows some use cases of our resource followed by some final remarks.

2 Related Work

The diachronic analysis of language via word embeddings has been an active area of research during the past decade that has generated many resources for several time periods, languages and genres. Kim et al. (2014) used Google Ngram as a diachronic resource to build word embeddings via Word2Vec on a random sample of the 10 million 5-grams from the English fiction portion of the corpus. The authors made the resource available, but due to space limitations, they released the word embeddings only for the 5-year time period. A similar approach was proposed by Grayson et al. (2016), where Word2Vec embeddings are trained on the Eighteenth-Century Collections Online corpus (ECCO-TCP) by taking into account five twenty-year periods for 150 million words randomly sampled from the "Literature and Language" section of the corpus. Hamilton et al. (2016b) also trained word embeddings on the Google Ngram for detecting semantic changes. The authors analysed four different languages, i.e. English, French, German and Chinese, and created a resource which has been successfully used in subsequent studies (Garg et al., 2017; Hamilton et al., 2016a). A different approach to detect the semantic shift of words was adopted by Kulkarni et al. (2015). The authors adopt a change point detection algorithm on the time series generated by computing the cosine similarity between word embeddings trained on several corpora, such as: Twitter, Amazon reviews, and the Google Book Ngrams. A similar approach is proposed in Basile and McGillivray (2018), in which the Temporal Random Indexing (TRI) is adopted for building a distributed, time-based, word representation for the JISC UK Web Domain Dataset (1996-2013) corpus.

Other research efforts have been directed to release resources and applications for the visual analysis and querying of these diachronic collections. The Google Ngram viewer (Michel et al., 2011) was released as a tool for allowing users to query the Google Ngram corpus, a collection of ngram occurrences spanning several years and languages extracted from the Google Book project. Hellrich and Hahn (2017) proposed a system that allows users to explore different corpora via a diachronic semantic search. They used the Corpus of Historical American English, the Deutsches Textarchiv "German Text Archive", and the Royal Society Corpus, in addition to the Google Books Ngram Corpus.

Research directed toward the specific problem of detecting changes in the context surrounding named entities has attracted limited attention compared to the broader area of automatic detection of the semantic shift of words. Some previous work on named entities focused on problems related to searching (Berberich et al., 2009; Kanhabua and Nørvåg, 2010a; Zhang et al., 2016). Tahmasebi et al. (2012) proposed an interesting approach to identify the evolution of named entities. Berberich et al. (2009) defined a method for query reformulations able to paraphrase the user's information need using terminology prevalent in the past. In this work, the original dataset is enriched with annotated phrases extracted from the text by using Wikipedia page titles. In Kanhabua and Nørvåg (2010a), Wikipedia internal links and redirect pages are exploited for finding synonyms across time by using different snapshot of Wikipedia. The identified synonyms are used for query expansion in order to increase the retrieval effectiveness. In some respects, this approach is similar to ours. However, it does not use page revisions and the relation between concepts, surface forms and contexts. Zhang et al. (2016) described an approach to find *past* similar terms closest to a given *present* term. The goal was to improve the retrieval effectiveness in archives and collections of past documents. In this work, Wikipedia is only functional to the creation of the test set, where only the information about the entity lifetime is used (e.g. the time when the name of a country or a company changed). Regarding named entity evolution, Tahmasebi et al. (2012) proposed a method to capture the evolution of one name into another by using a sliding window of co-occurrence terms. The corpus used for the evaluation is the New York Times Annotated Corpus. Lansdall-Welfare et al. (2017) analysed a collection of historical data spanning 150 years of British articles. The authors focus on historical and cultural changes that are tracked via a quantitative analysis of word frequencies. However, they expand their methodology to a "semantic" level by working on named entities extracted from text. The work proposed in Szymanski (2017) is the first attempt to highlight the potential of diachronic word embeddings for solving analogy tasks involving entities and relationships, although this work does not seek to capture named entities in an explicit way. Moreover, Caputo et al. (2018) applied a method to recognise and linking named entities in the whole New York Times corpus. The Temporal Random Indexing is then applied on the annotated corpus in order to build a semantic vector representation for entities and tracking significant shift in their contexts. An explicit representation of named entities is also provided in (Bianchi et al., 2018) where the authors tackle the problem of incorporating time in the Knowledge Graph embeddings in order to provide a similarity measure between entities that accounts for temporal factors.

3 Methodology

The revision history associated with each Wikipedia page opens the way to different diachronic analyses of the highly interconnected concepts represented by its pages. In Wikipedia, pages are interconnected by internal links manually created by users that consist of a surface form and a target. The target is another Wikipedia page, and can be regarded as a "conceptual" link created by the user between the surface form and a specific concept (the Wikipedia page). The same surface form can link several entities and the same entity can be linked to several surface forms. Moreover, since a surface form occurs in a specific context, we can define the surface context as a window of n words to the left and to the right of the surface form. Each page has multiple revisions created every time a user edits that page, and each revision page is associated with a timestamp, so that it is possible to track the changes over time of the temporal relation existing between the surface form, the target and context. For example, it is possible to track the change over time of different surface forms linking to a specific target or to detect the change in the target context. All these capabilities open several possibilities to the analysis of entities using a diachronic perspective.

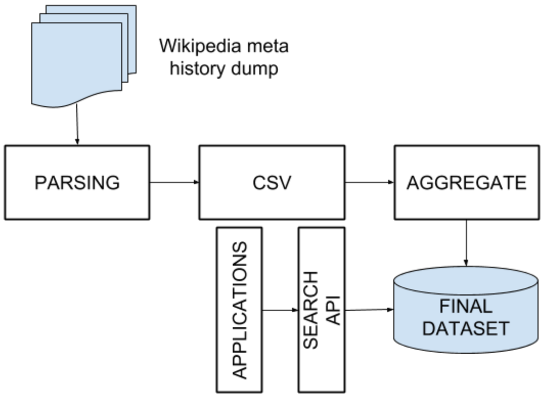


Figure 1: Flowchart of the dataset creation.

Figure 1 depicts the process followed for the creation of our resource. The starting point is the Wikipedia meta history dump which includes all the page revisions in XML format. The dump is composed of several XML files containing the page revisions in Mediawiki syntax. Each XML file is parsed using the DKPro-JWPL API[1], which is able to produce the accurate Abstract Syntax Tree (AST) of each page revision. From the AST, we extract all the internal links that refer to standard[2] Wikipedia pages; each internal link has a surface form and the name of the linked Wikipedia page. In addition, we extract the year from the revision timestamp and the context as the n words around the internal link. The context is processed using the *StandardAnalyzer* provided by the Apache Lucene API[3]. Each extracted internal link is saved in a CSV file as a record consisting of: year, pageId, target, surface form, left context and right context.

An example of a row in a CSV file is reported below:

[1] https://github.com/dkpro/dkpro-jwpl

[2] We remove links to special pages, such as category and user pages.

[3] http://lucene.apache.org/

```
  2003 11057 forge forge forging
term shaping metal use heat
hammer basic smithy contains
sometimes called hearth heating
metals commonly iron steel
malleable temperature
```

The row meaning is that in page *11057* in the year *2003* the target *forge* is linked by the surface form *forge* with the right context *forging, term, ...* and the left context *sometimes, called, ...*.

Since the tuple <*year, surface form, target*> can occur multiple times, we aggregate multiple tuple occurrences in a single record. The aggregation step is performed several times, one time for each dump file plus a final step that aggregates all the records in a single file that represents our final dataset.

In the final file, information is stored as follows:

- A row starting with the sequence *#T* <*TAB*>T_i which identifies the beginning of a sequence of rows in the file that are related to the page (concept) T_i (until a new row starting with *#T* is encountered). T_i represents the Wikipedia page title;
- A sequence of rows containing several values separated by the *tabular* character in the form: year y_k, surface form s_j, the number of time that the surface s_j is used for linking T_i in the year y_k. Then, we build a Bag-of-Word (BoW) from the words occurring in the context, and in the same row we provide the BoW size followed by all the words in the BoW represented as a sequence of pairs <*word, occurrences*>.

A row in the aggregate format is shown in the following example:

```
  #T Apple Computer
2018 Apple Computer 2 30 freedos
1 x 1 supports 1 support 3
officially 1 10 1 s 1 programming
1 9 1 scsi 1 bda 1 2005 1 usb
2 mac 3 announced 2 storage 2
august 1 31 1 ray 2 advanced 1 os
3 its 2 interface 2 blu 2 joined
1 aspi 1 march 1 8.5.1 1 disc 2
mass 2
2018 Apple 1 21 developed 1
computer 1 years 1 independently
1 group 1 computer's 1 1987
1 he 1 while 1 advanced 1
henson 1 associates 1 eric 1
tracking 1 facial 1 technology
1 collaborated 1 six 1 starting 1
worked 1 animation 1
```

The aggregated format shows that the surface form *Apple Computer* was used twice for linking the target *Apple Computer*, while the surface form *Apple* was used only once. The BoW follows each surface form. In the first aggregation step, an aggregated file is created for each segment of the Wikipedia dump, then in the second aggregation step, all the segments are merged in the final dataset.

In this first version of the dataset we do not take into account disambiguation pages and redirects. Managing redirects is a very challenging problem since they are not consistent over dumps.

Relying on this final dataset, we built a search API for easily retrieving all the information related to the target, the surface form and the context according to a specific time period[4].

We exploit the meta history dump dated 1st February 2019; the first Wikipedia pages are dated 2001. The original dump size is about 950GB, the total size of the CSV files is about 30GB, while the final dataset obtained by aggregating data from the CSVs is about 47GB. We set to 10 the dimension of the context window. Since a page can have multiple revisions in the same year, in building our resource we consider only the latest one for each year. It is possible to perform a more fine-grained analysis by taking into account more revisions per year (e.g. a revision for each month). The total number of distinct targets is about 31M, which is larger than the effective number of Wikipedia pages for several reasons: 1) some targets are a redirect to other targets; 2) some pages have been removed or renamed over the years; 3) some targets are a link to a specific section of a page. In this release, we do not take into account these issues, which we plan to tackle in a future release.

The search API can be used for building several applications, such as a RESTful Web Services for remotely querying the data, data analysis for discovering when named entities or surface forms change their usage, and data visualisation.

It is important to underline that the proposed approach is completely unsupervised and language independent since it does not require any NLP pre-

[4] The dataset and the source code are available here https://github.com/pippokill/dae

processing step. Moreover, the proposed methodology is intrinsically multi-language because it is possible to rely on the specific Wikipedia dump of the language under analysis. In addition, it is possible to exploit multi-language Wikipedia links for comparing the evolution of named entities across different languages.

One limit of our approach is the short time frame taken into account since Wikipedia was launched in 2001. However, our approach is incremental and the dataset can grow when new Wikipedia dumps are available. Moreover, the dataset is not only useful for diachronic analysis of entities, but the detection of semantic changes over a short period of time can be exploited to improve the performance of several algorithms, such as entity linking, relation extraction and ontology/knowledge graph population.

4 Use cases

In this section we report some use cases that have emerged from an exploratory analysis of the proposed dataset. We perform the analysis by indexing the 1M most frequent targets extracted from the final dataset. We build an API for querying the dataset by using the Apache Lucene library. Each following subsection reports details about a specific use case.

4.1 Concepts linked by a surface form

The first use case concerns the analysis of the concepts linked by a surface form over time. Table 1 shows the concepts linked by the surface form "Donald Trump". While before 2015 there is only one concept linked by this surface form, since 2016, the concepts related to the presidential campaign have emerged, with the concept "Presidency of Donald Trump" occurring in the first top-5 concepts since 2018. It is important to underline that the first column (2015) reports only one concept since no other concepts are related to the surface form "Donald Trump" in the 2015. This is due to the fact that in this preliminary study we limited our analysis to the 1M most frequent targets and not the whole set of 33M targets. A reverse analysis shows the usage of the surface form "President Trump" to refer to the concept "Donald Trump" since 2017.

4.2 Contexts of a given target

Another interesting analysis concerns the change over time of the contexts of a given target. In this case, it is possible to compute the displacement over time of the target concept by computing the cosine similarity between the context BoWs. For each pair of years, we build a BoW vector for the context of the target concept. Then, we generate a time series by computing the cosine similarity between the BoW of two consecutive years (BoW_{y_i} and $BoW_{y_{i+1}}$). Figure 2 reports the time series generated for the concept "Donald Trump"; we observe a change point corresponding to a drop in similarity between 2015 and 2016.

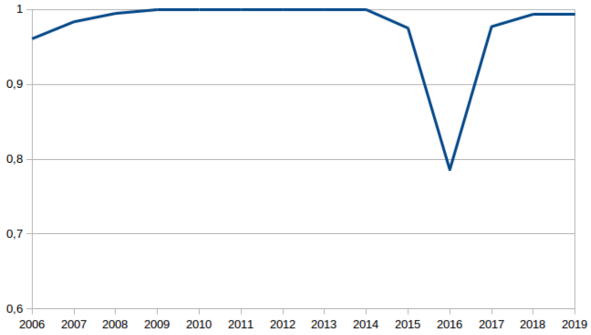


Figure 2: BoW cosine similarity time series for the concept "Donald Trump".

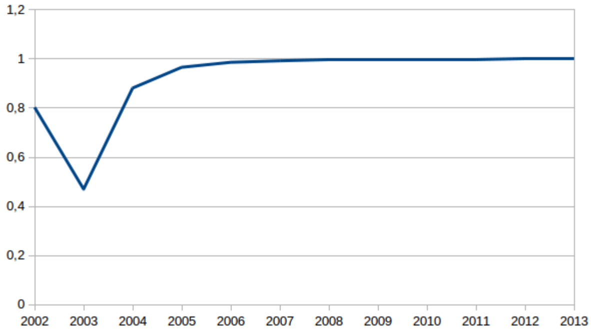


Figure 3: BoW cosine similarity time series for the concept "Arnold Schwarzenegger".

A similar analysis performed for the concept "Arnold Schwarzenegger" shows a change point between 2002-2003 and 2003-2004, as reported in Figure 3. Through the analysis of the most frequent words in the BoWs of the contexts of "Arnold Schwarzenegger" in the period 2002-2004, it emerged that while the most frequent words in 2002 were *film, actor, movie, terminator*, in 2003 new words such as *governor* and *California* related to "Arnold Schwarzenegger" political

2015	2016	2017	2018	2019
Donald Trump	Donald Trump presidential campaign, 2016	Donald Trump	Donald Trump	Donald Trump
	Protests against Donald Trump	Protests against Donald Trump	Protests against Donald Trump	Donald Trump presidential campaign, 2016
	Donald Trump sexual misconduct allegations	Donald Trump presidential campaign, 2016	Donald Trump presidential campaign, 2016	Protests against Donald Trump
	Political positions of Donald Trump	Donald Trump sexual misconduct allegations	Donald Trump sexual misconduct allegations	Donald Trump sexual misconduct allegations
	Stop Trump movement	Donald Trump (Last Week Tonight)	Presidency of Donald Trump	Presidency of Donald Trump

Table 1: Top-5 concepts linked by the surface form "Donald Trump".

activity have started to appear, to become the most frequent words in the BoWs since 2004.

Another interesting use case is the analysis of the BoWs of the targets linked by the same surface form. This analysis may highlight changes in the way common words are used for referring to named entities. For example, analysing the usage of the surface form "tweet", we observe that since 2012 it has been used to refer to the concept "Twitter", while before 2012 it did not refer to any concept.

4.3 Similarity between two entities over time

The last scenario shows the possibility to compute the similarity between two entities over time as the cosine similarity between the target contexts. Figure 4 reports the time series of similarities between three pairs of entities (Apple-Microsoft, Apple-IBM, IBM-Microsoft). It is interesting to observe that the similarity between IBM and Microsoft is higher then the similarity between Apple and the other two entities, although Apple is equally related to both of them.

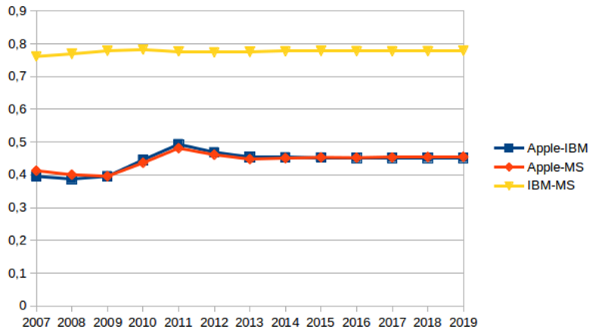


Figure 4: Comparison between pair of entities.

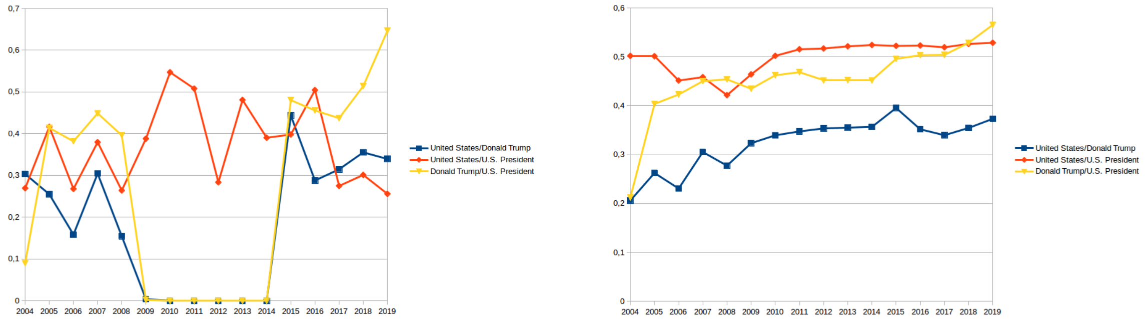
Finally, the plots in Figure 5 show the cosine similarity between the BoWs of two different targets (concepts). Using this approach it is possible to show how the similarity between two targets changes over time. In particular, for each time point we build the BoW of each concept and then we compute the similarity between the BoWs. It is important to point out that the target BoW is built by taking into account the context around each occurrence of the target in the corpus. In this way, if two targets occur in similar contexts their BoWs will be similar. We adopt two strategies:

point-wise: each BoW is built by taking into account only the target occurrences at time t_i;

cumulative: each BoW is built by taking into account all the target occurrences up to time t_i, including time t_i. The idea is to take into account all the previous history of the target and not only the time period under analysis.

Observing the plots in Figure 5, it is possible to note that the similarity between *United States-U.S. President* and *United States-Donald Trump* is constant across time, while we observe an increment in similarity between *U.S. President-Donald Trump* after the year 2018. This increment is clearly evident in the point-wise analysis (Figure 5a), as we expected. It is important to underline that in Figure 5a some points are near zero (2009-2014) this means that the targets do not occur in similar contexts in that periods and indeed the two BoWs share just a few words. Figure 5b show a different trend, since we take into account all the previous target occurrences before the time t_i by exploiting the cumulative approach.

The promising results obtained in this preliminary case study about BoW similarity suggest that it is possible to build effective "time-dependent" embeddings by using our resource.



(a) Plot of the point-wise targets BoW cosine similarity over time.

(b) Plot of the cumulative targets BoW cosine similarity over time.

Figure 5: BoW analysis of pair of targets: plot over time of the cosine similarity between BoWs of two targets with point-wise (a) and cumulative (b) strategy.

5 Conclusions and Future Work

In this paper, we described the construction and utilisation of a new resource built upon Wikipedia page revisions that enables the diachronic analysis of entities. Using the timestamp provided by each revision, we tracked Wikipedia internal links in order to extract the temporal relations between surface forms, contexts, and concepts (Wikipedia pages). We provided some preliminary use cases which show the effectiveness of this resource. These preliminary results show the potentiality of our methodology and open several research scenarios that can be investigated as future work, such as semantic change point detection of entities, entity linking in diachronic collections, event detection, and temporal entity search. The preliminary version of our dataset has some issues that we plan to fix in future versions such as redirects, disambiguation pages and character encoding issues.

Acknowledgement

This work is partially funded by the "TALIA - Territorial Appropriation of Leading-edge Innovation Action" project, Interreg-Mediterranean program for increasing transnational activity of innovative clusters and networks of key sectors of the MED area (2018-2019) and by the ADAPT Centre for Digital Content Technology, funded under the SFI - Science Foundation Ireland Research Centres Programme (Grant SFI 13/RC/2106) and is co-funded under the European Regional Development Fund and by the EU2020 - European Unions Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No.: EU2020-713567.

References

Pierpaolo Basile and Barbara McGillivray. 2018. *Discovery Science*, Springer-Verlag, volume 11198 of *Lecture Notes in Computer Science*, chapter Exploiting the Web for Semantic Change Detection.

Klaus Berberich, Srikanta J Bedathur, Mauro Sozio, and Gerhard Weikum. 2009. Bridging the terminology gap in web archive search. In *WebDB*.

Federico Bianchi, Matteo Palmonari, and Debora Nozza. 2018. Towards encoding time in text-based entity embeddings. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018*. Springer International Publishing, pages 56–71.

Annalina Caputo, Gary Munnelly, and Séamus Lawless. 2018. Temporal entity random indexing. In Jonathan Girón Palau and Isabel Galina Russell, editors, *Digital Humanities 2018, DH 2018, Book of Abstracts, El Colegio de México, UNAM, and RedHD, Mexico City, Mexico, June 26-29, 2018*. Red de Humanidades Digitales A. C., pages 460–461. https://dh2018.adho.org/en/temporal-entity-random-indexing/.

Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. 2017. Word Embeddings Quantify 100 Years of Gender and Ethnic Stereotypes. *PNAS* 115. http://arxiv.org/abs/1711.08412.

Mihai Georgescu, Nattiya Kanhabua, Daniel Krause, Wolfgang Nejdl, and Stefan Siersdorfer. 2013. Extracting event-related information from article updates in wikipedia. In Pavel Serdyukov, Pavel Braslavski, Sergei O. Kuznetsov, Jaap Kamps, Stefan Rüger, Eugene Agichtein, Ilya Segalovich, and Emine Yilmaz, editors, *Advances in Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 254–266.

Siobhán Grayson, Maria Mulvany, Karen Wade, Gerardine Meaney, and Derek Greene. 2016. Novel2Vec

: Characterising 19th Century Fiction via Word Embeddings. In *24th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'16), University College Dublin, Dublin, Ireland, 20-21 September 2016*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Cultural shift or linguistic drift? comparing two computational measures of semantic change. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2116–2121. https://doi.org/10.18653/v1/D16-1229.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. *arXiv preprint arXiv:1605.09096* .

Johannes Hellrich and Udo Hahn. 2017. Exploring Diachronic Lexical Semantics with J E S EM E. In *Proceedings of ACL 2017, System Demonstrations*. pages 31–36. http://aclweb.org/anthology/P17-4006.

Nattiya Kanhabua and Kjetil Nørvåg. 2010a. Exploiting time-based synonyms in searching document archives. In *Proceedings of the 10th annual joint conference on Digital libraries*. ACM, pages 79–88.

Nattiya Kanhabua and Kjetil Nørvåg. 2010b. Quest: Query expansion using synonyms over time. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 595–598.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal Analysis of Language through Neural Language Models. *Arxiv* pages 61–65. http://arxiv.org/abs/1405.3515.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, pages 625–635.

Thomas Lansdall-Welfare, Saatviga Sudhahar, James Thompson, Justin Lewis, FindMyPast Newspaper Team, and Nello Cristianini. 2017. Content analysis of 150 years of British periodicals. *Proceedings of the National Academy of Sciences* 114(4):E457–E465.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *science* 331(6014):176–182.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 448–453. https://doi.org/10.18653/v1/P17-2071.

Nina Tahmasebi, Gerhard Gossen, Nattiya Kanhabua, Helge Holzmann, and Thomas Risse. 2012. Neer: An unsupervised method for named entity evolution recognition. *Proceedings of COLING 2012* pages 2553–2568.

Yating Zhang, Adam Jatowt, Sourav S Bhowmick, and Katsumi Tanaka. 2016. The past is not a foreign country: Detecting semantically similar terms across time. *IEEE Transactions on Knowledge and Data Engineering* 28(10):2793–2807.

Using a Lexical Semantic Network for the Ontology Building

Nadia Bebeshina-Clairet
LIRMM / LIMICS

Sylvie Despres
LIMICS
74 rue Marcel Cachin
93017 Bobigny

Mathieu Lafourcade
LIRMM
860 rue de St Priest
34095 Montpellier

Abstract

Building multilingual ontologies is a hard task as ontologies are often data-rich resources. We introduce an approach which allows exploiting structured lexical semantic knowledge for the ontology building. Given a multilingual lexical semantic (non ontological) resource and an ontology model, it allows mining relevant semantic knowledge and make the ontology building and enhancement process faster.

1 Introduction

Nowadays, termino-ontological resources are increasingly rich in terms of data they rely upon. The scientific community works intensively on data acquisition for the ontology building. In particular, the NeOn project[1] has been set up to provide a methodology for the ontology engineering by integrating preexisting knowledge resources into an ontology building process. The NeOn methodology contains a consistent framework for modular ontology building as well as for setting up ontology networks. Here we focus on exploiting Lexical Semantic Networks (LSNs) to enrich an ontology or accompany the ontology building process. We assume that LSNs represent knowledge as it is expressed through the human language whereas ontologies provide a formal description (specification) of a conceptualization (concepts and relationships between those concepts) shared by a community of agents. A concept corresponds to a set of individuals sharing similar characteristics and may or may not be lexicalized. Thus, the ontology labels cannot be polysemous. The strength of ontologies is in their formal consistency. The weaknesses are linked to their coverage, size (as stated in (Raad and Cruz, 2015), ”large ontologies usually cause serious scalability problems”), and human effort needed for their building. The potential of the LSNs is linked to the large amount of explicit semantic information they contain. However, a filtering process is needed to discriminate irrelevant information (polysemy, noise).

2 State of the Art

The opportunity of ontology construction empowered by the use of Natural Language Processing (NLP) techniques and tools has been explored for more than 20 years. Among the achievements, one can distinguish the tools which take into account the difference between the lexical term and the ontology concept (differentiated tools) and those that do not make such distinction. Differentiated tools and methods suggest extracting the terminological units from texts and organizing them as a network using a set of hierarchical and equivalence relation types. Such network guides the ontology expert through the conceptualisation and ontology building process. Such process relies on an intermediary structure, a termino-ontology (for instance, the *Terminae* suite, (Szulman, 2012)). Undifferentiated tools use some statistical information to suggest the can-

[1] http://neon-project.org/nw/About_NeOn.html

didate concepts. They exploit such methods as formal concept analysis (Mondary, 2011) or knowledge based methods (for instance, *TextToOnto*[2]). ”Lexical ontologies” (Abu Helou et al., 2014) are successfully used for ontology building. Numerous approaches targeted at high level ontology or information retrieval ontology based on general knowledge (such as (Marciniak, 2013)) rely on PWN. Others use PWN and domain specific semantic lexicons for forming the concepts (Turcato et al., 2000). Many other ontology learning techniques use distributional semantics to learn lightweight ontologies, for example, (Wong, 2009).

In the framework of corpora based approaches to the ontology building such as described in (Kietz et al., 2000), the idea of *notable (salient, relevant) element* or *relevant piece of knowledge* (RPK) has been introduced. It corresponds either to the frequent terms appearing in a corpus and to the tacit knowledge contained in texts. Such tacit knowledge corresponds to the semantic relationships (subsomption relationship and other specialized relationships). Their presence in texts may take the form of ”indices”. In contrast, the explicit elements may reveal the presence of concepts. The main drawback such definition of RPKs is that it relies on the features defined or recorded for a particular language. In addition, statistical criteria are often preferred and it is difficult to qualify such RPKs from the semantic point of view and in a language independent manner. In this paper, we will detail the experiments we conducted to provide a new definition of the RPKs based on the structured lexical semantic information and describe the way so defined RPKs can be used to help the top down ontology building process.

3 Ressources

In the present section, we will describe the resources we used in our experiments.

The **RezoJDM** (Lafourcade, 2007) is a lexical semantic network (LSN) for French built using crowdsourcing methods and, in particular, games with a purpose (GWAPS) such as JeuxDeMots[3] and additional games . This commons sense network has been built since 2007. It is a directed, typed and weighted graph. At the time of our writing, RezoJDM contains 2.7 millions of terms that are modeled as nodes of the graph and 240 millions of relations (arcs).

The **MLSN** (Bebeshina-Clairet, 2019) is a multilingual LSN (it covers French, English, Spanish, and Russian) with an interlingual pivot built for the cuisine and nutrition domain. This network is inspired by the RezoJDM in terms of its model. Structurally speaking, the MLSN is a directed, typed, and valuated graph. It contains k sub-graphs corresponding to each of the k languages it covers and a specific sub-graph which fulfills the role of the *interlingual pivot*. Similar to the RezoJDM, we call *terms* the nodes of the MLSN and *relations* - its typed, weighed, and directed arcs. The MLSN nodes may correspond to one of the following types : **lexical items** (*garlic*), **interlingual items** (pertaining to the interlingual pivot and also called *covering terms*), **relational items** (i.e. relationship reifications such as *salad[r_has_part]garlic*), and **category items** parts of speech or other morpho-syntactic features (i.e. *Noun:AccusativeCase*).

As it has been difficult to set up the pivot using a multilingual embedding (joining multiple spaces, one per language) as well as to avoid pairwise alignment based on combinatorial criteria, the pivot has been started as a natural one using the English edition of DBNary (Sérasset, 2014). It incrementally evolves to become interlingual. The pivot evolution relies on sense-based alignments between the languages of the MLSN and aims at taking into account the difference of sense ”granularity” in different languages. For example, as *stew*in English can be translated as into French as *pot-au-feu* and *ragoût*. It reflects the conceptualization discrepancy as *ragoût* refers to sauté the ingredients and then add water

[2] https://sourceforge.net/p/texttoonto/wiki/

[3] http://www.jeuxdemots.org

whereas *pot-au-feu* refers to boiling them together in a larger amount of water than used for the *ragoût*. In the MLSN pivot we have the interlingual term corresponding to stew (which covers the English term) with two hyponyms corresponding to 0rench terms. The alignments are progressively obtained through external resources or by inference. Thus it can be considered as a union of word senses lexicalized or identified in the languages covered by the MLSN. . Even though we assume the pivot as being interlingual, it is still close to a natural one. A relation $r \in R$ is a sextuplet $r =< s, t, type, v, l_s, l_t >$ where s and t correspond respectively to the source and the target term of the relation. The relation $type$ is a typical relation type. It may model different features such as taxonomic and part-whole relations (*r_isa, r_hypo, r_has_part, r_matter, r_holo*), possible predicate-argument relations (typical object *r_object*, location *r_location*, instrument *r_instr* of an action), "modifier" relations (typical characteristic *r_carac*, typical manner *r_manner*) and more[4]. The relationship valuation v corresponds to the characteristics of the relation which are its weight, confidence score, and annotation. The relation weight may be negative in order to model noise and keep the information about erroneous relations easy to access programmatically so they could not affect the inference processes. The confidence score is a score attributed to a particular data origin (external resource, inference process). In practice, this feature is an array as different origins may provide the same relation. The confidence information is provided as an argument to the function that maps from some external knowledge resource to the MLSN. In case of relation calculated by an inference process, it corresponds to the precision evaluated on a sample of candidate relations returned by this process. To *annotate* a relation we add a complementary information that allows qualifying this relation. The figure 1 details and exemplifies the annotation scheme.

The labels l_s and l_t correspond to the language (subgraph) labels. At the time of our writing, the MLSN contains 821 781 nodes and 2 231 197 arcs. It covers 4 languages : English, French, Russian, and Spanish.

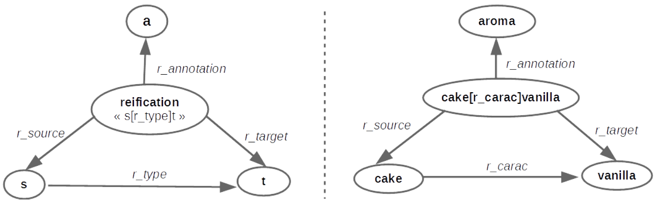


Figure 1: MLSN: relationship annotation scheme.

MIAM (Desprès, 2016)[5] is a modular termino-ontology for the digital cooking. It provides knowledge necessary for the elaboration of general nutritional suggestions. The knowledge model of this ontology gathers expert knowledge on food, food transformation, cooking actions, relevant dishes that reflect french culinary tradition, recipes necessary to cook such dishes. MIAM contains about 7 000 nodes and 30 000 semantic (non subsumption/ontological is-a) relations.

4 Method: Immersion - Projection

4.1 Summing up the Method

Our method is built upon the idea of projecting a model (the MIAM model) onto a multilingual or monolingual LSN (respectively MLSN and Rezo-JDM) in order to extract an intermediary resource that can be used by ontology or domain experts in the scope of information retrieval or validation of the automatically suggested pieces of knowledge.

Our method differs from others by the definition of the RPK and by the use of a non ontological semantically structured resource for ontology building. We define RPK as follows : "a relevant piece of knowledge is either a term or a relation or a semantic structure which is known as qualified and qualifying". *Qualified* refers to the possibility to describe

[4] We also introduced more specific relation types such as *r_entailment*, *r_cause*, *r_telic_role*, *r_incompatible*, *r_before*, *r_after* etc.

[5] http://www-limics.smbh.univ-paris13.fr/ontoMIAM/

the RPK in a discrete way (i.e. by enumerating the typed relations). If the RPK is a term, it needs to have a high in-degree (which reveals its conceptual role as it is used to define other terms of the network). If the RPK is a relation, it needs to be contextualized (through the annotation mechanism represented on the figure 1 or through the constraints put on source and/or target terms of the relation). If the presumed RPM is a graph structure (path, subgraph), it needs to possess a certain number of occurrences in the network. *Qualifying* refers to the possibility to use the candidate RPK for endogenous inference process. If the RPK is a term, it needs to have hypernyms, hyponyms, synonyms among its neighbours. It has to be aligned with other terms pertaining to the other languages of the LSN (if such LSN is multiligual). If the RPK is a relation, it must not be unique (other real or potential[6] relations must exist in the network). If the candidate RPK is a structure, its terms and relations must be qualifying.

Here we detail the experiments that have been conducted on the basis of the MLSN in order to propose "pseudo-class" and "pseudo-property" candidate RPKs to enhance the MIAM ontology and those concerning the enrichment of an ontology draft using the monolingual LSN, RezoJDM (Lafourcade, 2007). These experiments rely on lexical knowledge. Therefore, the resulting RPKs have no pretension to the ontological validity. The decision pertains to the human expert.

4.2 Immersion

The projection of a given ontology model onto a LSN starts by the immersion of such model. The immersion mechanism uses a set of manually defined mapping rules. It is possible to generate them automatically for the ontological resources that exploit standard vocabularies (such as RDFS, SKOS and other machine readable formats). The input of the immersion algorithm is the reference ontology (MIAM) and the set of mapping rules whereas its output is the action of inferring terms and relations in the target LSN (MLSN, RezoJDM).

In their general form, the mapping rules state: "If x and y are respectively domain and range of an Object Property p of the ontology to be immersed and y is a subclass of C, then x has a relation R with y and y has a relation *is-a* with C in the receiving (target) LSN". Such rules have been defined for the multilingual experiment for two reasons. First, for each of the 93 MIAM properties, we determined relevant MLSN semantic relation types (or set of types). Thus, the *Object Property* `aPourProduitInitial` (*hasInitialProduct*) corresponds to the substance and part-whole meronymy (MLSN relations typed *r_has_part* and *r_matter*). Second, we mapped the ontology labels to the MLSN terms by coincidence (3 930 terms; i.e. *poulet basquaise* formally denotes a MIAM concept, a lexical item with the same label already exists in the MLSN) or by composition (4 135 terms, i.e.: *unité mesure capacité* doesn't correspond to any existing MLSN term because it doesn't correspond to any commonly used collocation in French; this label is split and integrated into MLSN with the semantic relations that link its parts.

As part of the monolingual experiment, 115 descriptors have been automatically expressed in French on the basis of their Uniform Resource Identifier (URI) strings. All the terms except one were already present in the RezoJDM network. We exploited the relations typed *r_carac* (typical characteristic) for this experiment. This relation has been annotated using the URIs of the ontology properties `aPourDescripteurBruit` (*hasSoundDescriptor* is immersed as follows: *croûte r_carac* :: *bruit croustillant* ("*croûte* has typical characteristic linked to the noise *croustillant*"[7]). The premises of the mapping rules rely on the contextualization of the LSN relations. Such contextualization is possible when using sets of hypernyms and neighborhood semantic relations of the source and target terms of a

[6] Relations that can be calculated using inference.

[7] *crust* has typical characteristic linked to the sound *crusty*

relation. Meta-information attached to the LSN relations (annotations, weight) may also be used. For instance: *pétrir r_object pâte* ∧ *pétrir r_isa technique de base* ∧ *pâte r_isa préparation*[8]).

As part of the immersion process the ontology labels become LSN terms that can be polysemous.

4.3 Projection

4.3.1 Inference in the LSN Context

In the MLSN context we set up several algorithms to discover relevant pieces of knowledge (RPK) of the types "class/individual" (ci) and "ontology property"(op). To discover the RPK(ci) we compare the neighborhood terms inside an hierarchical chain which goes up to a high level MIAM concept immersed into the LSN. For the RPK(op) we look for (real or possible) MLSN relations similar to the immersed MIAM properties. The inference scheme we use is the abduction scheme. When we have two similar terms (such as cohyponyms) the relations detained by one of them can be proposed for the other. For a term *T*, the abduction implies selecting a set of similar terms (according to some criteria) in order to propose the relations detained by those similar terms to T.

4.3.2 Discovering the RPK(ci)

In MIAM, the **general axioms** concern the disjunction between the MIAM classes which is the basis of the ontology consistence. To translate them in terms of MLSN, we considered the labels of the classes listed in the axioms in order to identify the criteria that could have determined the disjunction. We manually analyzed a subset of the MIAM axioms and came up with the following criteria: *affiliation* (*r_has_part* i.e. a specific label (organic)), *transformation* (*r_carac* i.e. boiled mixture, cubed vegetable), *composition* (*r_matter* i.e. *produit à base de poisson* "fish based product"), *category based distinctions* (*r_hypo* i.e. *volaille type dinde* "turkey type poultry"). This analysis allowed selecting a subset of relation types to consider during the experiment. The RPK(ci) inference includes two steps: validation of the hierarchical chain (figure 2) and RPK(ci) candidate suggestion.

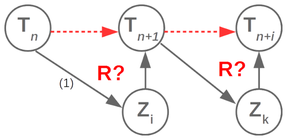


Figure 2: Hierarchy chain validation. T_i are the terms of the hierarchy chain. We check by triangulation their semantic relatedness and use a subset of relations types (such relations are noted R) for that.

We calculated and validated hierarchical chains corresponding to 1 322 top MIAM concepts pertaining to the *Aliment* module. First we obtained 132 213 chains. After filtering them by weight, the set has been reduced to 53 749 chains (40% of the initial set). Still, a certain number of redundancies may exist inside this statistically pre-filtered set since a long chain may include several shorter ones. The logical filtering by triangulation left us with a set containing 9 600 hierarchical chains (18% of the number of statistically filtered chains and 7% of the initial number of candidates).

Hierarchical chain examples after filtering:
(1) "***baguette complète***→*pain complet* →
pain→***ingrédient de recette de cuisine***→*aliment*"
(2) "***angélique***→confiserie→bonbon".
RPK(ci) examples:
truffe chocolat `subClassOf` chocolat
pomme à cidre `subClassOf` pomme
sucre de pomme `subClassOf` confiserie
The analysis and validation of the hierarchical chains corresponds to the important memory load. The complexity of the algorithm depends on the importance of the MIAM concept being processed as well as on the length of the hierarchical chains that are considered. The use of a subset of the semantic relations types available in the MLSN reduces the number of combinations to process. Thus, given

[8] *knead r_object dough* ∧ *knead r_isa basic technique* ∧ *dough r_isa mixture*

that the highest in-degree typed *r_isa* in the French sub-graph of the MLSN is 5 264 (for the term *aliment*) and that the maximum length $l = 9$, the complexity in the worst case would be $O(d_{isa}^{l}$ or $O(5\,264^{9}) = 3,103436942 \times 10^{33}$.

The table 1 introduces the results obtained for the discovery of RPK(ci) related to the top level concept *Aliment* in the French sub-graph of the MLSN. The automatic evaluation of the proposed RPK(ci) would mostly rely on similarity measures. However, the projection step implicitly relies on relatedness and similarity between the LSN terms. Thus, in our future work, the RPK(ci) evaluation will need human expert decisions.

#candidates	#valid	%valid	#new	%new
11 520	11 289	98%	4 741	42%

Table 1: The RPK(ci) discovery.

4.3.3 The RPK(op) Discovery

The RPK(op) discovery seems to be particularly useful in the context of multilingual ontology building or localization of an existing ontology. In our experience, each module of the MIAM ontology has its own hierarchy of properties. While immersing them into the MLSN, these properties have been expressed in terms of semantic relations contextualized using annotations. The choice of the MLSN semantic relation type made for these properties allows us to distinguish the following cases for the MIAM Object Properties (OPs): *composition based* (`aPourProduitInitial`, ”hasInitialProduct”); *related to processes* (`aPourMethodeDeConservation`, ”hasConservationMethod”); *temporal and spatial relation based* (`aPourMoisPrimeur`, ”hasEarlyMonth”); *characteristic based* (`aPourEtat`, ”hasState”); *OPs with a specific sub-graph*[9] (`aPourAlimentAmi`, ”hasFriendlyFoodItem”).

The MIAM ontology we try to enrich counts 21 565 instances of *Object Properties*. Once they are immersed into the MLSN, one could consider that we have the same number of inference rule instances that can be used for the cross-lingual RPK(op) discovery. A naive approach would be setting up a cross-lingual inference mechanism. However, such approach would be error-prone due to the potential alignment and polysemy issues. In addition, as MIAM has been built according to the top-down methodology by a community of domain experts, it contains a variable number of instances per property. The naive approach would reiterate this imbalance.

To refine the RPK(op) discovery, we experimented a rule based approach. First, the validity of the rule for the source language is calculated. Second, structures similar to those specified by the rule are being discovered in the MLSN (in other languages).

The rule has the following form:
`property=aPourEtatPhysique`
(property name)
`src=?s` *(domain, set of terms)*
`reltype=r_carac` *(relation type)*
`tgt=?o`*(range, set of terms)*
`source_isa=aliment`*(src hypernyms)*
`target_isa=etat physique`
(tgt hypernyms)
`annotation=int:physical state`
(meta-information)
`src_feat=OUT/r_pos/int:Noun`
(in and out relations that characterize terms in the source set)
`tgt_feat=OUT/r_pos/int:Adj` *(in and out relations that characterize terms in the target set)*
If a given rule allowed detecting enough structures in the source language (at least, 2 structures), it is considered as a valid one and can generate the qualifying object. Thanks to this object, candidate structures are detected in the other language sub-graphs of the MLSN. The mechanism of RPK(op) discovery reveals the following elements that allowed discovering new pieces of knowledge : **possibly annotated semantic relation** (case of the properties such

[9] A subset of MLSN terms connected through semantic relations.

as `aPresenceLactose`, `aPresenceGluten`); **specific pattern** (defined by rules); **complex structure** for properties related to processes. The results obtained using possibly annotated relations(in particular, *Data Properties*)) are presented in the table 2. the potential improvement is estimated as an increase compared to the initial number of property instances (impr.%).

#DP	#MIAM	#RPK(op)	filt.	+%
aTeneurLipide	0	4 741	3 271	-
aPresenceLactose	2 593	530	408	+16%
aPresenceGluten	289	820	762	+263%

Table 2: The RPK(op) discovery on the basis of simple semantic relations.

Example of the output of the rule-based algorithm: *ru:jarkoje* aPourProduitDiscriminant *podlivka* ("stew hasDiscriminatingProduct sauce") and *en:stew* aPourProduitInitial *en:vegetable (produce)* ("stew hasInitialProduct vegetable"). Our rule-based RPK(op) experiment (given the actual state of the MLSN) yielded the results listed in the table 3.

Fully automated structure-based evaluation such as described in (Fernández et al., 2009) may be chosen to compare to other resources available on the Web such as (Dooley et al., 2018). To address the ontology accuracy, completeness, conciseness, efficiency, consistency, and other features (Raad and Cruz, 2015), a combination of methods is needed. In particular, gold standard ontology, specific tasks and corpora may be used for evaluation. A task-based evaluation such as semantic analysis (Bebeshina-Clairet, 2019), dietary conflict detection from dish titles (Clairet, 2017) have been used for the MLSN. To evaluate the output of the immersion- projection method, we need to organize our triples into a fully structured ontology. This will be one of the priorities of our future work.

4.3.4 Towards the Automatic Suggestion of the RPKs(op)

To extend the RPK discovery experiment, we tried to automatically suggest pseudo ontology properties to be submitted to the domain and ontology human experts. We considered the ontology SensoMIAM[10] for this experiment. This ontology is a MIAM module but we considered it as a "draft" ontology as the sensory aspect modeling is a flourishing research and development area and the SensoMIAM could be improved. We used the monolingual LSN (RezoJDM). The SensoMIAM contains sensory descriptors such as `DescripteurTact` ("TactileDescriptor") = {*astringent, filandreux, ..., nerveux*}; `DescripteurSubstance` ("SubstanceDescriptor")= {*aéré, dense, . . . , épais*}

To calculate the RPK descriptors RPK(desc), we explored the semantics of the source terms of the relations typed *r_carac*. If the set of outgoing relations of such terms connects them to a food item and if they have a set of typical characteristics shared with other terms with an hypernym ≈ "food", the target term of their outgoing relations typed r_carac that is not present in the SensoMIAM can be suggested as a potential RPK(desc). The relation typed *r_carac* is annotated. The process is represented on the figure 3. The experience allowed to suggest the RPK(desc) such as: `DescripteurArome`={*sucré-salé, miellé, . . . , vinaigré*} or `DescripteurTact` = {*écailleux, spongieux,. . . , floconneux*}.

We automatically suggested and semi-automatically validated 342 RPKs(desc). We explored the possibility of suggesting relevant RPKs to human experts. We defined 3 pseudo-properties for testing: *aPourComposantFlaveur* ("hasFlavourComponent"), *aPourComposantToucher* ("hasTouchComponent"), and *aPourComposantAspect* ("hasAspectComponent"). To populate them, we explored the RezoJDM relations typed *r_has_part* and *r_matter* and considered the characteristics that can be shared

[10] `www-limics.smbh.univ-paris13.fr/sensoMIAM/`

m	prop	#in	en	fr	es	ru	*#out*	*%aug*
A	aPourProdInit.	2031	292	1208	203	2 245	3 039	+149%
A	aPourEtatPhys.	543	30	29	10	53	85	+16%
A	aPourForme	39	77	78	5	37	132	+338%
A	aPourLabel	114	15	11	3	1	29	26%
A	aPourMethodC.	115	94	101	13	156	309	+269%
A	aPourMois	116	117	221	23	28	116	+288%
A	aPourRegion	289	98	71	2	57	216	+75%
A	aPourProdCon.	98	256	302	143	103	570	+582%
A	aPourProdInitialA.	41	94	147	12	567	259	+633%
P	aPourTypeDeCuis.	23	155	124	80	285	686	+2 986%
P	aPourDomCul.	82	112	92	120	1313	1276	+1 557%
P	aPourDecoupe	82	82	78	56	77	272	+332%
S	aPourSaveur	752	51	78	47	98	232	+31%
S	aPourDescripteurBr.	119	67	80	10	6	159	+134%
S	aPourCouleur	233	192	451	59	423	911	+391%
S	aPourAspectSurf.	176	40	35	12	52	101	+58%
S	aPourSensationT.	54	84	77	21	12	155	+287%
-	Total	5388	2384	3960	937	4953	9531	+177%

Table 3: Rule-based approach. **m** -name of module (*Aliment* (A), *Preparation* (P), *Sensory* (S)), **prop** - property, **#in** - MIAM triples, **en**, **fr**, **es**, **ru** - MLSN sub-graphs. **#out** - overall number of suggested RPK(op) after filtering, **%aug** - potential improvement.

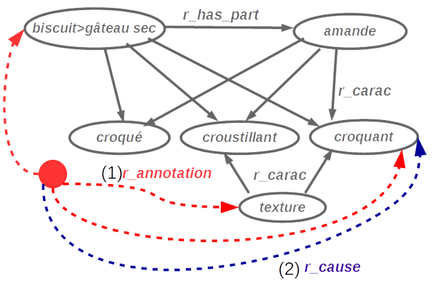


Figure 3: (1) relation annotation. (2) RPK(op) suggestion.

by a ”whole” and its ”parts”. We tried to generalize to the ”whole” some of the characteristics of its parts. Automatically suggested toy triples: *veau Orloff* `aPourComposantFlaveur` *lard* from {*gras, viande*} ”veal Prince Orloff has a component that influences its flavour *lard* because they share *fat, meat*”; *gratin* `aPourComposantAspect` *fromage* from {*gratiné, gras, brûlé*} ”gratin has a component that influences its aspect *cheese* as they share characteristics *grilled, fat, burned*”. We automatically suggested 1 709 RPKs(op) for the pseudo-properties we explored. They have been automatically validated by constraining the range of the pseudo-property (it must be related to the terms ”flavour”, ”touch”, and ”aspect”) and by checking the sufficient intersection size between the relation sets typed *r_carac* of the ”whole” and its ”part”.

Conclusion and Perspectives

We described a method that attaches an intermediary resource containing relevant pieces of knowledge harvested from semantically structured resources in different languages to the main building process. The method suits for other domains of knowledge and the amount of work necessary for the immersion process is proportional to the size and to the type of the ontological resource to be enhanced. It is also possible to use such resources as WordNets [11] as the basis for the intermediary resource. Among the difficulties linked to our method appear the differences between formal representation paradigms as well as the availability of well structured and semantically rich resources.

[11] http://globalwordnet.org/

References

Mamoun Abu Helou, Mustafa Jarrar, Matteo Palmonari, and Ch Fellbaum. 2014. Towards building lexical ontology via cross-language matching. pages 346–354.

Keith Allan. 2001. *Natural Language Semantics*. Blackwell.

Nadia Bebeshina-Clairet. 2019. *Construction d'une ressource termino-ontologique multilingue pour les domaines de la cuisine et de la nutrition*. Theses, Université Paris 13.

Christian Biemann. 2005. Ontology learning from text: A survey of methods. *LDV Forum* 20:75–93.

Nadia Clairet. 2017. Dish classification using knowledge based dietary conflict detection. In *RANLP 2017*.

Sylvie Desprès. 2016. Construction d'une ontologie modulaire. application au domaine de la cuisine numérique. *Revue d'Intelligence Artificielle* 30(5):509–532. https://doi.org/10.3166/ria.30.509-532.

Zhendong Dong, Qiang Dong, and Changling Hao. 2010. Hownet and its computation of meaning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 53–56. http://dl.acm.org/citation.cfm?id=1944284.1944298.

Damion Dooley, Emma Griffiths, Gurinder S. Gosal, Pier Luigi Buttigieg, Robert Hoehndorf, Matthew C. Lange, Lynn M. Schriml, Fiona S. L. Brinkman, and William W. L. Hsiao. 2018. Foodon: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food* 2. https://doi.org/10.1038/s41538-018-0032-6.

Miriam Fernández, Chwhynny Overbeeke, Marta Sabou, and Enrico Motta. 2009. What makes a good ontology? a case-study in fine-grained knowledge reuse. In Asunción Gómez-Pérez, Yong Yu, and Ying Ding, editors, *The Semantic Web*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 61–75.

E. Gaillard, J. Lieber, and E. Nauer. 2015. Improving ingredient substitution using formal concept analysis and adaptation of ingredient quantities with mixed linear optimization. In *Computer Cooking Contest Workshop*. Frankfort, Germany. https://hal.inria.fr/hal-01240383.

J.U. Kietz, A. Maedche, and R. Volz. 2000. A method for semi-automatic ontology acquisition from a corporate intranet. *EKAW-2000 Workshop Ontologies and Text, Juan-Les-Pins, France, October 2000* .

Mathieu Lafourcade. 2007. Making people play for Lexical Acquisition with the JeuxDeMots prototype. In *SNLP'07: 7th International Symposium on Natural Language Processing*. Pattaya, Chonburi, Thailand, page 7. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00200883.

Mathieu Lafourcade. 2011. *Lexique et analyse sémantique de textes - structures, acquisitions, calculs, et jeux de mots. (Lexicon and semantic analysis of texts - structures, acquisition, computation and games with words)*. https://tel.archives-ouvertes.fr/tel-00649851.

Ora Lassila and Deborah McGuinness. 2001. The role of frame-based representation on the semantic web. Technical report, Knowledge Systems Laboratory Report KSL-01-02, Stanford University, Stanford (USA).

Jacek Marciniak. 2013. Building wordnet based ontologies with expert knowledge. In *LTC*.

Thibault Mondary. 2011. *Construction d'ontologies à partir de textes. L'apport de l'analyse de concepts formels.*. Theses, Université Paris-Nord - Paris XIII. Equipe RCLN. https://tel.archives-ouvertes.fr/tel-00596825.

Joe Raad and Christophe Cruz. 2015. A Survey on Ontology Evaluation Methods. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management* . Lisbonne, Portugal. https://doi.org/10.5220/0005591001790186.

Lionel Ramadier. 2016. *Indexation and learning of terms and relations from reports of radiology*. Theses, Université de Montpellier. https://hal-lirmm.ccsd.cnrs.fr/tel-01479769.

Christophe Roche. 2007. Le terme et le concept : fondements d'une ontoterminologie. In *TOTh 2007 : Terminologie et Ontologie : Théories et Applications*. Annecy, France, pages 1–22. 22 pages. https://hal.archives-ouvertes.fr/hal-00202645.

Gilles Sérasset. 2014. DBnary: Wiktionary as a Lemon-Based Multilingual Lexical Resource in RDF. *Semantic Web – Interoperability, Usability, Applicability* pages –. To appear. https://hal.archives-ouvertes.fr/hal-00953638.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*. European Languages Resources Association (ELRA), Istanbul, Turkey, pages 3679–3686.

Sylvie Szulman. 2012. Logiciel Terminae - Version 2012. TERMINAE est une plateforme d'aide à la construction de ressources termino-ontologiques à partir de ressources textuelles. https://hal.archives-ouvertes.fr/hal-00719453.

Andon Tchechmedjiev. 2016. *Semantic Interoperability of Multilingual Lexical Resources in Lexical Linked Data*. Theses, Université Grenoble Alpes. https://tel.archives-ouvertes.fr/tel-01681358.

Davide Turcato, Fred Popowich, Janine Toole, Dan Fass, Devlan Nicholson, and Gordon Tisher. 2000. Adapting a synonym database to specific domains. In *ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval*. Association for Computational Linguistics, Hong Kong, China, pages 1–11. https://doi.org/10.3115/1117755.1117757.

Piek Vossen. 2012. Ontologies. *The Oxford Handbook of Computational Linguistics* https://doi.org/10.1093/oxfordhb/9780199276349.013.0025.

Wilson Wong. 2009. *Learning lightweight ontologies from text across different domains using the web as background knowledge*. Ph.D. thesis.

Manel Zarrouk. 2015. *Endogeneous Consolidation of Lexical Semantic Networks*. Theses, Université de Montpellier. https://hal-lirmm.ccsd.cnrs.fr/tel-01300285.

Naive Regularizers for Low-Resource Neural Machine Translation

Meriem Beloucif[1], **Ana Valeria Gonzalez**[2], **Marcel Bollmann**[2], **and Anders Søgaard**[2]

[1]Language Technology Group, Universität Hamburg, Hamburg, Germany
[2]Dpt. of Computer Science, University of Copenhagen, Copenhagen, Denmark

Abstract

Neural machine translation models have little inductive bias, which can be a disadvantage in low-resource scenarios. They require large volumes of data and often perform poorly when limited data is available. We show that using naive regularization methods, based on sentence length, punctuation and word frequencies, to penalize translations that are very different from the input sentences, consistently improves the translation quality across multiple low-resource languages. We experiment with 12 language pairs, varying the training data size between 17k to 230k sentence pairs. Our best regularizer achieves an average increase of 1.5 BLEU score and 1.0 TER score across all the language pairs. For example, we achieve a BLEU score of 26.70 on the IWSLT15 English–Vietnamese translation task simply by using relative differences in punctuation as a regularizer.

1 Introduction

One of the major challenges when training neural networks is overfitting. Overfitting is what happens when a neural network in part memorizes the training data rather than learning to generalize from it. To prevent this, neural machine translation (NMT) models are typically trained with an L_1 or L_2 penalty, dropout, momentum, or other general-purpose regularizers. General-purpose regularizers and large volumes of training data have enabled us to train flexible, expressive neural machine translation architectures that have provided a new state of the art in machine translation.

For low-resource language pairs, however, where large volumes of training data are *not* available, neural machine translation has come with diminishing returns (Koehn and Knowles, 2017). The general-purpose regularizers do not provide enough inductive bias to enable generalization, it seems. This is an area of active research, and other work has explored multi-task learning (Firat et al., 2016; Dong et al., 2015), zero-shot learning (Johnson et al., 2016), and unsupervised machine translation (Gehring et al., 2017) to resolve the data bottleneck. In this paper, we consider a fully complementary, but much simpler alternative: naive, linguistically motivated regularizers that penalize the output sentences of translation models departing heavily from simple characteristics of the input sentences.

The proposed regularizers are based on three surface properties of sentences: their length (measured as number of tokens), their amount of punctuation (measured as number of punctuation signs), and the frequencies of their words (as measured on external corpora). While there are languages that do not make use of punctuation (e.g., Lao and Thai), in general these three properties are roughly preserved across translations into most languages. If we translate a sentence such as (1), for example:

(1) That dog is a Chinook.

it is relatively safe to assume that a good translation will be short, contain at most one dot, and contain at least one relatively frequent word (for *dog*) and at least one relatively infrequent word (for *Chinook*). This assumption is the main motivation for our work.

Contributions Our contribution is three-fold: (a) We propose three relatively naïve, yet linguistically motivated, regularization methods for machine translation with low-resource languages.

Two of the regularizers are derived directly from the input, without relying on any additional linguistic resources. This makes them adequate for low-resource settings, where the availability of linguistic resources can generally not assumed. Our third regularizer (frequency) only assumes access to unlabeled data. (b) We show that regularizing a standard NMT architecture using naive regularization methods consistently improves machine translation quality across multiple low-resource languages, also compared to using more standard methods such as dropout. We also show that combining these regularizers leads to further improvements. (c) Finally, we present examples and analysis showing *how* the more linguistically motivated regularizers we propose, help low-resource machine translation.

2 Related Work

End-to-end neural machine translation is based on encoder–decoder architectures (Sutskever et al., 2014; Cho et al., 2014; Luong et al., 2015a, 2017), in which a source sentence $x = (x_1, x_2, ..., x_n)$ is encoded into a vector (or a weighted average over a sequence of vectors) $z = (z_1, z_2, ..., z_n)$. The hidden state representing z is then fed to the transducer (also called decoder) which generates translations, noted as $y = (y_1, y_2, ..., y_m)$.

Neural machine translation has achieved state-of-the-art performance for various language pairs (Luong et al., 2015a; Sennrich et al., 2015; Luong and Manning, 2016; Neubig, 2015; Vaswani et al., 2017), especially when trained on large volumes of parallel data, i.e., millions of parallel sentences (also called bi-sentences), humanly translated or validated. Such amounts of training data, however, are difficult to obtain for low-resource languages such as Slovene or Vietnamese, and in their absence, neural machine translation is known to come with diminishing returns, suffering from overfitting (Koehn and Knowles, 2017).

In order to avoid overfitting, NMT models are often trained with L_1 or L_2 regularization, as well as other forms of regularization such as momentum training or dropout (Srivastava et al., 2014; Wang et al., 2015; Miceli Barone et al., 2017). However, these regularization methods are very general and do not carry any language specific information.

On the other hand, it has been shown that transfer learning approaches using out of domain data, such as the European Parliament data[1], to regularize the learning helps improve the translation quality (Miceli Barone et al., 2017). This approach produces good results, but it is not applicable in low-resource settings because it requires large amounts of data in the language of interest. To the best of our knowledge, our work is the first to introduce naive, linguistically motivated regularization methods such as sentence length, punctuation and word frequency.

3 Model Description

3.1 Baseline

In order to show the impact that our regularizers have on the translation quality, we use an off-the-shelf NMT system described by Luong et al. (2017) as our baseline. The model consists of two multi-layer recurrent neural networks (RNNs), one that encodes the source language and one that decodes onto the target language. For the encoder cell, we use a single Long Short-Term Memory (LSTM) layer (Hochreiter and Schmidhuber, 1997) and output the hidden state, which then gets passed to the decoder cell.

We train our models to minimize the cross-entropy loss and back-propagate the loss to update the parameters of our model. We update network weights using Adam optimization (Kingma and Ba, 2014), which calculates the exponential moving average of the gradient and squared gradient, and combines the advantages of AdaGrad and RMSProp. For the purpose of comparison, we set the dropout to 0.2, similar to Luong et al. (2015b).

3.2 Regularized NMT

To apply our new regularizers, we add each regularizer to the loss function during the training of the NMT model (Luong et al., 2015a; Luong and Manning, 2016; Luong et al., 2017). Since we aim to minimize the cross-entropy loss, this means that we favor training instances which have a low penalty from the regularizers (e.g., a small length difference). Importantly, we do *not* use dropout in this scenario, as we want to contrast our naive, but linguistically motivated signals with a traditional, but not linguistically motivated regularization method, i.e., dropout.

Furthermore, we do not explore alternative ways for adding regularizers to the loss function here (other alternatives could be to have a

[1] http://www.statmt.org/europarl/

weighted penalty which is then tuned to find the best penalty and added to the loss function for testing). The main purpose of this work is to study the effect of naive linguistically motivated regularizers and show that they can improve translation quality; we leave it to future work to find the optimal configuration of regularizers that maximizes the overall translation quality.

4 Naive Regularizers

4.1 Length-Based Regularizer

NMT models have shown to suffer "the curse of sentence length", and it has been hypothesized that this is due to a lack of representation at the decoder level (Cho et al., 2014; Pouget-Abadie et al., 2014). Our proposed sentence-length-based regularizer penalizes relative differences between the input and the MT output lengths during the training of the NMT model:

$$\text{reg}_{\text{length}} = |l_0 - l_1| \tag{1}$$

Here, l_0 and l_1 represent the input sentence and the MT output sentence lengths, respectively, as measured by the number of words (not to be confused with L_1 and L_2 regularization methods).

Note that this regularizer is different from the word penalty feature in phrase-based machine translation (Zens and Ney, 2004), which only penalizes the target sentence length. The relative difference between the input and the MT output sentence lengths is also used as a feature in Marie and Fujita (2018).

4.2 Punctuation-Based Regularizer

The punctuation-based regularizer penalizes training instances whenever the amount of punctuation marks in the input sentence differs from the amount in the MT output sentence. It is computed as follows:

$$\text{reg}_{\text{punct}} = |p_0 - p_1| \tag{2}$$

Here, p_0 and p_1 is the total number of punctuation marks in the input and the MT output sentence, respectively.

Unfortunately, the only available methods to generate more efficient NMT models have included data intensive methods such as sentence alignment (Bahdanau et al., 2014). Some very early research done in alignment used simple methodologies such as punctuation-based alignment (Chuang et al., 2004). Our second regularizer is based on this simple idea, as it penalizes training instances where the quantities of punctuation marks differ between input and MT output sentences. Example (2) is taken from the training set of the French–English translation task:

(2) IN *Pas parce qu'ils sont moins bons, pas parce qu'ils sont moins travailleurs.*
REF And it's not because they're less smart, and it's not because they're less diligent.
OUT And . . . .

We note that the punctuation in the French input sentence matches the punctuation of the desired English reference. However, during an early training step, the NMT model translates the input to a sequence containing six times the number of punctuation marks in the input sentence, which is obviously incorrect. Our punctuation regularizer further penalizes examples like this one.

4.3 Frequency-Based Regularizer

Our last regularizer is based on the distribution of word frequencies between the source and the target sentences. Generally speaking, if the source sentence contains an uncommon word, we assume that its translation in the target language is also uncommon. The intuition behind this regularizer is that if the source sentence contains one uncommon word and three common words, then its accurate translation should contain similar word frequencies. The example below is extracted from the English–French translation task:

(3) IN *But now there is a bold new solution to get us out of this mess.*
REF Mais il exist une solution audacieuse pour nous en sortir.
OUT Mais maintenant il y a une solution pour nous en sortir.

The English sentence contains the frequent word *there* and the less frequent word *bold*. The French output sentence is acceptable, but it is not accurate since the English word *bold* (*audacieuse* in the reference translation) was omitted in the output. During training, the frequency regularizer penalizes such cases that have a big divergence between the word frequencies in the input and output sentences.

The purpose of our frequency-based regularizer

Languages	#Words
Czech	1.7M
English	85.57M
French	55.72M
German	35.47M
Russian	2.5M
Slovene	1.45M
Vietnamese	3.5M

Table 1: The size of the Wikipedia dumps (#words) used to calculate word frequencies for each language.

is to calculate how different the MT output sentence is from the source input in terms of vocabulary distribution. For instance, the frequency of using the word *chauve-souris* in French is almost similar to the frequency of using its English translation *bat* in English. The same could be applied for the more frequent words such as *et* in French and its English translation *and*.

We start by computing the frequency vectors $\overrightarrow{v_{\text{in}}}$ and $\overrightarrow{v_{\text{out}}}$, containing the frequency for every word w_i in the input and MT output sentence, respectively:

$$\overrightarrow{v} = \langle f(w_1), \ldots, f(w_n) \rangle \tag{3}$$

To calculate the word frequencies $f(w)$ for each language, we use the Wikipedia database[2] as an external resource. Table 1 contains the size of the datasets (in number of words) used to estimate these. We note that there is considerably more data for English and French than for e.g. Vietnamese (cf. Table. 1); we discuss the effect that this might have on the results in Sec. 6.

We interpret the resulting frequency vectors $\overrightarrow{v}$ as distributions, for which we now calculate the Kullback-Leibler (KL) divergence to obtain our regularization term:

$$\text{reg}_{\text{freq}} = D_{\text{KL}}(\overrightarrow{v_{\text{in}}}, \overrightarrow{v_{\text{out}}}) \tag{4}$$

Essentially, this regularizer penalizes translations if their word frequency distributions diverge too strongly from those of the source sentence.

(4) IN It was a big lady who wore a fur around her neck
REF C'était une dame forte qui portait une fourrure autour du cou
OUT C'était une femme forte portant une fourrure autour du cou

Example (4) shows an input sentence and its MT output, for which we would compute the frequency vectors as follows:

$$\overrightarrow{v_{\text{in}}} = \langle f(\text{'it'}), \quad f(\text{'was'}), \quad \ldots, f(\text{'neck'})\rangle$$
$$\overrightarrow{v_{\text{out}}} = \langle f(\text{'c'était'}), \quad f(\text{'une'}), \quad \ldots, f(\text{'cou'})\rangle$$

Languages	Sentence Pairs		
	Train	Development	Test
Czech	122,382	480	1,327
French	232,825	890	1,210
German	206,112	888	1,080
Russian	178,165	887	1,701
Slovene	17,125	1,144	1,411
Vietnamese	133,317	1,553	1,268

Table 2: The size of the training data in sentence pairs. To test our proposed models, we experiment by translating to/from English for every non-English language.

5 Experiments

5.1 Data

The purpose of our experiments is to show that signals such as sentence length, punctuation or word frequency help improve the translation quality of a standard neural machine translation architecture. To that effect, we experiment with 12 translation tasks, translating from English to six low-resource languages, and vice versa.

The six languages represent the following language families: Slavic, Romance, Germanic, and Austro-Asian. We further vary the size of the training data to test how our regularization methods affect the quality of the MT output in different setups. Table 2 contains the size of the training, development and test set for every language pair. Note that the training sets vary considerably in size, from 17k sentence pairs for Slovene to almost 233k for French.

The data is from the International Workshop on Spoken Language Translation (IWSLT), except for Russian, Slovene and Vietnamese which are from IWSLT 2015, the data for the remaining translation tasks is from IWSLT 2017 (Cettolo et al., 2012).

[2] https://en.wikipedia.org/wiki/Wikipedia:Database

Preprocessing The purpose of our experiments is to learn how to efficiently translate low-resource languages. For that purpose, we do not use any advanced preprocessing for any of our translation tasks except tokenization where we use the script from the Moses toolkit (Koehn et al., 2007). We also set the maximum sentence length to 70 tokens and the vocabulary size to 50k.

5.2 Training Details

We use the attention-based model described in Luong et al. (2015b). Our model is composed of two LSTM layers each of which has 512-dimensional units and embeddings; we also use a mini-batch size of 128. Adding an attention mechanism in neural machine translation helps to encode relevant parts of the source sentence when learning the model. We propose to add additional regularizers on top of the attention-based model at each translation step.

We have noticed that the convergence highly depends on the language pairs involved. While our baseline model is identical to the NMT model described by Luong et al. (2015b), we deviate from their training procedure by continuing the training until convergence, which for us took 15 epochs instead of the 12 epochs described by the authors. The convergence in our case is measured by the models having no improvements on the development set over five epochs.

Table 3 shows that our baseline is +1.5 BLEU points better than the scores reported by Luong et al. (2015b). On top of that, our length-based and punctuation-based models produce a statistically significant improvement over the baseline (+0.5 BLEU points).

We train all our models automatically until convergence. In Table 4, we report the number of epochs it took to converge by translation task when translating to/from English. We note that except for Czech and Slovene, which converged the quickest, most of the translation tasks took between 15k and 20k steps to converge.

6 Evaluation

In order to show that the naive regularizers which we propose in this paper significantly boost the translation quality, we test the machine translation output using the toolkit MultEval defined in Clark et al. (2011). In this paper, we report the results using three commonly used metrics: the n-gram based metrics BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005), as well as the error-rate based metric TER (Snover et al., 2006). The evaluation metric BLEU (Papineni et al., 2002) is based on n-gram matching between the input and the output, whereas the error-rate based metric TER (Snover et al., 2006) measures how many edits are needed so that the machine translation resembles the man-made reference.

System	**BLEU**
Luong et al. (2015)	23.30
Luong et al. (2017) (dropout=0.2)	25.10
Baseline (dropout=0.2)	26.43
+ Length	**26.77**
+ Punct	26.71
+ Frequency	26.12
+ Combined	**27.13**

Table 3: Baseline vs. our proposed models on the English–Vietnamese translation task, using the same dataset as Luong et al. (2015b). The results in bold represent statistically significant results compared to the baseline according to MultEval (Clark et al., 2011).

Translation Task	**#Steps**
Lang→English	
Czech	12K
French	20K
German	20K
Russian	22K
Slovene	10K
Vietnamese	15K
English→Lang	
Czech	12K
French	22K
German	20K
Russian	18K
Slovene	11K
Vietnamese	15K

Table 4: Number of steps it took until the models stopped improving for all the translation tasks.

6.1 Results

Table 5 shows the results for all language pairs and all metrics. We observe an improvement over the

	System	Languages					
		Czech	French	German	Russian	Slovene	Vietnamese
EN→Lang	Baseline	14.01	32.13	22.07	12.87	5.60	26.43
	Length	**14.65**	32.32	21.64	12.81	4.98	**26.77**
	Punct	**14.98**	**32.79**	**22.89**	13.06	5.64	26.71
	Frequency	**14.75**	**33.47**	22.14	**13.50**	1.95	26.12
Lang→EN	Baseline	21.32	31.51	24.41	15.39	8.85	24.94
	Length	21.83	31.09	24.56	15.29	9.05	**25.87**
	Punct	**21.96**	**32.43**	**25.17**	**16.36**	**9.63**	25.32
	Frequency	21.88	**32.26**	24.87	15.90	9.18	24.35

(a) BLEU

EN→Lang	Baseline	17.62	51.11	40.47	16.12	26.52	11.46
	Length	**18.41**	51.10	39.93	**16.80**	**27.03**	12.01
	Punct	**18.43**	**51.67**	**41.18**	16.77	27.00	**12.30**
	Frequency	**18.16**	**52.10**	40.57	16.79	26.95	12.29
Lang→EN	Baseline	24.66	31.77	27.23	20.63	16.28	28.11
	Length	**25.07**	31.55	27.11	20.65	15.95	**28.71**
	Punct	**25.10**	**32.31**	**27.75**	**21.45**	**17.05**	**28.48**
	Frequency	**25.27**	**32.16**	27.43	20.80	**16.85**	27.86

(b) METEOR

EN→Lang	Baseline	62.64	49.21	57.17	70.17	77.20	54.29
	Length	62.18	48.96	57.90	70.85	79.51	**53.93**
	Punct	**61.69**	**48.57**	57.24	70.04	**77.02**	54.03
	Frequency	62.46	48.87	57.63	**69.40**	87.20	54.99
Lang→EN	Baseline	57.06	46.42	53.31	63.62	72.46	53.66
	Length	**55.68**	46.44	53.29	63.31	72.54	**52.74**
	Punct	**56.29**	**45.37**	**52.31**	**62.24**	**72.11**	53.51
	Frequency	57.32	**45.55**	52.75	62.10	75.73	54.72

(c) TER

Table 5: Contrasting our three proposed models to the baseline (NMT; Luong et al., 2017) across 12 translation tasks. We evaluate all the models using BLEU, METEOR and TER. The bold values represent the models that show statistically significant improvements over the baseline ($p < 0.001$; Clark et al., 2011). Note that for BLEU and METEOR, higher is better, while for TER, lower is better. All regularization schemes almost consistently lead to improvements, with the punctuation-based regularizer achieving the highest gains.

baseline across almost all language pairs for all models and across all metrics. We obtain statistically significant results for almost all translation tasks for at least one regularization method.

More specifically, the punctuation regularizer outperforms all the other models on all translation tasks except for French–English and English–French. For the latter, we observe that the word frequency regularizer is better than the other systems. This could be explained by the fact that the English vocabulary has many words borrowed from French, which makes the word frequency regularizer a better signal than punctuation or sentence length for this specific task. It also could be due to the fact that both English and French have the largest vocabulary for training the word frequencies (cf. Table 1; English has around 80M words and French has around 50M words, whereas all other languages have much less data).

The most challenging translation tasks are Slovene–English and English–Slovene, especially in terms of error rate. The results show that with 17k sentence pairs as a training set, it becomes more challenging to efficiently learn anything. The results we obtained are between 2 and 5 BLEU points when translating from English. The Slovene output contained many non-translated words. Specifically, this task greatly suffers when using the word frequency regularizer, with an error increase of about 10 TER points from English to Slovene. We do not observe such losses for the Czech–English and English–Czech translation tasks, even though the vocabulary size for estimating the word frequencies is lower for Czech. We hypothesize that this is due to the Czech training set being seven times larger than the Slovene one. We hypothesize that this is due to the fact that for Slovene we only have 17K sentence pairs for the training step; whereas for Czech, we have 122K sentence pairs, which helped control the model compared to Slovene.

One case where the punctuation regularizer succeeds consistently is on the English–German and German–English translation tasks, with an error reduction of about 1 TER point. This reflects the similarity in punctuation between these languages. Although we also observe improvements using the other regularization methods, e.g. the length-based method, these are not statistically significant here as calculated by MultEval (Clark et al., 2011).

Table 3 shows the BLEU scores of seven different systems including the one where we combine our three regularizers on the English–Vietnamese translation task. The combined regularizer does not only produce a statistically significant improvement of almost 1-BLEU point over the attention based baseline, but it also outperforms all the other regularizers achieving a BLEU score of 27.23.

7 Translation Examples

The punctuation regularizer outperforms the baseline in most cases, and all of our regularization methods show statistically significant improvements in at least one language. Below we present examples, extracted from the test data, of how each of the regularization methods affects the output in comparison to the baseline model. The purpose of the examples is to show how each objective function in the learning component affects the performance component.

7.1 Frequency-Based Regularizer

The frequency-based regularization method penalizes cases where the distribution of the target vocabulary greatly differs from the source vocabulary. We have noted a significant improvement for this specific regularizer when translating from French to English and vice-versa. Examples (5) and (6) show how this regularizer is improving the translation output.

(5) IN 90 % de notre temps entourés par l'architecture .
REF That's 90 percent of our time surrounded by architecture .
BASE <unk> percent of our time **via** architecture .
FREQ <unk> percent of our time **surrounded** by architecture .

(6) IN Débloquer ce potentiel est dans **l'intérêt** de chacun d'entre nous .
REF Unlocking this potential is in the **interest** of every single one of us .
BASE <unk> that potential is in all of us .
FREQ <unk> that potential is in the **interest** of all of us .

More precisely, *entourés* in French is almost as frequent as *surrounded* in English, which is a word that our model with frequency-based regularization translates correctly, while the baseline does not. Additionally, in Example (6), our model has a better fluency and adequacy than the baseline since it not only correctly translates *l'intérêt* to *interest*, but also correctly produces *of all* instead of *in all*, as in the baseline output.

7.2 Punctuation-Based Regularizer

The punctuation-based regularization performs best in the German–English and English–German translation tasks. This regularizer penalizes cases where the difference in the number of punctuation between the source and the target sentences is particularly large. As seen in Example (7), simply introducing this bias into a translation model leads to an output which more closely matches the punctuation of the source and target sentences.

(7) IN Und die Antwort , glaube ich , ist ja . [" F = T ∇ S$\mathcal{T}$ "] . Was Sie gerade sehen , ist wahrscheinlich die beste Entsprechung zu $E = mc^2$ für Intelligenz , die ich gesehen habe .
REF And the answer , I believe , is yes . [" F = T ∇ S$\mathcal{T}$ "] What you're seeing is probably the closest equivalent to an $E = mc^2$ for intelligence that I've seen .
BASE And the answer , I think , is yes .
PUNC And the answer , I think , is yes . [**" R = T T <unk> "**] **What you're looking at is probably the best <unk> <unk> <unk> of intelligence that I've seen .**

The baseline MT output completely fails to capture anything from the input except for the first part up to "...is yes." Our punctuation-based model, however, manages to capture most parts of the sentence.

7.3 Length-Based Regularizer

Finally, the length-based regularization method leads to noticeable improvements in the Czech–English and English–Czech translation tasks. Example (8) shows that introducing an input sentence length bias led to an MT output that is much closer to the reference than the baseline. The input sentence consists of 12 tokens (including punctuation), the baseline output consists of 10 tokens, while our length based regularization model preserves the length of 12 tokens.

(8) IN V roce 2009 jsem ztratila někoho , koho jsem velmi milovala .
REF In 2009 , I lost someone I loved very much .
BASE In 2009 , I lost somebody who I loved .
LEN In 2009 , I lost somebody who I loved **very much .**

7.4 General Improvements

The Slovene dataset is our smallest with about 17k sentence pairs for training. Despite the low amount of resources available in Slovene, we found that introducing very naive linguistic biases into our machine translation models actually leads to subtle differences that result in an output closer to the reference, not only lexically, but also semantically. In Example (9), we compare the output of the frequency based system against the baseline for the Slovene to English translation:

(9) IN In kaj potem ?
REF And so , what after that ?
BASE And then **then** ?
FREQ And then **, what** ?

In this particular case, the frequency based regularization model takes care of the translation of the word *what*, and although the word *so* is not translated, the overall meaning of the source is preserved.

(10) IN Imeti moraš otroke , da preživiš .
REF You need to have children to survive .
BASE **Well you have the kids that you need** to educate .
FREQ You **have to have kids** to educate .

Example (10) shows another case of how the output of the frequency-based regularization system actually shows overall improvements in an extremely low-resource language. The output of our system is semantically closer to the reference than the baseline output, up to the word *educate*. In addition, the system preserves a similar length as the source sentence.

(11) IN Mi smo tu na vrhu .
REF We are here on top .
BASE **What** we are at the top .
FREQ We are **here** at the top .

Finally, Example (11) shows a low-resource case where our system manages to make subtle changes in order to reach the correct translation, whereas the baseline system does not.

8 Conclusion

We have shown that using naive regularization methods based on sentence length, punctuation, and word frequency consistently improves the translation quality in twelve low-resource translation tasks. The improvement is consistent across multiple language pairs and is not dependent on the language family. We have reported and discussed examples demonstrating why and how each regularizer is improving the translation quality.

Our proposed approach shows that even naive, but linguistically motivated, regularizers help improve the translation quality when training NMT models. We believe this shows the usefulness of using task-related regularizers for improving neural models, and opens the door for future work to exploit these regularization methods in an even more efficient manner by experimenting with different ways of combining the regularizers with the loss function.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, pages 65–72. https://www.aclweb.org/anthology/W05-0909.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*. Trento, Italy, pages 261–268.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, pages 103–111. https://www.aclweb.org/anthology/W14-4012.

Thomas C Chuang, Jian-Cheng Wu, Tracy Lin, Wen-Chie Shei, and Jason S Chang. 2004. Bilingual sentence alignment based on punctuation statistics and lexicon. In *International Conference on Natural Language Processing*. Springer, pages 224–232.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 176–181. https://www.aclweb.org/anthology/P11-2031.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1723–1732. https://doi.org/10.3115/v1/P15-1166.

Orhan Firat, KyungHyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *CoRR* abs/1601.01073. http://arxiv.org/abs/1601.01073.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR* abs/1705.03122. http://arxiv.org/abs/1705.03122.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558. http://arxiv.org/abs/1611.04558.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. https://www.aclweb.org/anthology/P07-2045.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, pages 28–39. https://www.aclweb.org/anthology/W17-3204.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. https://github.com/tensorflow/nmt.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1054–1063. https://doi.org/10.18653/v1/P16-1100.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025. http://arxiv.org/abs/1508.04025.

Benjamin Marie and Atsushi Fujita. 2018. A smorgasbord of features to combine phrase-based and neural machine translation. In *AMTA*.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 1490–1495. https://www.aclweb.org/anthology/D17-1156.

Graham Neubig. 2015. lamtram: A toolkit for language and translation modeling using neural networks. https://github.com/neubig/lamtram.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In

Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. https://www.aclweb.org/anthology/P02-1040.

Jean Pouget-Abadie, Dzmitry Bahdanau, Bart van Merriënboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *Syntax, Semantics and Structure in Statistical Translation* page 78.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR* abs/1508.07909. http://arxiv.org/abs/1508.07909.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*. The Association for Machine Translation in the Americas, pages 223–231. http://mt-archive.info/AMTA-2006-Snover.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Zhen Wang, Tingsong Jiang, Baobao Chang, and Zhifang Sui. 2015. Chinese semantic role labeling with bidirectional recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1626–1631. https://doi.org/10.18653/v1/D15-1186.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 257–264. https://www.aclweb.org/anthology/N04-1033.

Exploring Graph-Algebraic CCG Combinators for Syntactic-Semantic AMR Parsing

Sebastian Beschke
University of Hamburg, Germany
beschke@informatik.uni-hamburg.de

Abstract

We describe a new approach to semantic parsing based on Combinatory Categorial Grammar (CCG). The grammar's semantic construction operators are defined in terms of a graph algebra, which allows our system to induce a compact CCG lexicon. We introduce an expectation maximisation algorithm which we use to filter our lexicon down to 2500 lexical templates. Our system achieves a semantic triple (Smatch) precision that is competitive with other CCG-based AMR parsing approaches.

1 Introduction

Parsing sentences to formal meaning representations, known as *Semantic Parsing*, is a task at the frontier of Natural Language Understanding. Abstract Meaning Representation (AMR) is a meaning representation language that represents sentence semantics in the form of graphs. Research on AMR parsing systems has been very productive in recent years with many competing approaches.

Current AMR parsers vary regarding the extent to which they rely on a formal grammar. Some of the most successful systems generate AMRs through an end-to-end neural architecture, with no explicit symbolic derivations (Zhang et al., 2019). Other parsers employ transition systems with limited explanatory power (Peng et al., 2018). Constructing grammar-based semantic analyses that can be understood in terms of linguistic theory is a more difficult task than end-to-end AMR parsing because of the additional structural requirements on the output and the algorithmic constraints imposed thereby.

In this paper, we explore how semantic parsers can be built to be *interpretable* and *transparent*. Interpretability means that our system exposes rich symbolic information in the form of CCG derivations. Transparency means that it works with a compact and intuitively plausible lexicon. The lexicon is itself an artifact that can be inspected.

We achieve these goals by equipping CCG with graph-based semantics. Meaning reprentations are constructed through the operations of a simple graph algebra, which effectively constrains the search space for parsing and lexicon induction and makes the available operations and resulting lexical items easy to understand.

Technical contributions of this paper include a modified expectation-maximisation (EM) algorithm to induce compact delexicalised CCG lexica, a technique for training a syntactic-semantic supertagger with incomplete labels, and a hybrid update mechanism for training the linear parsing model.

1.1 Related Work

This work builds upon the concept of graph-algebraic CCG, which has so far been tested only in the context of lexicon induction (Beschke and Menzel, 2018). We extend the lexicon induction process by delexicalisation and EM filtering and demonstrate the first end-to-end parsing system based on graph-algebraic CCG. The idea of applying graph algebras to AMR parsing has also been applied in the context of Interpreted Regular Tree Grammar (Groschwitz et al., 2018). Furthermore, improved definitions of graph-composing CCG combinators have been proposed (Blodgett and Schneider, 2019) to cover a wider range of semantic phenomena.

Other systems that apply CCG to AMR parsing use an encoding of AMR graphs to λ-calculus expressions (Artzi et al., 2015; Misra and Artzi, 2016). One drawback of these systems is that lexicon induction is coupled to the training loop of a

Proceedings of Recent Advances in Natural Language Processing, pages 112–121,
Varna, Bulgaria, Sep 2–4 2019.

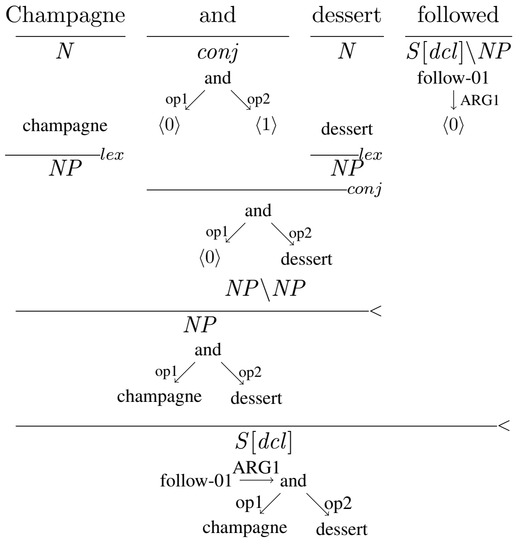


Figure 1: An example graph-algebraic CCG derivation.

parser, which makes it compute-intensive and difficult to manage. We address this issue by performing lexicon induction in a separate step.

Besides AMR parsing, CCG has also been used for joint syntactic-semantic parsing in other contexts (Krishnamurthy and Mitchell, 2014; Lewis et al., 2015).

2 Background

This paper uses Combinatory Categorial Grammar (CCG) to derive Abstract Meaning Representations (AMR) using a graph-algebraic modification of CCG's syntax-semantics interface. These concepts are briefly introduced in this below.

2.1 CCG for Semantic Parsing

Combinatory Categorial Grammar (CCG) describes syntax and semantics as part of the same derivation process (Steedman, 2000). CCG derivations are trees where every node is annotated with both a syntactic and a semantic category. The categories at the leaves of the tree are drawn from a *lexicon*, while categories at the inner nodes result from the application of *combinatory rules* to the child nodes' categories. The syntax-semantics interface in CCG is *transparent*, meaning that the same rule is always applied to syntactic and semantic categories.

In CCG, categories are understood as n-ary *functions*. *Syntactic categories* essentially express the type of the associated semantic category by specifying the types of constituents that can be accepted as arguments, either to the right of the constituent or to the left. This directionality is expressed by forward and backward slashes. E.g., given the atomic syntactic categories S for sentences and NP for noun phrases, the complex category $(S\backslash NP)/NP$ represents a transitive verb, accepting first an NP to the right and then another NP to the left to produce a sentence.

Semantic categories contain building blocks for sentential meaning representations. Traditionally, λ-calculus is used to represent the compositionality of semantic categories, while the object language that is being composed is a logical representation of sentence meaning. This paper deviates from that tradition by using a graph representation for semantic categories which is defined Section 2.3.

2.2 Abstract Meaning Representation

The Abstract Meaning Representation (AMR; Banarescu et al., 2013) is a meaning representation language that underlies much recent work in semantic parsing. In AMR, meaning is annotated on the sentence level in the form of a labeled, directed graph. While the nodes of the graph represent instances of concepts, edges represent roles that these entities play with respect to each other.

2.2.1 Evaluation of AMRs

AMR parsers are usually evaluated with respect to the *Smatch* metric (Cai and Knight, 2013), which measures precision and recall of semantic triples in an AMR graphs with respect to a gold standard graph. The computation of Smatch relies on finding an optimal alignment between the two graphs, which is usually approximated.

2.3 Graph-Algebraic CCG

Graph algebras are an established means to model the derivation of AMRs (Koller, 2015). A modification of CCG that applies graph-algebraic operators to semantic categories has first been presented by Beschke and Menzel (2018). They define a set of semantic operators that apply to *s-graphs*, which contain specially marked *source nodes*, which are consecutively indexed starting from 0. They also define three semantic operators:

- *Application*, which 1) merges the root of an argument graph with the highest-indexed source node of the function graph and 2) merges all source nodes that have the same index.

Combinator	Left Operand	Right Operand		Result
$>$	$X/Y : \bigcirc\!\!\rightarrow \langle 0\rangle$	$Y : \Diamond$	$\Rightarrow$	$X : \bigcirc\!\!\rightarrow\!\Diamond$
$<$	$Y : \Diamond$	$X\backslash Y : \bigcirc\!\!\rightarrow \langle 0\rangle$	$\Rightarrow$	$X : \bigcirc\!\!\rightarrow\!\Diamond$
$\mathbf{B}>$	$X/Y : \bigcirc\!\!\rightarrow \langle 0\rangle$	$Y/Z : \Diamond\!\!\rightarrow \langle 0\rangle$	$\Rightarrow$	$X/Z : \bigcirc\!\!\rightarrow\!\Diamond\!\!\rightarrow \langle 0\rangle$
$\mathbf{B}_{\times}<$	$Y/Z : \Diamond\!\!\rightarrow \langle 0\rangle$	$X\backslash Y : \bigcirc\!\!\rightarrow \langle 0\rangle$	$\Rightarrow$	$X/Z : \bigcirc\!\!\rightarrow\!\Diamond\!\!\rightarrow \langle 0\rangle$
$\mathbf{B}^2>$	$X/Y : \bigcirc\!\!\rightarrow \langle 0\rangle$	$(Y/Z_1)/Z_2 : \Diamond\!\!\rightarrow \langle 0\rangle$ $\searrow \langle 1\rangle$	$\Rightarrow$	$(X/Z_1)/Z_2 : \bigcirc\!\!\rightarrow\!\Diamond\!\!\rightarrow \langle 0\rangle$ $\searrow \langle 1\rangle$
conj	conj : $\mathit{conj} \rightarrow \langle 0\rangle$ $\searrow \langle 1\rangle$	$X : \bigcirc\!\!\rightarrow \langle 0\rangle$	$\Rightarrow$	$X\backslash X$: $\mathit{conj} \rightarrow \langle 1\rangle$ $\searrow \bigcirc\!\!\rightarrow \langle 0\rangle$
rp	$X : \Diamond$	$. : \epsilon$	$\Rightarrow$	X : $\Diamond$
lp	$. : \epsilon$	$X : \Diamond$	$\Rightarrow$	X : $\Diamond$

Table 1: The set of binary combinators used in our system. Circles and diamonds correspond to arbitrary AMR subgraphs. X and Y represent arbitrary syntactic categories. The *conj* node represents any concept corresponding to a conjunction, such as *and* or *contrast*. Edge labels are omitted.
The combinators Forward Application ($>$), Backward Application ($<$), Forward Composition ($\mathbf{B}>$), Backward Crossed Composition ($\mathbf{B}_{\times}<$), and Forward Generalised Composition ($\mathbf{B}^2>$) all use the *Application* semantic operator. The Conjunction (*conj*) combinator uses the *Conjunction* semantic operator, and Left and Right Punctuation (*lp*, *rp*) use *Identity*.

- *Conjunction*, which 1) merges the root of an argument graph with the 1-indexed source node of the conjunction graph, and 2) renames the 0-indexed source node of the conjunction graph so that it becomes the highest-indexed source node in the combined graph (thus becoming accessible for application).

- *Identity*, in which the function graph is empty and the argument graph is returned unchanged.

An overview of the rules as well as how they are applied in the context of CCG derivations is given in Table 1.

An example derivation is given in Figure 2.

3 Lexicon Induction

For parsing with graph algebraic CCG, a lexicon must first be obtained. We achieve this using the *recursive splitting* algorithm by Beschke and Menzel (2018), which uses the following information to induce lexical items from an AMR-annotated sentence:

- The sentence's AMR

- A syntactic CCG parse obtained from a syntax parser

- A set of alignments linking tokens in the sentence to nodes in the meaning representation, obtained from automatic alignment tools

A set of lexical items explaining the sentence can then be obtained by walking down the syntactic parse tree, starting at the root with the full sentential meaning representation. At each binary derivation step, the meaning representation is partitioned into two subgraphs by unmerging nodes as appropriate. Each split is done in such a way that it can be reversed using a graph algebraic combinator and the token-to-node alignments are honored.

For any token, this procedure may generate several or no lexical entries. If the alignments do not uniquely specify how the meaning representation should be divided in a splitting step, all alternatives are explored. Also, splitting may abort at an inner node of the derivation if there is no combinator that satisfies the alignment constraint.

This work adds two steps to the lexicon induction process: the delexicalisation of lexical items, followed by filtering for the most probable derivation for each sentence according to EM estimates.

3.1 Delexicalisation

We achieve generalisation over content words by delexicalising lexical entries. We follow the approach from Kwiatkowski et al. (2011) which divides lexical entries into *templates* and *lexemes*. A template is a graph wherein up to one node has been replaced by a *lex* marker. A lexeme x—y is a pair of a word x and a node label y. For examples of templates and lexemes, see Table 2.

The idea of the delexicalisation algorithm is that a node in the graph which corresponds to the lexical meaning of the lexical entry is replaced by a marker, converting it into a template. Since it is not known in advance which node carries the lexical meaning, we replace every node in turn and add all resulting templates to the lexicon. Every replaced node label is associated with the token currently under consideration and stored as a lexeme.

Not all lexical entries contain a node with lexical meaning, e.g. in the case of function words. Therefore, the original meaning representation is also added to the lexicon as a template along with an empty lexeme.

Special lexemes are also added that map any word to a node labeled by the word's lemma, its surface form in quotes, or any of the propbank frame names associated with its lemma.

This process creates a large amount of superflous template/lexeme pairs. Therefore, the lexicon is subsequently filtered using Expectation Maximisation.

3.2 Expectation Maximisation

Both splitting and delexicalisation generate spurious templates and lexemes. We wish to keep only those that generalise well by being broadly applicable. In contrast, noise introduced during grammar induction should be removed.

This noise manifests itself in spurious derivations for the sentences of the training set. Expectation Maximisation (EM) is applied to identify a single most likely derivation per sentence. Every template and lexeme that in not used in at least one of these derivations is deleted.

We use a variant of the inside-outside algorithm (Baker, 1979) to estimate multinomial distributions P_t for templates and P_l for lexemes. From these, we derive a probability distribution over derivations:

$$P(d) = \prod_{(t,l)\in\textsc{lex}(d)} P_t(t)P_l(l)$$

where $\textsc{lex}(d)$ gives all template-lexeme pairs instantiated by the derivation.

Algorithm 1 Variation of the inside-outside algorithm to estimate parameters over CCG derivations. See Section 3.2 for function definitions.

```
Input: Data set $S$; scoring function $\textsc{Score}^i$
Output: Distributions $P_T^{i+1}$ and $P_L^{i+1}$
$count_T[j] \leftarrow 0$ for $0 \le j < |T|$
$count_L[j] \leftarrow 0$ for $0 \le j < |L|$
for $s \in S$ do
    $chart \leftarrow \textsc{Split}(s)$
    $likelihood \leftarrow \sum_{e\in chart[0,|s|-1]} \textsc{in}^{i+1}(e)$
    for $e \in chart$ do
        $c \leftarrow \frac{\textsc{Score}^i(e)\textsc{out}^{i+1}(e)}{likelihood}$
        for $(t,l) \in \textsc{Delex}(e)$ do
            $count_T[t] \leftarrow count_T[t] + c$
            $count_L[l] \leftarrow count_L[l] + c$
        end for
    end for
end for
$P_T^{i+1}(t) = \frac{count_T[t]}{\sum_{t'\in T} count_T[t']|}$ for $t \in T$
$P_L^{i+1}(l) = \frac{count_L[l]}{\sum_{l'\in L} count_L(l')|}$ for $l \in L$
```

Our inside-outside algorithm operates on *split charts*, which keep track of all derivation nodes created during recursive splitting. A split chart c for a sentence s contains cells $c[i,j]$ with $0 < i \le j, j < |s|$. A cell contains a number of entries e, each of which is associated with a meaning representation $\textsc{MR}(e)$ and a (possibly empty) set of child pairs $(l,r) \in \textsc{Cld}(e)$, which are in turn entries. An entry can also have several parents $e' \in \textsc{Par}(e)$, in which case it also has a neighbour $\textsc{Nb}(e,e')$ for every parent e'.

To compute a probability for an entry, we employ a function $\textsc{Delex}$ which decomposes the entry's meaning representation into all possible template-lexeme pairs.

Inside and outside probabilities for entries are calculated recursively as follows:

$$\textsc{in}^{i+1}(e) = \textsc{Score}^i(e) + \sum_{(l,r)\in\textsc{Cld}(e)} \textsc{in}^{i+1}(l)\cdot\textsc{in}^{i+1}(r)$$

$$\textsc{out}^{i+1}(e) = \sum_{e'\in\textsc{Par}(e)} \textsc{out}^{i+1}(e') \cdot \textsc{in}^{i+1}(\textsc{Nb}(e,e'))$$

where

$$\textsc{Score}^i(e) = \sum_{(t,l)\in\textsc{Delex}(\textsc{MR}(e))} P_T^i(t)P_L^i(l)$$

A given meaning representation $\textsc{MR}(e)$ can be created by either instantiating a lexical entry with probability $\textsc{Score}^i(e)$, or by deriving it using any of its pairs of children (l, r) with probability $\textsc{IN}^i(l)\textsc{IN}^i(r)$. All of these are alternative choices; therefore, the probabilities are summed to make up the inside probability. The outside probability is composed of the entry's parents' outside probabilities and the entry's neighbours' inside probabilities.

Algorithm 3.2 describes how an updated set of parameters is estimated using these calculations.

4 Parsing

Our parser uses a CKY-style chart parsing algorithm to parse sentences to AMR. For each token, template-lexeme pairs are drawn from the lexicon. Recursively, derivation nodes are created according to CCG/AMR rules. All candidate derivation nodes are evaluated with respect to a linear model. A beam search limitation is applied, meaning that only the top n candidates from each chart cell are kept.

The flip side of using a delexicalised lexicon is that every template can now be applied to every token. To limit the number of leaves that have to be considered, we employ a supertagger which predicts the most suitable template for each token. We then limit our search to the most probable templates as predicted by the supertagger.

4.1 Supertagging

For supertagging, we use a single-layer BiLSTM. For inputs, the raw tokens and syntactic CCG categories predicted by a CCG supertagger are used. The model is then trained to predict the template instantiated by each token.

The following preprocessing steps are applied:

- Tokens are embedded using the third layer produced by ELMo (Peters et al., 2018).

- CCG supertags, as well as templates, that occur in less than two sentences are replaced by `UNK`.

To predict supertags on the *dev* and *test* sections, we train the supertagger on the entire *train* section and output the predicted token-wise supertag distributions (clipping at 99% cumulative probability). To obtain supertag predictions on the *train* section, we employ 5-way jackknifing: the data is split into five parts and predictions for each part are obtained by training on the remaining four parts.

During training, the occurrence of the correct label within the top-10 predictions for every token is monitored and training aborted when this measure stops improving (early stopping).

4.1.1 Limited Supervision

The grammar induction process as described in Section 3 attempts to find lexical items for every individual token, but may stop early if no combinatory rule fitting the alignment constraint is available. In this case, no supervision for training the tagger is available at the token level. We overcome this issue by labelling the respective tokens as `UNK` (the same label used for rare templates occurring only once) and masking `UNK` tokens in the loss function.

This allows the tagger to fill in the gaps with reasonable templates that are in the lexicon. However, it also means that not every sentence from the train set can be perfectly parsed any more, because it is possible that its meaning representation cannot be constructed using the induced token-level lexical entries.

5 Training

To drive the parser, we train a linear model over graph algebraic CCG derivations. Since we do not observe derivations in the data, this is an instance of latent variable learning and a supervision signal must be generated. We take a dual approach by combining two weak supervision signals:

1. An oracle is used to heuristically generate silver-standard derivations, which can then be used for training.

2. The derivations found by the parser are evaluated and used for cost-sensitive parameter updates.

5.1 Model

We train a linear model using a structured perceptron algorithm (Collins, 2002) with Adadelta updates (Zeiler, 2012). We use features over paths in the graph as well as the identities of invoked templates, lexemes, and combinators.

Template	Lexeme	Combined
$N:\ lex$	weapons—weapon	$N:$ weapon
$NP/N:\ \langle 0\rangle$	the—∅	$NP/N:\ \langle 0\rangle$
$N/N:\ \langle 0\rangle \xrightarrow{\text{mod}} lex$	nuclear—nucleus	$N/N:\ \langle 0\rangle \xrightarrow{\text{mod}}$ nucleus
$(S\backslash NP)/NP:\ lex \xrightarrow{\text{ARG0}} \langle 0\rangle$ $\xrightarrow{ARG1} \langle 1\rangle$	said—say-01	$(S\backslash NP)/NP:$ say-01 $\xrightarrow{\text{ARG0}} \langle 0\rangle$ $\xrightarrow{ARG1} \langle 1\rangle$
$NP:$ country $\xrightarrow{\text{name}}$ name $\xrightarrow{\text{op1}}$ lex	Iran—"Iran"	$NP:$ country $\xrightarrow{\text{name}}$ name $\xrightarrow{\text{op1}}$ "Iran"

Table 2: Selected templates and lexemes from the induced lexicon. The templates are among the 20 most highly scored according to EM parameters; the lexemes among the top 50.

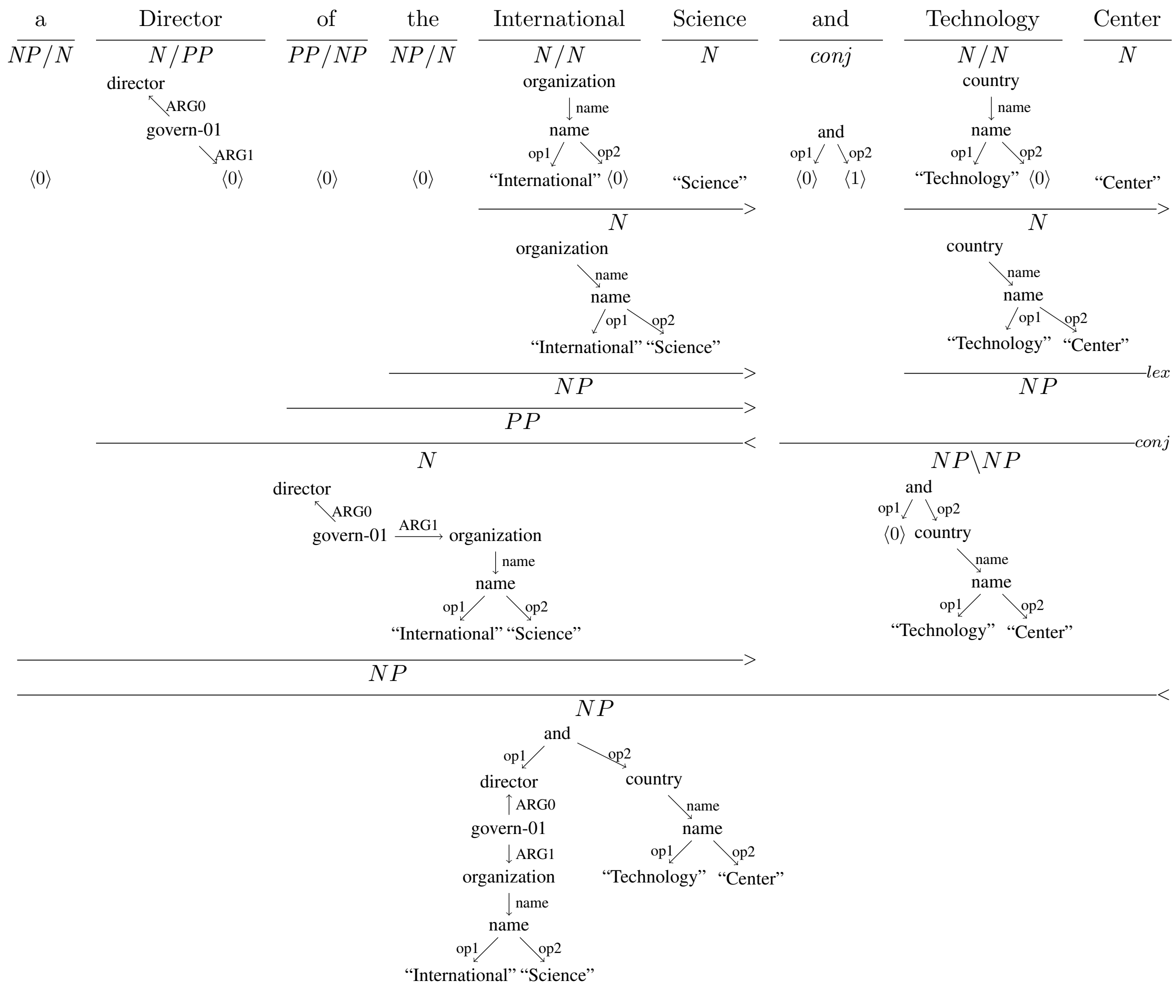


Figure 2: A derivation for a subsequence of PROXY_NYT_ENG_20020406_0118.25, as produced by our parser.

5.2 Oracle Parsing

In latent variable learning in structured prediction settings, the challenge is to obtain an unobserved derivation for a known gold-standard result. In this case, gold-standard sentential AMRs are annotated in the AMR corpus, but they are not related to the sentence by a grammatical derivation.

A common approach to this challenge is *forced decoding* (Artzi et al., 2015): the parser is used to construct derivations which lead to the correct result by pruning all hypotheses from the search space which deviate from the gold-standard AMR. E.g., all AMRs that contain elements not present in the gold-standard could be pruned.

However, as noted in Section 4.1, not every gold-standard AMR can be reconstructed perfectly using the induced lexicon due to the incompleteness of the splitting algorithm, which defies finding correct parses using forced decoding.

Instead, we train the parser using an oracle driven by a heuristic scoring function which scores the correctness and completeness of an intermediate hypothesised AMR. We parse the sentence using CKY with beam search, ranking intermediate results according to the harmonic mean of the following values:

- **Triple precision:** the proportion of node-edge-node triples in the intermediate result that also occur in the gold standard meaning representation.

- **Alignment recall:** the proportion of node labels that are linked by an alignment edge to one of the intermediate result's tokens that also occur in the intermediate result.

This scoring function is designed to rank results in proportion to their deviation from the gold standard, achieving a soft form of pruning.

Having obtained a set of derivations using oracle parsing, we finally re-rank these derivations by their Smatch f1 scores and use the best derivation to perform a parameter update using an early update strategy (Collins and Roark, 2004).

5.3 Cost-Sensitive Update

Another approach to training with weak supervision for structured prediction are cost-sensitive updates. While the gold-standard to update towards is unknown, an evaluation metric is available for the AMR that results from a specific derivation. Cost-sensitive updates let the parser search for complete derivations and enforce a margin between the best derivations in the beam and all the others. We follow Singh-Miller and Collins (2007) by implementing a cost-sensitive perceptron algorithm which weights hypotheses according to their Smatch f1 score.

5.4 Combined Update Strategy

While early updates are efficient, our oracle is imperfect. To allow the parser to improve over oracle parses, we use a cost-sensitive update whenever a parse has been found whose Smatch f1 score surpasses that of the oracle parse.

6 Experimental Setup

We evaluate our parser[1] on the *proxy* section of the AMR 1.0 corpus (LDC2014T12; Knight et al., 2014). This section consists of newswire texts.

Sentences are tokenised and lemmatised using Stanford NLP (Manning et al., 2014). We use EasyCCG to obtain CCG parses and supertags (Lewis and Steedman, 2014). Token-to-AMR alignments are obtained by combining outputs generated by the JAMR aligner (Flanigan et al., 2014) and the ISI aligner (Pourdamghani et al., 2014), as described by Beschke and Menzel (2018).

First, we induce a CCG/AMR lexicon from the entire *proxy-training* section, delexicalise the entries, and filter for the best derivations using EM, as described in Section 3. We perform 100 iterations of EM. Sentences longer than 100 tokens are filtered out. The resulting lexicon contains 15630 templates and 10504 lexemes.

Next, we extract template tag sequences and train our suppertagger on them. First, tags for the training data are predicted using 5-way jackknifing. Then, a model is trained on the entire training section and used to predict tags for the *dev* and *test* sections of the corpus. Since only templates are predicted that occur in at least two training sentences, a set of 2453 templates is used for prediction. The top-10 recall of the annotated supertags is 96.4% on a randomly chosen held-out portion of the training set.

Finally, the induced lexicon as well as the predicted tag sequences are used to parse the *proxy-test* section of the AMR corpus. We use a beam

[1] For information on reproducing the experiments, see `https://gitlab.com/nats/gramr-ranlp19/`.

System	P	R	F
This paper	0.688	0.423	0.524
Artzi et al. (2015)	0.668	0.657	0.663
Misra and Artzi (2016)	0.681	0.642	0.661
Liu et al. (2018)	-	-	**0.731**

Table 3: Smatch results on the *proxy-test* section of LDC2014T12. Liu et al. (2018) did not report precision and recall in their paper. P stands for *precision*, R for *recall*, F for *f1 score*.

size of 15 during parsing and 20 for finding oracle derivations (see Section 5.2). Parses whose root categories do not match any of the top-10 derivations produced by EasyCCG are dropped from the parser output[2].

The `smatch` tool[3]is used to calculate Smatch precision, recall, and f1 scores for the parser output.

7 Results

We compare our system to two previous CCG-based AMR parsers (Artzi et al., 2015; Misra and Artzi, 2016), as well as the current state of the art in AMR parsing on this data set (Liu et al., 2018). The results are shown in Table 7. The system introduced in this paper achieves comparable precision to the other CCG-based systems, but lower recall.

This gap is largely, but not completely, explained by sentences that were not parsed at all: when unparsed sentences are excluded from the evaluation, our system achieves a precision of 0.701 and a recall of 0.615[4]. Oracle parsing achieves a Smatch precision of 0.886 and an f1 score of 0.706.

The evaluation set contains 823 sentences in total, of which 170 were not parsed, resulting in a coverage of 79%. Of these sentences, 68 were skipped because they were longer than 40 tokens. The remaining 102 are unparsed because the parser failed to find a complete parse.

7.1 Discussion

The parser output in Figure 5 shows some of the most common errors produced by our parser. Firstly, the sequence *International Science and Technology Center* is not recognised as a contiguous named entity. Additionally, *Technology Center* is misrecognised as a country. Both of these issues can be classified as supertagging errors, as they result from the templates chosen from the lexicon. In this specific case, the supertagger's behaviour could likely be improved by adding named entity features to its input. In general, the supertagging task is challenging, especially in the case of function words, which tend to be highly polysemous.

Additionally, the scopes of *and* and *of* are inverted. This can be interpreted as a weakness of the parsing model, which misjudges the probability of the respective scope assignments. Although one would hope for a semantic parser to improve precisely upon these semantically informed syntactic decisions, this behaviour is perhaps to be expected given that we train a sparse linear model with a relatively small amount of training data. Replacing the linear classifier with a neural model that computes embeddings of graph meanings, such as the architecture proposed by (Misra and Artzi, 2016), could improve the parser's judgment.

8 Conclusion

We have introduced a pipeline for training a CCG parser which jointly models syntax and semantics. A central element of our architecture are efforts to reduce the lexicon size. With 2453 delexicalised templates, our parser uses a relatively small lexicon despite the templates being induced automatically. We employ a semantic construction mechanism that is less powerful with λ-calculus, but still achieve competitive precision.

Future directions in this line of work could include applications that make use of the system's transparency, such as the interactive training of parsers without gold-standard annotations, or the application of external constraints such as contextual knowledge to the parser.

Acknowledgments

The author would like to thank Wolfgang Menzel for discussions and advice, as well as the anonymous reviewers.

[2] This restriction was included because the parser tended to favour interpretations of sentences as *NP* instead of *S*.

[3] `https://github.com/snowblink14/smatch`, revision `ad7e655`

[4] The precision improves when unparsed sentences are excluded because the `smatch` tool does not permit empty AMRs to be specified. Unparsed sentences are therefore represented by single-node placeholder AMRs, which are penalised in terms of precision.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1699–1710. http://aclweb.org/anthology/D15-1198.

J. K. Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America* 65(S1):S132–S132. https://doi.org/10.1121/1.2017061.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Sebastian Beschke and Wolfgang Menzel. 2018. Graph algebraic combinatory categorial grammar. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, New Orleans, Louisiana, pages 54–64. https://doi.org/10.18653/v1/S18-2006.

Austin Blodgett and Nathan Schneider. 2019. An improved approach for semantic graph composition with CCG. In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*. Association for Computational Linguistics, Gothenburg, Sweden, pages 55–70. https://www.aclweb.org/anthology/W19-0405.

Shu Cai and Kevin Knight. 2013. Smatch: an Evaluation Metric for Semantic Feature Structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 748–752. http://www.aclweb.org/anthology/P13-2131.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 111–118. https://doi.org/10.3115/1218955.1218970.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1426–1436. http://www.aclweb.org/anthology/P14-1134.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. AMR dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 1831–1841. http://aclweb.org/anthology/P18-1170.

Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. *Abstract Meaning Representation (AMR) Annotation Release 1.0 LDC2014T12*. Linguistic Data Consortium, Philadelphia. https://catalog.ldc.upenn.edu/LDC2014T12.

Alexander Koller. 2015. Semantic construction with graph grammars. In *Proceedings of the 11th International Conference on Computational Semantics*. Association for Computational Linguistics, London, UK, pages 228–238. http://www.aclweb.org/anthology/W15-0127.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1188–1198. https://doi.org/10.3115/v1/P14-1112.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 1512–1523. http://dl.acm.org/citation.cfm?id=2145593.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG Parsing and Semantic Role Labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1444–1454. http://aclweb.org/anthology/D15-1169.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 990–1000. http://www.aclweb.org/anthology/D14-1107.

Yijia Liu, Wanxiang Che, Bo Zheng, Bing Qin, and Ting Liu. 2018. An AMR Aligner Tuned by Transition-based Parser. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 2422–2430. https://www.aclweb.org/anthology/D18-1264.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Kumar Dipendra Misra and Yoav Artzi. 2016. Neural Shift-Reduce CCG Semantic Parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1775–1786. http://aclweb.org/anthology/D16-1183.

Xiaochang Peng, Daniel Gildea, and Giorgio Satta. 2018. AMR Parsing with Cache Transition Systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, New Orleans, USA. https://www.cs.rochester.edu/u/gildea/pubs/peng-aaai18.pdf.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English Strings with Abstract Meaning Representation Graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 425–429. http://www.aclweb.org/anthology/D14-1048.

Natasha Singh-Miller and Michael Collins. 2007. Trigger-Based Language Modeling Using a Loss-Sensitive Perceptron Algorithm. *IEEE ICASSP* .

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Matthew D Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* .

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019. AMR Parsing as Sequence-to-Graph Transduction. https://arxiv.org/abs/1905.08704.

Quasi Bidirectional Encoder Representations from Transformers for Word Sense Disambiguation

Michele Bevilacqua and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
{bevilacqua,navigli}@di.uniroma1.it

Abstract

While contextualized embeddings have produced performance breakthroughs in many Natural Language Processing (NLP) tasks, Word Sense Disambiguation (WSD) has not benefited from them yet. In this paper, we introduce QBERT, a Transformer-based architecture for contextualized embeddings which makes use of a co-attentive layer to produce more deeply bidirectional representations, better-fitting for the WSD task. As a result, we are able to train a WSD system that beats the state of the art on the concatenation of all evaluation datasets by over 3 points, also outperforming a comparable model using ELMo.

1 Introduction

Word Sense Disambiguation (WSD) is the task of associating a word in context with the right meaning among a finite set of possible senses (Navigli, 2009). Consider the following sentence, in which SERVED is the target word:

(1) The waiter standing near the counter SERVED the revolutionary cause well.

In WordNet (Fellbaum, 1998), the most used English computational lexicon in NLP, the following two senses are associated (among many others) to the verb *to serve*:

1. ***devotion***: *devote (part of) one's life or efforts to, as of countries, institutions, or ideas*;
2. ***food***: *help to some food; help with some food or drink*;

The WSD system, in this case, would be tasked to associate the target word with the correct meaning – i.e. the *devotion* sense.

Currently the best WSD systems are supervised, i.e. they leverage annotated corpora as training data (Yuan et al., 2016; Vial et al., 2018; Melacci et al., 2018). However, data labeling is a bottleneck for WSD, even more so than in other fields of NLP. Semantic annotation is a costly process, requiring expert annotators (Taghipour and Ng, 2015; Pasini and Navigli, 2017). If we consider that neural networks, the best performing approach in virtually every task in NLP, are particularly data-hungry, it appears unlikely that there will be much progress in WSD unless either more data is available, or less data is needed.

Between the two directions, we believe efforts towards the latter will prove more fruitful, firstly, because of scalability considerations, and secondly, and more importantly, because of the recent growth in the use of transfer learning, as exemplified by contextualized embeddings. Contextualized embeddings have been shown to produce much better results on downstream tasks compared to end-to-end training, even when less data is provided (Peters et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019; He et al., 2018; Akbik et al., 2018). Contextualized embeddings that use words as tokenization units, such as ELMo (Peters et al., 2018), are most suited to WSD. They are usually trained through self-supervised Causal Language Modeling (CLM) (Lample and Conneau, 2019): given a word sequence $w_1, w_2, \ldots, w_n$ the system has to use w_1 to predict w_2, the sequence $w_{1:2}$ to predict w_3 and so on. CLM is inherently unidirectional, as the model must not be able to "peek" at the word it has to predict. Thus to encode the left and the right contexts two separate networks have to be used, even if they often share part of the weights and are jointly trained.

As regards the use of contextualized embeddings in WSD, this is bound to pose a problem.

Consider the sentence (1) above. It features attractors, i.e. words or phrases pushing the sense interpretation in one direction or the other, with the left context providing a strong cue for the *food* sense and the right for the *devotion* sense. In this paper, we propose a modification of the usual CLM architecture for transfer learning that enables us to train a high-performance WSD system. In this context, we make the following contributions:

- we introduce the BiTransformer, a novel Transformer-based (Vaswani et al., 2017) co-attentive layer allowing deeper bidirectionality;
- we introduce QBERT (Quasi Bidirectional Encoder Representations from Transformers), a novel Transformer-based architecture for CLM making use of the BiTransformer;
- we train a WSD model using QBERT contextualized embeddings, outperforming on the standard evaluation datasets both the previously established state of the art (by a large margin) and a comparable model using ELMo;
- we use QBERT to beat ELMo on the recently established Word-in-Context (WiC) task (Pilehvar and Camacho-Collados, 2019).

2 Related Work

Despite the limited availability of training data, the WSD systems offering the best performances are supervised ones. Many of the approaches are still end-to-end, i.e. they only make use of the information learned during the WSD training.

End-to-end WSD Systems In WSD traditional machine learning techniques are still very competitive because they are not as data-hungry as neural networks. The very popular It Makes Sense (IMS) system (Zhong and Ng, 2010), based on Support Vector Machines and hand-crafted features, performs very well when word embeddings are used as additional features (Iacobacci et al., 2016); the classifier by Papandrea et al. (2017) also gets competitive results. The system of Weissenborn et al. (2015) attains very high performances, but only disambiguates nouns. More recently, neural models have been developed (Kagebäck and Salomonsson, 2016; Uslu et al., 2018; Luo et al., 2018). Some of the most successful offer an intuitive framing of WSD as a tagging task (Raganato et al., 2017a; Vial et al., 2018).

Transfer Learning WSD Systems One of the best performing WSD systems (Yuan et al., 2016) employs a semi-supervised neural architecture, whereby a unidirectional LSTM was trained to predict a masked token on huge amounts of unlabeled data (over 100B tokens). The trained LSTM was used to produce contextualized embeddings for tagged tokens in SemCor; then kNN or a more sophisticated label propagation algorithm was used to predict a sense. The size of the training data makes replication difficult – a reimplementation attempt with a smaller corpus led to worse results (Le et al., 2018). A similar approach using ELMo contextualized embeddings has been presented by Peters et al. (2018), but the results were underwhelming. Another attempt at using transfer learning in WSD has been carried out by Melacci et al. (2018). The authors enhanced IMS with context2vec (Melamud et al., 2016), obtaining performance roughly on a par with Yuan et al. (2016).

Contextualized Embeddings Most of the approaches to contextualized embeddings involve CLM pretraining of directional (either attentive or recurrent) networks. Very successful CLM-based models include ELMo, in which two separate directional LSTMs are fed the output of a shared character-based Convolutional Neural Network (CNN) encoder (Peters et al., 2018), and OpenAI GPT, using Transformers instead of LSTMs and a BPE vocabulary (Sennrich et al., 2016) with regular embeddings instead of the CNN encoder (Radford et al., 2018). Another popular approach, Flair, features character-level LSTMs, outputting hidden states at word boundaries (Akbik et al., 2018). As CLM architectures are normally unidirectional, one alternative in order to guarantee a joint encoding of the context is the Masked Language Modeling (MLM) of BERT (Devlin et al., 2019), which, however, requires a variety of tricks at training time.

3 The QBERT Architecture

Similarly to other LM-based approaches to contextualized embeddings (Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018; Devlin et al., 2019), the architecture we hereby propose has two main components, which we will refer to

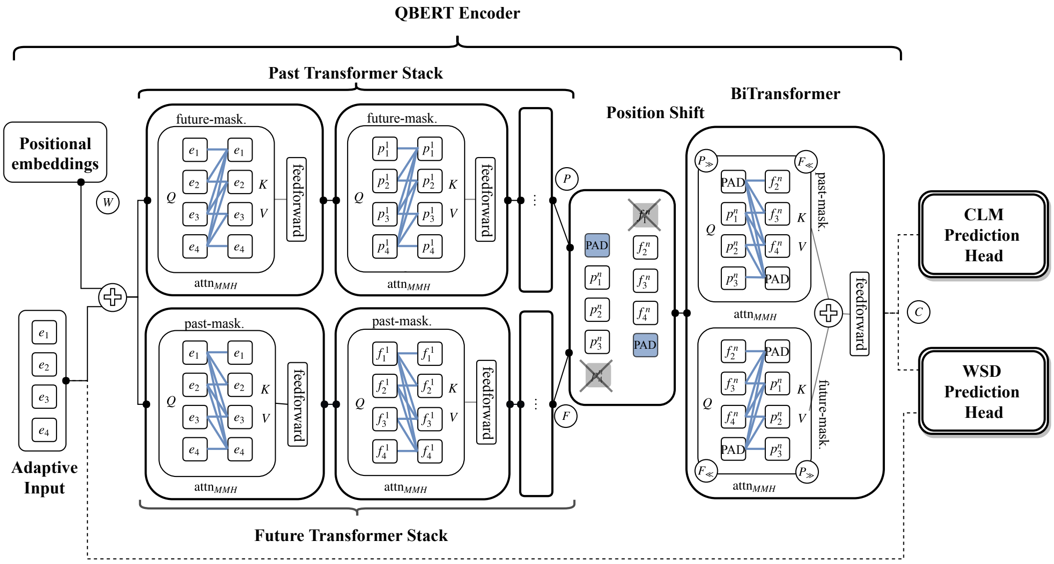


Figure 1: A high-level view of the QBERT architecture.

as Encoder and task-specific Prediction Head. In Figure 1 we show a high-level view of our system. Raw tokens are fed to the encoder, which embeds them into context-independent fixed-length vector representations (the *word embeddings*, W in Figure 1), then uses them to produce context-dependent hidden representations (the *contextualized embeddings*, C), where the context is some subset of the sequence itself. The Prediction Head exploits the vectors produced by the Encoder to perform a task.

3.1 Encoder

As will become clear in what follows, the Encoder of the QBERT architecture is able to compute the hidden representation of a word w_t in a sequence $w_{1:n}$ as a function of the weights and of the whole sequence except w_t itself, i.e. of $w_{1:t-1}$ and $w_{t+1:n}$. To embed tokens, the Encoder uses the Adaptive Input layer (Baevski and Auli, 2018). Sinusoidal positional embeddings are added to the output and passed to two separate stacks of masked Transformers, computing two directional encodings of the sequence, with one (P) having past and present ($w_{1:t}$) information encoded in the present-token hidden vector and the other (F) having present and future ($w_{t:n}$) information instead. Since in the CLM training information about the present token must be hidden from the output layer, we shift and pad the sequences in order to have only the past tokens encoded in the output of the first stack ($P_{\gg}$) and only future tokens encoded in the output of the second ($F_{\ll}$). To combine the shifted sequences we use a novel Transformer layer variant taking them both as input, the BiTransformer, featuring a co-attentive mechanism in which $P_{\gg}$ attends over $F_{\ll}$ and $F_{\ll}$ attends over $P_{\gg}$. The Encoder is trained on CLM using an Adaptive Softmax layer (Grave et al., 2017) as Prediction Head.

3.2 Transformer Variants in QBERT

In the QBERT Encoder we employ three distinct variants of the plain Transformer: the future-masked Transformer, the past-masked Transformer and the BiTransformer. To introduce them we first need to elaborate further into the inner workings of the layer. A vanilla Transformer layer (Vaswani et al., 2017) can be defined as a multi-head self attention submodule followed by a time-wise 2-layer feedforward network, with additional residual connection (He et al., 2016) and layer normalization stabilizing training (Ba et al., 2016).

Core (Self) Attention The intuition behind the attention mechanism is very simple (Bahdanau et al., 2015; Luong et al., 2015). We have a sequence of vectors (the n_q queries Q of dimension d_q) and we want to compute relevance scores against some other sequence of vectors (the n_k keys K of dimension d_k) specific to each couple of vectors q and k. The $n_q \times n_k$ relevance score matrix is then used to compute n_q weighted means of another sequence of vectors (the n_k values V of dimension d_v). So, if we pack Q, K, V into matrices, the mechanism can be distilled into the formula:

$$\text{attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

where $\sqrt{d_k}$ is a normalization factor meant to prevent the dot products from getting too large. In the case of the *self* attention mechanism Q, K, V stand for the same matrix. In the Transformer (Vaswani et al., 2017), multi-head attention is used, in which n attentions (the heads) are computed in parallel, concatenated and then combined through dot product with a $d_q n \times d_o$ matrix W^o. For each attention head h_i, there are three weight matrices W_i^Q, W_i^K, W_i^V, multiplying respectively Q, K and V. Formally, we define multi-head attention (attn_{MH}) as:

$$\text{attn}_{\text{MH}}(Q, K, V) = \bigoplus_{i=0}^{n}[\text{softmax}(\frac{(QW_i^Q)(KW_i^K)^T}{\sqrt{d_k}})(VW_i^Q)]W^o \quad (2)$$

where $\oplus$ denotes concatenation along the second dimension.

Transformer Masking In the past and future stacks, as well as in the BiTransfomer layer, we employ a masking mechanism on attention to enforce directionality, i.e. to force the relevance scores computed between Q and K to be 0 for tokens following or preceding the current one, as needed. Masking can be implemented by performing an elementwise sum between QK^T and a $n_q \times n_k$ masking matrix M, whose values are set to $-\infty$ if K_i and V_j are to be excluded from the attention computation, else 0. In our architecture we employ two different masking matrices: a future masking-matrix M_f which is set to $-\infty$ when $i < j$ and a past-masking matrix M_p set to $-\infty$ when $i > j$; note that $M_f = M_p^T$. Multihead and masked attention can be combined by simply using the same masking matrix in each attention head. We use M_f in the past Transformer stack and M_p in the future stack, producing, respectively, P and F. To encourage the network to encode comparable representations we tie the weights between layers at the same depth on the past and future stack.

3.2.1 Timestep Shift

Present-token information is still encoded in both P and F. To remove it, we use a simple shifting approach where we detach the nth timestep from P and the first from F and add padding to the opposite sides. This effectively shifts the hidden representation by one place to the left and by one place to the right. We refer to the resulting sequences as, respectively, $P_{\gg}$ and $F_{\ll}$. As a result, the ith position of $P_{\gg}$ encodes information from tokens $w_{1:i-1}$ while $F_{\ll}$ encodes $w_{i+1:n}$. Formally:

$$\begin{aligned} P_{\gg} &= PAD \oplus P_{1:n-1} \\ F_{\ll} &= F_{2:n} \oplus PAD \end{aligned} \quad (3)$$

where $\oplus$ denotes concatenation along the timestep dimension. The padding vector PAD is learned during training. The process is visualized in Figure 1, where tokens are aligned according to their shifted positions.

3.2.2 BiTransformer

To combine $P_{\gg}$ and $F_{\ll}$ we employ the BiTransformer, a novel Transformer layer variant that uses a masked coattentive multihead attention mechanism over two sequences. Masking allows $P_{\gg}$ to attend over $F_{\ll}$ while keeping present-token knowledge hidden from the network, and vice versa. This allows deeper bidirectionality in that the resulting output is not a naive combination of two separate directional representations but rather the result of a whole-sequence attention, albeit computed in a two-step process, where the first step can be arbitrarily deep (the masked Transformer stacks) and the second is always shallow (the BiTransformer). Unfortunately, BiTransformers cannot be stacked as each timestep in the output of the layer encodes information about every token in the sequence but the one it has to predict in CLM, so any further use of attention would make pretraining impossible.

The BiTransformer requires modifications to the first part of the vanilla Transformer intramodule architecture. First, both input sequences are layer normalized separately. We compute a masked multihead attention using the future-masked sequence $P_{\gg}$ as Q, the past-masked sequence $F_{\ll}$ as K and V, using the past-masking matrix M_p. To give an insight into what happens, the position i of the n queries, encoding information about words 1 to $i - 1$, is allowed to look at positions i to n of the keys, encoding words $w_{i+1:n}$, $w_{i+2:n}$ and so on. Then we compute the reverse, using $F_{\ll}$, $P_{\gg}$ and future-masking matrix M_f. This process results in two sequences to which input residuals are added, and then added together via a simple elementwise sum. The rest of the layer works just like a regular Transformer layer. We formally describe this coattentive mechanism as follows:

$$\begin{aligned} P'_{\gg} &= \text{LayerNorm}(P_{\gg}) \\ F'_{\ll} &= \text{LayerNorm}(F_{\ll}) \\ O &= \text{attn}_{MMH}(P'_{\gg}, F'_{\ll}, F'_{\ll}, M_p) + \\ &\quad \text{attn}_{MMH}(F'_{\ll}, P'_{\gg}, P'_{\gg}, M_f) + P_{\gg} + F_{\ll} \end{aligned} \tag{4}$$

O goes through the 2-layer feedforward to produce contextualized embeddings, which are used as input for the task-specific Prediction Heads. We describe them in the relevant paragraphs of Sections 4.1 and 5.

4 Experimental Setup

In what follows we first describe the Encoder architecture hyperparameters and CLM pretraining details (Section 4.1). In Section 4.2 we describe the contextualized embeddings systems we use as comparison in the WSD and Word-in-Context experiments. Finally, we report the setup and results of the experiments in Section 5.

4.1 QBERT Encoder Pretraining

CLM Prediction Head and Hyperparameters To train the QBERT Encoder on CLM we use an Adaptive Softmax (Grave et al., 2017) layer as Prediction Head. Following Baevski and Auli (2018), we tie the weights (Press and Wolf, 2017) of the embedding matrices but not the projective weights. Both Adaptive Input and Adaptive Softmax use a vocabulary of 400k words, with cutoffs set to 35k, 100k, 200k and a shrinking factor of 4. The past and future stacks as well as the BiTransformer feature an input and output size of 512, while the first layer of the internal feedforward projects the input to 2048 dimensions, the same as the base configuration in Vaswani et al. (2017). The masked Transformer stacks are both 5-layer deep.

Training Hyperparameters We train QBERT on the English UMBC corpus (Han et al., 2013), which contains around 3B tokens. In our training loop we feed the input in batches of 5000 tokens, splitting the corpus in sequences of max 100 tokens. We found it beneficial to accumulate the gradient for many training steps, performing an update every 16 batches, resulting in a virtual batch size of 80000 tokens. As an optimizer we employ regular Nesterov-accelerated SGD, with a learning rate that first increases linearly from 10^{-5} to 1 during a warmup phase lasting 2000 updates, and then varies from a maximum of 1 to a minimum of 10^{-5} according to a Cyclical Learning Rate (Smith, 2017) policy with cosine scheme, with a period of 2000 updates. With each cycle, the period is multiplied by 1.5 while both the maximum and minimum values are halved. We train until convergence.

We implement the system and training logic in `pytorch` with the help of the `fairseq` library.

4.2 Comparison Systems

In our experiments we compare QBERT with three different contextualized embeddings systems:

1. Off-the-shelf pretrained **ELMo** (Peters et al., 2018). We employ a model featuring 4096-sized bidirectional LSTMs and 512-sized contextualized embeddings[1], pretrained on the concatenation of a Wikipedia dump and a few English monolingual news corpora[2], for a total of 5.5B tokens.

2. Off-the-shelf pretrained **Flair** (Akbik et al., 2018). We employ the models the project's page[3] refers to as `mix-forward` and `mix-backward`, pretrained on "Web, Wikipedia, Subtitles"[4]. We concatenate their contextualized embeddings.

3. **SBERT** (Shallowly Bidirectional Encoder Representations from Transformers), a baseline featuring the same architecture as QBERT but missing the BiTransformer layer: the outputs of the past and future stacks are simply combined through elementwise sum after the position shift.

We do not include in our comparison the BPE-based systems BERT and GPT (Devlin et al., 2019; Radford et al., 2018) as they use a different tokenization unit, which is not suitable for WSD.

[1] The model implementation and weights are available in the `allennlp` library.

[2] The corpora used are the 2008 to 2012 news crawls, available at `http://data.statmt.org/news-crawl/en/`

[3] `https://github.com/zalandoresearch/flair`

[4] There are no further specifications about the composition of the training corpus.

5 Evaluation tasks

As the first and main experiment we train and evaluate a WSD Transformer classifier (Section 5.1.1) using QBERT and comparison contextualized embeddings. To corroborate the results, as further experiments we evaluate the performance of the contextualized embeddings on the Word-in-Context task (Pilehvar and Camacho-Collados, 2019) (Section 5.2).

5.1 Word Sense Disambiguation

5.1.1 Setup

To perform our WSD experiment we train a simple Transformer-based classifier, which we evaluate on all-words WSD benchmark datasets. We use F1 on the test set as a measure of performance.

Architecture Our Transformer classifier takes as input the w-weighted mean between the word embeddings produced by the Encoder (the Adaptive Input layer in the case of QBERT, the character-level CNN in the case of ELMo) and the contextualized embeddings. We freeze the Encoder and only train w and the weights of the Transformer classifier. As Flair has no word embeddings, we concatenate the outputs of the forward and backward models with GloVe embeddings (Pennington et al., 2014), and substitute the weighted mean with a dense layer projecting the concatenated matrix to the Transformer hidden dimension. The classifier produces a probability distribution over an output vocabulary which includes all the possible synsets plus a special `<untagged>` symbol for words with no associated tag. During training only, we treat monosemous words as tagged. At test time, we predict the synset with the highest probability among those associated with the lemma of the target word. Importantly, we do not employ any Most Frequent Sense backoff strategy.

Hyperparameters All models are trained with Adam for a maximum of 60 epochs. We use a similar learning rate scheduling scheme as in the CLM training, first linearly increasing the value from 10^{-5} to 10^{-3}, then using a cosine CLR scheduler with period 200, maximum learning rate 10^{-3} and minimum learning rate 10^{-4}; with each cycle the maximum and minimum are halved, while the period is doubled.

Training and Test Data For each comparison system we train two WSD classifiers, one using only SemCor as training corpus and the other using the concatenation of SemCor and the corpus of WordNet's Tagged Glosses[5] (WTG). WTG includes 117659 manually disambiguated WordNet synset glosses, with 496776 annotated tokens. We test the performance of the models on the English all-words evaluation datasets from the SensEval and SemEval WSD evaluation campaigns, namely Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-07 (Pradhan et al., 2007), SemEval-13 (Navigli et al., 2013), SemEval-15 (Moro and Navigli, 2015) and their concatenation (ALL). We use SemEval-2015 as our development set, to select the best epoch of the run. We use the version of SemCor and the evaluation datasets included in the WSD framework[6] of Raganato et al. (2017b).

5.1.2 Results

We show in Table 1 and Table 2 the results of the evaluation on all-words WSD of the Prediction Head trained on top of QBERT and the comparison systems. Our best model beats all the previously established results on all evaluation datasets. While the performance of the systems using SBERT and ELMo are also very competitive, in many cases exceeding the state of the art, QBERT consistently outperforms them, achieving one of the largest performance gains in years.

SemCor If we restrict the comparison to models trained on SemCor (Table 1), QBERT beats the state of the art with a margin of 0.7 points on Semeval-07 and 1.5 points on SemEval-13. On our development set, SemEval-15, we get a score 2 points over the state of the art. On Semeval-2 and Senseval-3 our F1 score is in the same ballpark as, respectively, Yuan et al. (2016) and Uslu et al. (2018). QBERT also performs well measured against our comparison systems. SBERT achieves lower performance across the board, but attains overall competitive results on all the datasets. ELMo performs on a par with SBERT on the concatenation of all datasets, but gets better results than QBERT on the development set. Flair, perhaps as a result of its purely character-based nature, is severely outperformed on most datasets.

[5] http://wordnetcode.princeton.edu/glosstag.shtml

[6] http://lcl.uniroma1.it/wsdeval/

Systems	Dev. set	S2	S3	S07	S13	S15	ALL
IMS (Melacci et al., 2018)	–	0.702	0.688	0.622	0.653	0.693	0.681
IMSWE (Melacci et al., 2018)	–	0.722	0.699	0.629	0.662	0.719	0.696
IMSC2V$_{+PR}$ (Melacci et al., 2018)	–	**0.738**	0.719	0.633	0.682	0.728	0.713
supWSDEmb (Papandrea et al., 2017)	–	0.727	0.706	0.631	0.668	0.718	–
BiLSTM$_{\text{att+lex}}$ (Raganato et al., 2017a)	S07	0.720	0.694	0.637	0.664	0.724	0.699
GAS$_{\text{ext}}$(concat) (Luo et al., 2018)	S07	0.722	0.705	–	0.672	0.726	0.706
BiLSTM (Vial et al., 2018)	WTG	$0.735_{\dagger}$	$0.709_{\dagger}$	$0.625_{\dagger}$	$0.676_{\dagger}$	$0.716_{\dagger}$	$0.705_{\dagger}$
BiLSTM+VR (ensemble) (Vial et al., 2018)	WTG	0.731	0.706	0.613	0.712	0.716	0.718
LSTM+LP (Yuan et al., 2016)	–	**0.738**	0.718	0.635	0.695	0.726	–
fastSense (Uslu et al., 2018)	S2	0.735	**0.735**	0.624	0.662	0.732	–
SotA (single model)	–	0.735	0.735	0.637	0.695	0.728	0.713
SotA (ensemble)	–	0.735	0.735	0.637	0.712	0.728	0.718
ELMo + WSD Pred. Head	S15	0.719	0.718	0.607	0.703	**0.762**	0.714
Flair + WSD Pred. Head	S15	0.702	0.702	0.615	0.694	0.732	0.699
SBERT + WSD Pred. Head	S15	0.731	0.719	0.640	0.694	0.741	0.715
QBERT + WSD Pred. Head$\star$	S15	0.734	0.732	**0.644**	**0.710**	0.743	**0.724**

Table 1: Results of the evaluation on the English datasets of models trained on SemCor. We include as competitors supervised systems capable of performing all-words WSD on the whole WordNet inventory. We report in the 'Dev set.' column the development corpus used (if any). The † symbol indicates that the result is an average of 20 training runs. **Bold** means that the result is the highest one among non ensemble models. We use $\star$ to mark significant improvement against best single model performance on ALL according to a z-test ($p < 0.05$). We report in the four row blocks 1) competitor SVM-based systems; 2) competitor neural networks; 3) state of the art as the maximum value in the previous rows; 4) QBERT and our comparison systems.

Systems	Dev. set	S2	S3	S07	S13	S15	ALL
BiLSTM (Vial et al., 2018)	SMP	$0.744_{\dagger}$	$0.708_{\dagger}$	$0.625_{\dagger}$	$0.708_{\dagger}$	$0.745_{\dagger}$	$0.719_{\dagger}$
BiLSTM+VR (ensemble) (Vial et al., 2018)	SMP	0.752	0.701	0.668	0.726	0.745	0.727
SotA (single model)	–	0.744	0.735	0.637	0.708	0.745	0.719
SotA (ensemble)	–	0.752	0.735	0.668	0.726	0.745	0.727
ELMo + WSD Pred. Head$\star$	S15	0.743	0.726	0.648	**0.754**	0.786	0.741
Flair + WSD Pred. Head	S15	0.728	0.715	0.646	0.725	0.775	0.725
SBERT + WSD Pred. Head$\star$	S15	0.746	0.722	**0.675**	0.717	0.783	0.734
QBERT + WSD Pred. Head$\star$	S15	**0.757**	**0.739**	0.659	0.746	**0.791**	**0.749**

Table 2: Results of the evaluation on the English datasets of models trained on the concatenation of SemCor and WTG. We use the same notation as in Table 1, employing $\star$ to mark significance against the single model state of the art. Models from Vial et al. (2018), marked by SMP, use a random sample of sentences from SemCor and WTG as development. In the row blocks we report 1) competitor neural networks; 2) the state of the art as the maximum value in the previous rows and in Table 1; 3) QBERT and our comparison systems.

SemCor and WTG When we report in the comparison systems trained on the concatenation of SemCor and WTG (Table 2), QBERT beats the state of the art more consistently and by a larger margin. We reach 1.3 points above the previous state of the art on Senseval-2, 0.4 points on Senseval-3, 2.4 on Semeval-07, 3.8 on Semeval-13 and 4.6 on Semeval-15 (which is however our development set). On the concatenation of all datasets, our margin is of 3 points. Even if we consider the ensemble of 20 models trained on SemCor and WTG by Vial et al. (2018), we get better results on every dataset with the exception of SemEval-07, with a difference of 2.2 points on ALL. With respect to our own comparison systems, QBERT performs better than ELMo, Flair and SBERT in this setting as well. ELMo gets very competitive results compared to the previous state of the art, which it beats on many datasets. Compared to QBERT, however, it gets worse results on almost every dataset, with the single exception of SemEval-13. Flair underperforms also in this setting. SBERT achieves good performances, but still consistently lower than QBERT, except for SemEval-07, which is however a small dataset whose F1 scores show high variance across different training runs.

5.2 Word-in-Context

The Word-in-Context task (WiC) was recently established by Pilehvar and Camacho-Collados (2019). Like WSD, WiC requires identification of a contextually appropriate meaning, but it is framed as a simpler binary classification task: given two contextual occurrences of the same lemma, predict whether the pair shares the same sense. The dataset includes 8320 context pairs, divided between training and development (the test set has not yet been released). By including the same target word in each element of the pair, the dataset is constructed in such a way that context-insensitive word embeddings would not perform better than the random baseline. Thus, the dataset is an ideal evaluation set for assessing the quality of the semantic information encoded in contextualized embeddings.

5.2.1 Setup

Among the baselines offered by Pilehvar and Camacho-Collados (2019), one uses ELMo contextualized embeddings as input to a simple two-layer feed-forward classifier. We replicate the same setting, but using the concatenation of QBERT Encoder word and contextualized embeddings as input instead. Also, as the authors have not yet released the gold keys for the development set and evaluation can only be performed by uploading a prediction file to the Codalab competition page[7], we take $\frac{1}{10}$ of the training instances as our development set, and use the provided development set for testing. We train the system for a max of 40 epochs, submitting the epoch with best accuracy on the development split. WiC's scorer reports the accuracy calculated on the predictions. We implement the same system employing ELMo, Flair (using the concatenation of GloVe and contextualized embeddings) and SBERT as well. Performance is measured by mean accuracy over 5 runs.

5.2.2 Results

In Table 3 we show the results of the evaluation on the WiC development set.

System	Acc. μ	Acc. σ
Elmo (ours)	59.97	1.41
Flair	60.23	0.91
SBERT	60.03	1.13
QBERT	**60.74**	1.22

Table 3: Results of our ELMo, SBERT and QBERT models on the WiC dataset evaluation dataset. We report the mean and standard deviation of the accuracy for 5 runs.

In this setting ELMo performs on a par with SBERT and Flair, while QBERT achieves the best result. Note that the quasi-deeply bidirectional encoding that QBERT can exploit through the BiTransformer might see its effectiveness reduced in this setting since many pairs feature limited context, even as short as 2 or 3 words. Still, the results of the WiC task corroborate those of the all-words WSD, providing evidence that joint encoding is crucial to better performance in word-level semantics.

6 Conclusion

In this paper we showed that the use of contextualized embeddings enables a WSD system to beat the previous state of the art. Moreover, we demonstrated that the use of the BiTransformer coattentive mechanism in the QBERT contextualized embeddings model itself results in even stronger performance. As a result, we attain one of the largest gains in WSD performance in years, with a margin of 3 points over the best reported single model on the concatenation of all datasets, and of 2.2 points over the best ensemble model in the literature. We leave for future work the assessment of whether the gains brought about by the use of the BiTransformer in QBERT carry over to other tasks, helping to bridge the gap between CLM-based and fully bidirectional MLM-based contextualized embeddings. We release the code to train the QBERT Encoder and the WSD classifier, along with pretrained models at https://github.com/mbevila/qbert.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 under the European Union's Horizon 2020 research and innovation programme.

[7] https://competitions.codalab.org/competitions/20010

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proc. of COLING*. pages 1638–1649. https://aclanthology.info/papers/C18-1139/c18-1139.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450. http://arxiv.org/abs/1607.06450.

Alexei Baevski and Michael Auli. 2018. Adaptive input representations for neural language modeling. *CoRR* abs/1809.10853. http://arxiv.org/abs/1809.10853.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*. https://arxiv.org/abs/1409.0473.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL-HLT*. pages 4171–4186. https://aclweb.org/anthology/papers/N/N19/N19-1423/.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proc. of SENSEVAL-2*. pages 1–5. https://www.aclweb.org/anthology/papers/S/S01/S01-1001/.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. In *Proc. of ICML*. pages 1302–1310. http://proceedings.mlr.press/v70/grave17a.html.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_ebiquity-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. pages 44–52. http://aclweb.org/anthology/S/S13/S13-1005.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*. pages 770–778. https://doi.org/10.1109/CVPR.2016.90.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In *Proc. of ACL*. pages 364–369. https://aclanthology.info/papers/P18-2058/p18-2058.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proc. of ACL*. pages 328–339. https://www.aclweb.org/anthology/papers/P/P18/P18-1031/.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proc. of ACL*. pages 897–907. http://aclweb.org/anthology/P/P16/P16-1085.pdf.

Mikael Kågebäck and Hans Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. In *Proc. of COLING*. pages 51–56. https://aclanthology.info/papers/W16-5307/w16-5307.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291. http://arxiv.org/abs/1901.07291.

Minh Le, Marten Postma, Jacopo Urbani, and Piek Vossen. 2018. A Deep Dive into Word Sense Disambiguation with LSTM. In *Proc. of COLING*. Association for Computational Linguistics, pages 354–365. https://aclanthology.info/papers/C18-1030/c18-1030.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018. Incorporating Glosses into Neural Word Sense Disambiguation. In *Proc. of ACL*. Association for Computational Linguistics, pages 2473–2482. https://aclanthology.info/papers/P18-1230/p18-1230.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proc. of EMNLP*. Association for Computational Linguistics, pages 1412–1421. http://aclweb.org/anthology/D/D15/D15-1166.pdf.

Stefano Melacci, Achille Globo, and Leonardo Rigutini. 2018. Enhancing Modern Supervised Word Sense Disambiguation Models by Semantic Lexical Resources. In *Proc. of LREC*. pages 1012–1017. https://www.aclweb.org/anthology/papers/L/L18/L18-1163/.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proc. of COLING*. pages 51–61. http://aclweb.org/anthology/K/K16/K16-1006.pdf.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*. pages 288–297. http://aclweb.org/anthology/S/S15/S15-2049.pdf.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2):10:1–10:69. https://doi.org/10.1145/1459352.1459355.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Proc. of SemEval*. pages 222–231. http://aclweb.org/anthology/S/S13/S13-2040.pdf.

Simone Papandrea, Alessandro Raganato, and Claudio Delli Bovi. 2017. SupWSD: A Flexible Toolkit for Supervised Word Sense Disambiguation. In *Proc. of EMNLP*. pages 103–108. https://aclanthology.info/papers/D17-2018/d17-2018.

Tommaso Pasini and Roberto Navigli. 2017. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. In *Proc. of EMNLP*. pages 78–88. https://aclanthology.info/papers/D17-1008/d17-1008.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proc. of EMNLP*. pages 1532–1543. http://aclweb.org/anthology/D/D14/D14-1162.pdf.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proc. of NAACL-HLT*. pages 2227–2237. https://aclanthology.info/papers/N18-1202/n18-1202.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proc. of NAACL-HLT*. pages 1267–1273. https://aclweb.org/anthology/papers/N/N19/N19-1128/.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task-17: English Lexical Sample, SRL and All Words. In *Proc. of SemEval 2007*. pages 87–92. http://aclweb.org/anthology/S07-1016.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proc. of EACL*. pages 157–163. https://aclanthology.info/papers/E17-2025/e17-2025.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training .

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017a. Neural Sequence Learning Models for Word Sense Disambiguation. In *Proc. of EMNLP*. pages 1156–1167. https://aclanthology.info/papers/D17-1120/d17-1120.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017b. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proc. of ACL*. pages 99–110. https://www.aclweb.org/anthology/papers/E/E17/E17-1010/.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of ACL*. pages 1717–1725. http://aclweb.org/anthology/P/P16/P16-1162.pdf.

Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *Proc. of WACV*. pages 464–472. https://doi.org/10.1109/WACV.2017.58.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proc. of SENSEVAL-3*. pages 41–43. https://aclanthology.info/papers/W04-0811/w04-0811.

Kaveh Taghipour and Hwee Tou Ng. 2015. One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction. In *Proc. of CoNLL*. pages 338–344. http://aclweb.org/anthology/K/K15/K15-1037.pdf.

Tolga Uslu, Alexander Mehler, Daniel Baumartz, Alexander Henlein, and Wahed Hemati. 2018. Fast-Sense: An Efficient Word Sense Disambiguation Classifier. In *Proc. of LREC*. pages 1042–1046. https://www.aclweb.org/anthology/papers/L/L18/L18-1168/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proc. of NIPS*. pages 6000–6010. https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2018. Improving the coverage and the generalization ability of neural word sense disambiguation through hypernymy and hyponymy relationships. *CoRR* abs/1811.00960. http://arxiv.org/abs/1811.00960.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-Objective Optimization for the Joint Disambiguation of Nouns and Named Entities. In *Proc. of ACL*. pages 596–605. http://aclweb.org/anthology/P/P15/P15-1058.pdf.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised Word Sense Disambiguation with Neural Models. In *Proc. of COLING*. pages 1374–1385. http://aclweb.org/anthology/C/C16/C16-1130.pdf.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. In *Proc. of ACL, System Demonstrations*. pages 78–83. http://www.aclweb.org/anthology/P10-4014.

Evaluating the Consistency of Word Embeddings from Small Data

Jelke Bloem♠ **Antske Fokkens**♣ **Aurélie Herbelot**♢
♠ Institute for Logic, Language and Computation. University of Amsterdam
♣ Computational Lexicology and Terminology Lab. Vrije Universiteit Amsterdam
♢ Center for Mind/Brain Sciences / DISI, University of Trento
j.bloem@uva.nl, antske.fokkens@vu.nl, aurelie.herbelot@unitn.it

Abstract

In this work, we address the evaluation of distributional semantic models trained on smaller, domain-specific texts, particularly philosophical text. Specifically, we inspect the behaviour of models using a pre-trained background space in learning. We propose a measure of *consistency* which can be used as an evaluation metric when no in-domain gold-standard data is available. This measure simply computes the ability of a model to learn similar embeddings from different parts of some homogeneous data. We show that in spite of being a simple evaluation, consistency actually depends on various combinations of factors, including the nature of the data itself, the model used to train the semantic space, and the frequency of the learned terms, both in the background space and in the in-domain data of interest.

1 Introduction

Distributional semantic (DS) models (Turney and Pantel, 2010; Erk, 2012; Clark, 2015) typically require very large corpora to construct accurate meaning representations of words (Bengio et al., 2003). This big data methodology presents challenges when working with text in a specific domain or a low-resource language. In this paper, we are interested in modeling concepts in philosophical corpora, which are far smaller than a typical web corpus. Instead of training directly on the philosophical in-domain data, which is too sparse for learning, we rely on a pre-trained background semantic space, thus simulating a speaker with some linguistic knowledge coming to a new domain.

Our focus is the evaluation problem encountered when working with domain-specific data. DS models are typically evaluated on gold standard datasets containing word association scores elicited from human subjects (e.g. Bruni et al., 2014; Hill et al., 2015). Beside the limited practical use of such evaluation metrics (e.g. Gladkova and Drozd, 2016), this is not a feasible method for evaluating DS models in low-resource situations. When domain-specific terminology is used and the meaning of words possibly deviate from their most dominant sense, creating regular evaluation resources can require significant time investment from domain experts. Evaluation metrics that do not depend on such resources are valuable. Thus, we introduce the metric of **consistency**, which requires a model to learn similar word embeddings for a given term across similar sources, for example, two halves of a book.

Philosophical texts make a suitable case study for out-of-domain data, as words may have very different meanings in philosophy than in general usage. For example, while a *proposition* is synonymous for *offer* or *proposal* in ordinary language, in philosophy it is, among other things, a bearer of truth-value (McGrath and Frank, 2018). Furthermore, philosophical writing is often precise and terminology tends to be defined or at least discussed in the text, so there should be enough information for modeling meaning even when working with small data, for instance in one or multiple works by a particular philosopher or from a particular philosophical tradition. Last but not least, the field of philosophy could benefit from this type of modeling — although philosophers have not yet made broad use of computational methods (Betti et al., 2019), it has been shown that new insights can be obtained using an information retrieval tool based on a distributional semantic model of digitalized philosophical texts (Ginammi et al., in press).

Using philosophical data, we perform a battery of tests which reveal interesting properties of consistency. We show that in spite of being a simple evaluation, consistency actually depends on various combinations of factors, including the nature of the data itself, the model used to train the semantic space, and the frequency of the learned terms, both in the background space and in the in-domain data of interest. This leads us to conclude that the evaluation of in-domain word embeddings from small data has to be controlled extremely carefully in order not to draw incorrect conclusions from experimental results.

2 Related Work

Learning embeddings for rare words is a very challenging process (Luong et al., 2013). Word2Vec (W2V, Mikolov et al., 2013a)'s skipgram model can learn embeddings from tiny data after modification, as shown by Herbelot and Baroni (2017) when it consists of just a single highly informative definitional sentence. However, philosophical data is typically *small data* rather than tiny data. While tiny data consists of a single definitional sentence, our small data consists of multiple context sentences per term that are not necessarily definitional. Herbelot and Baroni's (2017) Nonce2Vec (N2V) has not been tested on this type of data. W2V has been tested on smaller datasets, but was found to be suboptimal (Asr et al., 2016) and surpassed by SVD models on a 1 million word dataset (Sahlgren and Lenci, 2016).

Different DS evaluations test different aspects of the learned embeddings (i.e. Wang et al., 2019). Most existing methods are however not easily applicable to our task. The typical evaluation of comparing embedding similarities to a gold standard of word similarity scores, such as the SimLex-999 dataset (Hill et al., 2015) cannot be applied, because we are interested in the representation of specific terms: even if these terms are present in the evaluation set, their meaning in the philosophical domain is likely to differ. Manually creating a domain-specific resource requires labor-intensive effort by domain experts, which makes it impractical to port standard datasets to a specific type of corpora. This holds also for other evaluation methods such as analogy scores (Mikolov et al., 2013b), as well as coherence (Schnabel et al., 2015), which is based on the idea that pairs of similar words should be close in semantic space.

Methods where human raters directly respond to output of the model, such as comparative intrinsic evaluation (Schnabel et al., 2015) are interesting, but require domain experts, as well as instructions that elicit the desired type of semantic relation (i.e. similarity). Extrinsic evaluation requires a downstream task that can be evaluated, but in this use case we are interested in the information encoded by the DS model itself. QVEC (Tsvetkov et al., 2015) evaluates by aligning dimensions of a semantic space to linguistic features, but we are interested only in evaluating some vectors rather than an entire space (target term vectors but not the background space vectors), and this approach requires language-specific resources.

Nooralahzadeh et al. (2018) evaluate domain-specific embeddings by building a query inventory for their domain from a glossary containing synonym, antonym and alternative form information. Unfortunately, such structured glossaries are generally not available for specific philosophers. Hellrich and Hahn (2016) test their models for reliability in a study investigating historical English and German texts, another relatively low-resource domain. Their reliability metric involves training three identically parametrized models, and comparing the nearest neighbors of each word in each model using a modified Jaccard coefficient. This metric does not require any language-specific data, but it mainly serves as a test of the impact of the sources of randomness in Word2Vec, and not as a measure of the systematic semantic differences across various data sources.

3 Consistency Metric

We propose **consistency** as a useful metric to evaluate word embeddings in the absence of domain-specific evaluation datasets. We consider a model to be consistent if its output does not vary when its input should not trigger variation (i.e. because it is sampled from the same text). Thus, a model can only be as consistent as the input data it is trained on and it requires the experimenter to compute data consistency in addition to vector space consistency.

To evaluate *data consistency*, we create vectors for target terms in a domain corpus under two conditions: a) random sampling; b) equal split. The 'equal split' condition simply corresponds to splitting the data in the middle, thus obtaining two subcorpora of equal size and in diachronic order. Given a pre-trained background space kept frozen across experiments, the vector representation of a target is generated by simple vector addition over its context words. Therefore, the obtained vector directly represents the context the target term occurs in, and consequently, similar representations (in terms of cosine similarity) mean that the target term is used in a similar way in different parts of a book/corpus, and is thus consistently learnable. Crucially, though, this measure may interact with data size. Kabbach et al. (2019) recently noted a *sum effect* in the additive model, where summed vectors are close to each other. It may be the case that additive model vectors summed over more context data contain more information and may have higher similarity between each other, resulting in higher consistency scores. We test this in Section 6. When randomly sampling, we limit the number of sentences per sample to control for this.

To evaluate *space consistency*, we create identically parametrized models as in Hellrich and Hahn's (2016) reliability metric, but over different parts of the data, with the data being split in the middle, as just described. We consider two ways of comparing two vectors $\vec{a_1}$ and $\vec{a_2}$: by similarity, where a higher cosine similarity indicates more consistency, or by nearest neighbor rank, where a higher rank of $\vec{a_1}$ among the nearest neighbors of $\vec{a_2}$ indicates more consistency. Every vector in the background space, as well as $\vec{a_2}$, is ranked by cosine similarity to $\vec{a_1}$ to compute this rank value.

Although it is more complex than having a single metric, we must consider both rank and similarity simultaneously: rank is a more relative metric and helps to ground the similarity value in the local context of the target term. A vector with 0.8 similarity but lower rank is a worse result than a vector with 0.8 similarity and a high rank, as the low rank means that the vectors are in a dense part of the semantic space and a very high similarity is required to consistently identify which of the neighbouring vectors refers to the same concept. Conversely, a low-similarity, high-rank vector can be a cause for scepticism, as it may have been placed far out from the rest of the semantic space.

We take consistency to be a desirable property of word embeddings at the level of a certain domain. Of course, consistency only measures one specific desirable property of embeddings and should thus not be interpreted as a general quality or accuracy score. But even taken on its own, we will show that it exhibits complex behavior with respect to data, background vectors and term frequency.

4 Task Description

Our overall aim is to obtain consistent embeddings of terms central to the works of Willard Van Orman Quine (1908-2000), an influential 20th century philosopher and logician. As the meaning of terms may differ between authors and even between books by the same author, we need to learn such embeddings from small data, bounded by the occurrences of the term in one particular book.

Quine datasets We build novel datasets based on a corpus of 228 philosophical articles, books and bundles written by Quine, with a focus om two of Quine's books: *A System of Logistic* (Quine, 1934) and *Word & Object* (Quine, 1960). These Quine texts are part of a larger corpus of philosophical texts, which is still being compiled, that are central to the history of scientific ideas (Betti and van den Berg, 2016). We focus on these particular works from the corpus because testing consistency is best done on homogeneous data, and our philosophy domain experts informed us that Quine was a remarkably stable philosopher in his outlook (Betti and Oortwijn, p.c.).

The first book is a formula-heavy logic book, deviating strongly from ordinary language. Such a technical book is particularly likely to be internally consistent. It contains 80,279 tokens after tokenization and manual replacement of formulas with special tokens. The second book is more textual and consists of standard philosophical argumentation. Our domain experts consider it conceptually consistent. It contains 133,240 tokens after tokenization. The full corpus of the 228 Quine articles contains 1.7 million tokens and is pre-processed with NLTK (Bird et al., 2009) for sentence splitting and tokenization. A less common preprocessing step we took was to remove one-character tokens from the texts. These works contain many one-letter variable names, logical symbols and other formal language that a model might otherwise use to position vectors of Quine terminology in particular areas of the semantic space, as these tokens are highly infrequent in the general domain.

To obtain terms that would be relevant to model, we automatically extract terms from the two books' indexes, as the most important terminology is likely to be listed there. We include multi-word terms, and divide the terms into 70%/30% subsets for training and testing, resulting in a 157 / 67 split for *Logistic* and a 184 / 79 split for *Word & Ob-*

ject. The target terms thus differ per book, as each book lists different terms in its index. Instead of this automatic approach to obtaining target terms, an expert-created resource could provide a better set of target terms, if available. If neither this nor a terms glossary or index of terms is available, keyword extraction methods could be used as an alternative way of obtaining terms for evaluation. In cases where the model will not be used for any analysis of domain-specific content downstream, it may be sufficient to randomly select words from the text as target terms.

Next, we derive datasets from this corpus using our two conditions for data consistency: random sampling and equal split. In *random sampling*, for each target term that meets a frequency cutoff of 10, we randomly select five non-overlapping samples of up to 10 random sentences that contain the target term, divided evenly across the samples if the term occurs in fewer than 50 sentences. This gives us the datasets *Quine-WordObject-rnd* (with Word & Object core terms as target terms), *Quine-Logistic-rnd* (with System of Logistic core terms) for our two books of interest, and *Quine-all-rnd* sampled from the full Quine corpus, where we also use the Word & Object core terms as target terms.[1] In the *equal split* condition, we divide a book into two halves at a chapter boundary, and extract all sentences containing index terms that meet a frequency cuf-off of 2 in each half, resulting in the datasets *Quine-WordObject* and *Quine-Logistic*. With random sampling, we intend to capture the basic consistency of the model. With equal split, we aim to capture consistency across potential meaning development throughout the book.[2]

Wikipedia dataset For cross-domain comparison, we apply our method to a 140M word preprocessed Wikipedia snapshot using the same random sampling process. As target terms, we used 300 randomly sampled one-word Wikipedia page titles, following Herbelot and Baroni (2017).

5 Method

Before evaluating whether we have *space consistency*, we must establish to what extent we have *data consistency*, following our argumentation in Section 3. To obtain an embedding for a new target term, we use an additive model over its context words, using as background space ordinary language representations. For the in-domain context, we use a window size of 15, with the window being restricted to the sentence. The background space is based on a Wikipedia snapshot of 1.6*B* words trained with Word2Vec's Gensim implementation with default parameters, and containing 259,376 word vectors in 100 dimensions. For each target term, context words undergo subsampling, which randomly drops higher-frequency words.[3] The vectors of the remaining context words are summed to create a vector for the target term. This additive model was used by Lazaridou et al. (2017) for their textual data, and was shown by Herbelot and Baroni (2017) to work reasonably well on tiny data. We calculate the vectors separately per sample (or book half), yielding comparable term vectors.

Next, we turn to *space consistency*. We use our consistency metric to evaluate two models that are suited to learning embeddings from small data: Nonce2Vec (Herbelot and Baroni, 2017) and an SVD-reduced count-based model over concatenations of our datasets with general-domain data.

The first model, Nonce2Vec, modifies W2V's 'skip-gram' model (Mikolov et al., 2013a) in a way that is inspired by *fast mapping* (Carey and Bartlett, 1978) in humans. Human learners can acquire new words from just a single token and this process of fast mapping appears to build on concepts that are already known (Coutanche and Thompson-Schill, 2014). Nonce2Vec models this through incremental learning, an initial high learning rate, greedy processing, and parameter decay. To simulate the existence of background knowledge, Nonce2Vec maps its novel word vectors into a previously learned semantic space, based on the aforementioned Wikipedia snapshot and the same subsampling procedure. Target term vectors are initialized to their sum vector from the additive model. For each sentence, the model is trained on the target term, only updating the weights for that term and freezing all other network parameters. The learning rate and context window size decay in proportion to the number of times the target term has been seen, and the subsampling rate increases per sentence.

Secondly, we try a count-based approach, creat-

[1] Word & Object touches upon much of Quine's work, so its terminology can be considered representative.

[2] While our datasets are derived from copyrighted works and cannot be shared, we provide replication instructions, term lists and code here: https://bloemj.github.io/quine2vec/

[3] Some promising alternative subsampling methods for tiny data were recently discussed by Kabbach et al. (2019).

Dataset	cos-sim	rank
Quine-WordObject	0.938	1
Quine-Logistic	0.907	22.4
Quine-WordObject-rnd	0.919	1
Quine-Logistic-rnd	0.935	1
Quine-all-rnd	0.953	1
Wiki-rnd	0.927	1.001

Table 1: Consistency metrics on different data sets using the additive model.

n	cos-sim
1	0.794
2	0.837
3	0.905
4	0.923
8	0.956
all	0.987

Table 2: Data consistency for the term *analytical hypotheses* in Word & Object when varying the number of sentences per sample n.

ing vectors over the general-domain and in-domain data at the same time. In this procedure, we concatenate a particular Quine dataset with a 140M word Wikipedia corpus sample, in which the Quine terms are marked with special tokens in order to be trained separately from the same term in the Wikipedia data. We create embeddings from this corpus, applying PPMI weighting and singular value decomposition (SVD) to reduce the models to 100 dimensions, to match the dimensionality of our other models and because factorized count models have been shown to work well on smaller datasets (Sahlgren and Lenci, 2016). We use the *Hyperwords* implementation of Levy et al. (2015), with a window size of 5, and other hyperparameters set to the default values.

In both the above approaches, we can then compute vector space consistency between different vectors learned for the same term over different splits of the data.

6 Consistency of Data

We start by applying the additive model to quantify data consistency on the different datasets described in Section 4. We compute average similarities and nearest neighbor ranks over the vectors of all target terms in a dataset. For the randomly sampled data sets, we have five vectors per term, one from each sample, and compute the metrics over all unique combinations of 2 vectors. For the equal split setting, we compare the term vectors summed over each half of the book.

The additive model produces highly consistent embeddings on the training data: for most terms, the vectors summed over each book half are each other's nearest neighbors in the background space. This trend is also observed for the test sets presented in Table 1, where we observe high consistency for the embeddings from both books.

Using the book halves of System of Logistic (*Quine-Logistic*) gives us a slightly lower data consistency score than random sampling from that book (*Quine-Logistic-rnd*), possibly because the meaning of a term may evolve from the first half to the second half of a book. This suggests some utility of the data consistency measure in quantifying meaning development throughout a text, as long as other factors are controlled for. We also see that the Wikipedia domain data (*Wiki-rnd*) is less consistent than the Quine domain data (*Quine-all-rnd*), which is to be expected as it contains more diverse text.

These results seem to indicate that the additive model provides consistent embeddings. This means that it must be possible to learn consistent embeddings from these datasets, at least up to the consistency values reported here, as the additive model directly represents the contexts that predictive models use for training.

As already mentioned, however, the factor of data size may interfere with consistency. We do observe in Table 1 that the consistency of data sampled across the full Quine corpus is higher. Although we limited our samples to 10 sentences per term, not every core Quine term is used frequently enough to have 5 samples with the maximum size of 10 sentences. Specifically, in the full Quine dataset, 68.6% of terms reach the maximum size, while in the Word & Object dataset, only 32.1% of terms reach it. In the Wiki set, this is 90.9%, showing that its lower consistency is not explained by limited data. To fully control for data size, we would need to use artificial data: if we control for the number of sentences, the number of words and the number of words subsampled still affect data size. As we are mainly interested in the quality of models on our own philosophical corpus, we leave this for future work.

Instead, we test the effect of data size by summing two vectors for the same term over varying numbers of sentences, and computing the consistency between them. Table 2 shows a clear effect of data size: vectors summed over more sentences have higher data consistency. This shows that data consistency should ideally be computed within the constraints of a particular data size, because vectors summed over more context are more informative and thus more consistent.

7 Consistent Spaces

Having established that our data is consistent even with fairly small samples, we proceed to use two small data approaches to place terms consistently in vector space. We start with Nonce2Vec (N2V), which uses the sum vectors from the additive model for initialization and trains only on that vector, as it does not update the vectors in the background space, only that of the target term.

For this experiment, we modified N2V in two ways. Firstly, it now takes multiple sets of input sentences per target term, one from each sample or book half, and trains each term on all sets separately, resulting in multiple term vectors, one over each sample. Secondly, we implemented the consistency metrics described in Section 3 for comparing the different sample vectors and analyzing their position in the background space.

Using N2V's default parameters, we obtain low consistency scores. While N2V was designed for learning from a dataset with one sentence per term, the terms in our dataset occur in more sentences. A likely consequence of this difference, having small data instead of tiny data, is that the default parameters may include a too high learning rate and N2V's parameter decay may be too fast. A learning rate that is too high can result in models with low stability. To adapt to small data, we tune N2V's parameters on the full Quine dataset with the training set of target terms. We performed a grid search following a parameter space containing different learning rates ($[0.1, 0.5, \mathbf{1}, 1.5]$), the number of negative samples ($[1, \mathbf{3}, 5]$), the subsampling rate ($[100, 3000, 5000, \mathbf{10000}, 20000]$), learning rate decay ($[30, \mathbf{70}, 100, 180]$), subsampling rate decay ($[1.0, 1.3, \mathbf{1.9}, 2.5]$) window decay ($[1, 3, \mathbf{5}]$), window size ($[\mathbf{15}]$). Bold values are the best performing values in Herbelot and Baroni (2017) or defaults of N2V. We obtain our best performance with a learning rate of 0.1, 5 negative samples, a learning rate decay of 30 and a subsampling decay factor of 2.5.

We obtain fairly consistent embeddings with these parameters on the test set, as shown in Table 3: the vectors learned from the two book halves in the *Quine-WordObject* and *Quine-Logistic* datasets are often each other's nearest neighbour, with average nearest neighbour ranks of 1.21 and 1.48, respectively. Surprisingly, although this model is initialized using Wikipedia background vectors, that domain (*Wiki-rnd*) fares the worst in terms of consistency, as it does in the additive model. In general, these vector space consistency scores are lower than the data consistency scores we saw before, so there is room for improvement.

Dataset	cos-sim	rank
Quine-WordObject	0.686	1.21
Quine-Logistic	0.748	1.48
Quine-WordObject-rnd	0.695	2.11
Quine-Logistic-rnd	0.743	1
Quine-all-rnd	0.717	1.59
Wiki-rnd	0.589	507.8

Table 3: Consistency metrics on different data sets for the Nonce2Vec-based models.

Dataset	cos-sim
Quine-WordObject-rnd	0.352
Quine-Logistic-rnd	0.436
Quine-all-rnd	0.440
Wiki-rnd	0.321

Table 4: Consistency on different data sets for the SVD models.

We therefore turn to our other approach that is not based on the additive model: the SVD models over the concatenation of in-domain and general-domain data. When concatenating the datasets, we have to ensure that the target terms in our random samples of in-domain data are trained separately from the same term in the general domain and in other samples. We therefore mark them with a different ID for each sample. As before, we compute cosine similarities between these target terms from different samples to measure consistency.

Table 4 shows that the resulting embeddings are not very consistent, with much lower average cosine similarities between the samples that does not reflect the consistency of the data, as indicated by the additive model in Table 1. The consistency of the SVD vectors is also lower than that of the Nonce2Vec vectors from the previous experiment.

One possible explanation for the difficulty that both of these models have in learning from our data is in the bridging of the domain gap between the

Group of terms	similarity
System of Logistic	0.323
Word & Object	0.366
Q-High-freq W-Low-freq	0.735
Q-Low-freq W-Low-freq	0.417
Q-Low-freq W-High-freq	0.109
Q-High-freq W-High-freq	0.078

Table 5: Average similarities between Quine vectors and Wiki vectors in our SVD model. Q = Quine, W = Wiki.

Wikipedia general-domain spaces and the Quine terminology. To quantify the difference between domains, we selected all sentences from the Quine corpus containing 35 target terms and concatenated them with our 140M word Wikipedia sample, as in the previous experiment. These terms were selected to be either high-frequent or low-frequent in the Quine domain and either high-frequent or low-frequent in the general domain. Again, the Quine terms were marked in order to be trained separately from the same term in the Wikipedia domain, and we created a SVD model. In this SVD model, we computed the cosine similarities between each Quine term and its Wikipedia counterpart, and take this to be a measure of domain difference.

Table 5 shows a clear effect of term frequency. We grouped all terms according to two factors: their frequency in the Quine book they were selected for (low, relative frequency[4] < 0.0005 or high, relative frequency > 0.001) and their frequency in the Wikipedia domain (low, RF < 0.000025 or high, RF > 0.00005).[5] We observe that infrequent terms with a dominant philosophical sense such as *stimulus* have more similar vectors in both domains despite their sparsity in both corpora. Generally, terms that are highly frequent in the Quine-domain but have low frequency in the Wikipedia domain are more similar between the two domains (*Q-High-freq W-Low-freq*). To a lesser extent, this is also true for terms that are low-frequent in both domains.

This result indicates that bridging the domain gap should be easier with these philosophical core terms than with frequent Wikipedia terms. The fact that our models are less consistent on Wikipedia data also indicates that the generality of this domain is more relevant than any specific differences with the Quine domain. It must therefore be possible to learn good representations from this data by using background knowledge from the Wikipedia domain, but the models we tested did not reach the level of consistency of the additive model.

Dataset	cos-sim	rank
Quine-WordObject-rnd	0.352	22,947
Quine-Logistic-rnd	0.353	24,513
Quine-all-rnd	0.382	17,262
Wiki-rnd	0.475	2,902

Table 6: Average similarities between learned in-domain term vectors and pretrained general-domain background vector on different data sets for the Nonce2Vec-based models.

For better or for worse, our models do move away from what is in the background space. In our Nonce2Vec experiment on the *Quine-all-rnd* dataset, we also measured the average cosine similarity and nearest neighbour rank of the pretrained Word2Vec term vector from the background space, compared to the vectors we learned for that same term from the in-domain data. These numbers, shown in Table 6, reveal that the model does not stay close to the pre-trained background vectors in order to achieve high consistency, which could be a risk if consistency was used as a learning signal in combination with an invariant initialization. Furthermore, the vectors learned from the Wiki data are closer to the pre-trained vectors than those learned from the Quine data. This is expected of a good model, as there is no domain gap to bridge when training with Wikipedia context sentences into a Wikipedia background space. This also means that the vector representations for terms as used by Quine become more distinct after training, as our philosophy domain experts would expect of a good meaning representation of these in-domain terms.

We must again note that consistency is not the only desirable property of word embeddings. Unfortunately, other properties are more difficult to evaluate on low-resource data. Without a domain-specific evaluation set, we can only explore issues with quality by examining nearest neighbors of vectors that our metric marks as perfectly consistent. We observe both in our results, illustrated by cherry-picked examples from the Nonce2Vec model on the *Quine-WordObject* dataset. Table 7 shows that the nearest neighbours for both book half vectors for the term *talking* (*Word & Object*) look bad. The vectors' nearest neighbours are some

[4] $\frac{F}{C}$ where F is the term frequency and C is the corpus size.

[5] Different thresholds are necessary for the larger corpus.

Term		$\vec{a_1}$ NNs	$\vec{a_2}$ NNs
talking	1	wrongfulness	axiomatically
	2	axiomatically	epiphenomenon
	3	particularized	impredicative
verbs	1	logophoric	logophoric
	2	deverbal	resumptive
	3	adpositions	countability
	4	uninflected	adverbials

Table 7: Qualitative examination of some nearest neighbours of target term vectors computed over book halves 1 and 2 of *Word & Object*.

apparently unrelated words yet they are closest to each other (similarity 0.751). We thus have high consistency, but not a good semantic representation. The word *verb* is an example that does work: all nearest neighbours from the background space are linguistic terms. The two *verbs* vectors are also closest to each other (similarity 0.625).

8 Conclusion

Our results show that it is possible to learn consistent embeddings from small data in the context of a low-resource domain, as such data provides consistent contexts to learn from. Applying an additive model that sums general-domain vectors from a pre-trained background space resulted in similar vectors for the same terms across different contexts from the same domain. The Nonce2Vec model also results in consistent embeddings that are closer to vectors of the same term trained on different context sentences than to vectors of other terms. The summed vectors from the additive model applied to our philosophical small data are highly discriminative, distinguishing the target terms from background terms almost perfectly.

Our results show the benefits of using consistency as an intrinsic evaluation metric for distributional semantic models, particularly for low-resource situations in which no gold standard similarity scores are available. While the metric may appear simple, it proved useful both for evaluating the homogeneity of a dataset and for evaluating the stability of vector spaces generated by a given model. Consistency turns out to depend on various combinations of factors, including the nature of the data itself, the model used to train the semantic space, and the frequency of the learned terms, both in the background space and in the in-domain data of interest.

For the specific purpose of modeling philosophical terminology, consistency helps us assess the quality of embeddings for philosophical terms, which may differ in meaning across a book or an author's work, and for which no gold standard evaluation sets are available. These embeddings can then be used to aid in the examination of large volumes of philosophical text (Ginammi et al., in press). Beyond our use case, the consistency metric is quite broadly applicable — a relevant background semantic space is necessary, but this can be constructed from out-of-domain data.

Like any metric, the consistency metric does not answer all of our questions about the quality of our embeddings. Although the additive model is more consistent than the others, both its dependence on data size and the not-always-great qualitative results show that exploring other models is worthwhile for small data. Further research is required to determine whether the representations produced by the additive model are useful for downstream tasks. Using the knowledge of domain experts in a structured evaluation task would be a good, though resource-intensive, next step. Our metric helps quantify the reliability of a model before investing more resources into evaluation.

Our observation that the consistency metric depends on a variety of other factors shows that consistency is a non-trivial aspect of the evaluation of distributional semantic models that should not be overlooked. In future work, we will apply the consistency metric to evaluate other models, and datasets from other domains.

Acknowledgments

We are grateful to Yvette Oortwijn and Arianna Betti for their input as Quine domain experts. We also thank them as well as Lisa Dondorp, Thijs Ossenkoppele and Maud van Lier for their work on the Quine in Context corpus. We thank Pia Sommerauer for help with the SVD setup, and the UvA e-Ideas group for their fruitful discussion of a draft of this paper. We also thank the anonymous reviewers for their time and valuable comments. This research was supported by VICI grant *e-Ideas* (277-20-007) awarded to Arianna Betti and VENI grant *Reading between the lines* 275-89-029 awarded to Antske Fokkens, both financed by the Dutch Research Council (NWO).

References

Fatemeh Torabi Asr, Jon Willits, and Michael Jones. 2016. Comparing predictive and co-occurrence based models of lexical semantics trained on child-directed speech. In *CogSci*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Arianna Betti and Hein van den Berg. 2016. Towards a Computational History of Ideas. In *Proceedings of the Third Conference on Digital Humanities in Luxembourg with a Special Focus on Reading Historical Sources in the Digital Age. CEUR Workshop Proceedings, CEUR-WS.org*, volume 1681, Aachen.

Arianna Betti, Hein van den Berg, Yvette Oortwijn, and Caspar Treijtel. 2019. History of Philosophy in Ones and Zeros. In Mark Curtis and Eugen Fischer, editors, *Methodological Advances in Experimental Philosophy*, Advances in Experimental Philosophy, pages 295–332. Bloomsbury, London.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49(1-47).

Susan Carey and Elsa Bartlett. 1978. Acquiring a single new word. *Papers and Reports on Child Language Development*, 15:17–29.

Stephen Clark. 2015. Vector space models of lexical meaning. *The Handbook of Contemporary semantic theory*, pages 493–522.

Marc N Coutanche and Sharon L Thompson-Schill. 2014. Fast mapping rapidly integrates information into existing memory networks. *Journal of Experimental Psychology: General*, 143(6):2296.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Annapaola Ginammi, Jelke Bloem, Rob Koopman, Shenghui Wang, and Arianna Betti. in press. Bolzano, Kant, and the Traditional Theory of Concepts: A Computational Investigation. In *The Dynamics of Science: Computational Frontiers in History and Philosophy of Science*. Pittsburgh University Press.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42.

Johannes Hellrich and Udo Hahn. 2016. Bad company - neighborhoods in neural embedding spaces considered harmful. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2785–2796.

Aurélie Herbelot and Marco Baroni. 2017. High-risk learning: acquiring new word vectors from tiny data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 304–309.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Alexandre Kabbach, Kristina Gulordava, and Aurélie Herbelot. 2019. Towards incremental learning of word embeddings using context informativeness. In *Proceedings of the 57th Conference of the Association for Computational Linguistics: Student Research Workshop*, pages 162–168, Florence, Italy. Association for Computational Linguistics.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, 41:677–705.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Matthew McGrath and Devin Frank. 2018. Propositions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, spring 2018 edition. Metaphysics Research Lab, Stanford University.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. 2018. Evaluation of domain-specific word embeddings using knowledge resources. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Willard Van Orman Quine. 1934. *A system of logistic*. Harvard University Press.

Willard Van Orman Quine. 1960. *Word & Object*. MIT Press.

Magnus Sahlgren and Alessandro Lenci. 2016. The effects of data size and frequency range on distributional semantic models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 975–980.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C.-C. Jay Kuo. 2019. Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8:e19.

Cross-Domain Training for Goal-Oriented Conversational Agents

Alexandra Bodirlau, Stefania Budulan, Traian Rebedea
University Politehnica of Bucharest, Romania
alexandra.bodirlau@stud.acs.upb.ro
{stefania.budulan, traian.rebedea}@cs.pub.ro

Abstract

Goal-Oriented Chatbots in fields such as customer support, providing specific information or general help with bookings or reservations, suffer from low performance partly due to the difficulty of obtaining large domain-specific annotated datasets. Given that the problem is closely related to the domain of the conversational agent and that data belonging to a specific domain is difficult to annotate, there have been some attempts at surpassing these challenges such as unsupervised pre-training or transfer learning between different domains. A more thorough analysis of the transfer learning mechanism is justified by the significant boost of the results demonstrated in the results section. We describe extensive experiments using transfer learning and warm-starting techniques with improvements of more than 5% in relative percentage of success rate in the majority of cases, and up to 10x faster convergence as opposed to training the system without them.

1 Introduction

Goal-Oriented Conversational Agents (GO Chatbots) are seeing increased use to help users to achieve predetermined goals, but they can handle only very simple tasks, such as playing songs, searching information, set alarms or reminders. Building a dialogue agent to fulfill complex tasks remains one of the fundamental challenges for the Natural Language Processing (NLP) community and Artificial Intelligence (AI) in general.

There are two dominant approaches for solving this problem. The first one relies on (fully) supervised learning, e.g. using sequence-to-sequence (Sutskever et al., 2014) models, encoding a user's utterance and its context to decode the answer provided by the chatbot. However, this method does not explicitly allow to locate and make use of specific information such as entity recognition (e.g. a person's workplace) and requires large amounts of data in order to flawlessly extract and process particular pieces of information relevant for the task at hand, usually a mandatory requirement for GO Chatbots.

The second category entails partitioning the dialog system into smaller subsystems, usually implemented and trained separately. An example of such a system (Li et al., 2017) consists of several components: Natural Language Understanding, Dialog Manager, and Natural Language Generation. The Dialog Manager is often implemented with the aid of reinforcement learning (RL) based techniques, for instance using Deep Q-Nets (DQN) (Mnih et al., 2015) and having the main goal of learning the policy on account of which the agent will be able to provide answers.

The first approach is used with more favourable outcomes in the case of open-domain dialogue systems (Serban et al., 2016) than in closed-domain dialogue systems (Peng et al., 2017), because it does not require a method to reward the accomplishment of the task. Instead, the success of the conversation resides in the engagement of the user, measured in the level of coherence and cohesion of the dialog. The second approach better fits learning tasks having less labeled data, where the validity of the answer can be determined through evaluating the task's completion (e.g. making a restaurant reservation). These RL-based dialogue systems have the ability to simulate conversations, thus exploring the unknown dialogue space efficiently.

Currently reduced performance of domain-specific conversational agents in fields such as customer support, providing certain information or general help with reservations etc., is partly due to the difficulty of obtaining large annotated datasets. With each new domain and each new conversation flow introduced by a new task, newly annotated data need to be fed into the system in order to assimilate them and later provide the best answer for different inputs. The efforts for selecting, categorising and annotating the data are substantial, no matter the previous experience or domain-knowledge. This paper analyses the possibility to alleviate the data annotation endeavor through the inter-domain transfer learning technique (Ilievski et al., 2018). Alongside with unsupervised pre-training and others, transfer learning has proven a significant contribution to deliver better results, but it has only been tested with datasets from a small number of domains. We experiment with larger datasets, wider scenarios and we offer a richer understanding of the method, premises and results for overcoming the lack of data.

In this paper, we provide a thorough study on the impact of transfer learning in goal-oriented chatbots, starting from the work presented by (Ilievski et al., 2018). They proposed the possibility to reuse the knowledge gained from a source domain to boost the training and testing performance of a machine learning chatbot model on a different target domain, as described in more detail in the following sections. They identify two cases:

- **domain overlap** - the source and target domains are different, but share a fraction of actions, and
- **domain extension** - the source domain is extended by the target domain.

In both cases, there are common actions that justify the transfer learning between domains instead of independently training models for each of them. This approach has two effects: (1) the success rate of the model obtained with transfer learning is significantly higher than that of the model trained without any prior knowledge, and (2) transfer learning can be an alternative or complementary to warm starting, which also requires labeled data.

The results presented by the aforementioned authors represent a significant improvement for GO Chatbots, but they are obtained with only three domains, with relatively small datasets: Movie Booking, Restaurant Booking, and Tourist Info. In order to train a model for a more complex domain such as customer support, the improvement has to be validated on multiple datasets from different domains. Also, because the cost of annotating data for such a domain is very high, transfer learning methods should be studied for possible improvements that increase the automation of domain-specific conversations with few data.

The rest of the paper is organized as follows. Section 2 presents the related work for Goal Oriented Chatbots. The model used in our experiments is detailed in Section 3 and the datasets in Section 4. The results of our experiments are described in detail and interpreted in Section 5. Finally, future improvements and conclusions are presented in Section 6.

2 Related Work

We have already classified the existing solutions used for building machine learning chatbots in two categories, based on the learning method: supervised learning and reinforcement learning. In this section, we are providing a more in depth analysis of these two alternatives.

Serban et al. (2016) propose a solution for non-goal-driven systems, which uses an encoder-decoder model and word embeddings to generate the response of the agent starting from the utterance of the user as input. The architecture is composed from two RNNs: one for the utterance level, which treats the dialogue as a sequence of utterances, and one at the word level, which processes an utterance as a sequence of words. This model is trained on movie scripts and the dialogues include the speech acts. A detail worth mentioning here is that the pre-training is performed on a large related, but non-dialogue, corpus. The consequence is that the model accomplishes slightly better results compared with an initialization with fixed word embeddings.

Another supervised learning model for chatbots is presented by (Wen et al., 2017). The architecture is significantly more complex, and is divided in several modules. The utterances received from the user are converted into two representations: a probability distribution over the slot-value pairs called the belief state, and an intent representation generated by an intent network. A database operator selects the most probable values in the belief state and makes a query to the database. A policy network takes as input the intent representation, database result, and belief state and returns a representation of the next system action. Finally, a generation network uses the action representation to generate a template sequence, which is filled with actual values from the database. This system is very similar to the one used in the current paper, but the former is trained in a supervised fashion and, therefore, it is possible to fail at finding a good policy due to the shortcomings in dialogue exploration. Firstly, the policy is learned by a network instead of using RL (our case) and, secondly, the $\epsilon - greedy$ policy used in our experiments ensures the exploration of unknown states, instead of relying entirely on seen training data and rigid choices.

A possible solution for the disadvantage of using supervised training in the model presented above is proposed by Su et al. (2016). The architecture is similar, but there is a difference in the policy network training: it receives the current state and predicts the next system action in a supervised fashion in the first phase, followed by a reinforcement learning phase. The purpose of the second phase is to improve the generalization capacity of the policy by a better exploration of the action space using reinforcement learning.

A step forward in solving complex tasks is done by Peng et al. (2017). They introduce a hierarchical deep reinforcement learning architecture for solving composite tasks for travel planning, that are a collection of subtasks such as: book air ticket, reserve hotel room, buy train ticket, etc. This type of tasks are a challenge for RL approaches because of the reward sparsity, the slot constraints between different subtasks and the agent's tendency to switch between different subtasks frequently when conversing with users, which leads to poor user experience. The dialogue manager consists of (1) a top-level dialogue policy that selects subtasks, (2) a low-level dialogue policy that selects the actions in a given subtask, and (3) a global state tracker that supervises the cross-subtask constraints.

Ilievski et al. (2018) use the transfer learning mechanism for chatbots employing neural models to reduce the amount of training data and speed up the learning process for new domains. This can be accomplished with the transfer of the parameters learned in a source domain to a target domain, which has some common actions with the former. In order to apply the transfer, it is necessary to have the same state distribution in both domains, therefore the bots trained on the source domain must be aware of the actions in the target domain. They obtain an improvement of 65% in success rate in the case of domain extension and 20% for domain overlap. This represents a noteworthy result and one of the reasons we chose to study this mechanism in more detail. Another reason is the faster learning resulted from the combination of transfer learning with warm start.

In a more recent paper, Wolf et al. (2019) present the improvement brought by transfer learning in generative tasks such as open-domain dialog generation. A Transformer model (Vaswani et al., 2017) is pre-trained on a large unlabeled dataset, followed by a fine-tuning step in which two loss functions are optimized: (1) a next-utterance classification loss, and (2) a language modeling loss. As a result, the model outperforms the existing systems by a significant margin obtaining 51% absolute improvement in perplexity on the validation dataset.

Given that many works successfully engage unsupervised learning in various manners, there still remains the question: how does unsupervised pre-training work? An answer is formulated by (Erhan et al., 2010) and a possible explanation is that the pre-training guides the learning towards basins of attraction of minima that support better generalization from the training dataset. Therefore, it acts like a regularizer for the supervised fine-tunning phase, when the parameters are restricted to a relatively small space. This assumption is reinforced by the results that show an effectiveness' upgrade of pre-training as the number of units per layer increases, a better generalization performance, but worse training errors, and worse performance than random initialization for small networks, all characteristics of regularization. They also show a growth in the probability of finding a local minima by increasing the depth of a network with random initialization, compared to an unsupervised pre-training.

The most important advantage of pre-training is the possibility of using unlabeled data, which is really helpful given the high costs of data annotation. Therefore, the effect of pre-training with very large datasets observed in the experiments is the most surprising result of the paper (Erhan et al., 2010): the early examples determine the basin of attraction for the remainder of the training and the supervised fine-tuning cannot escape from it. The hypothesis is that those examples induce changes in the magnitude of the weights, which decreases the number of regions accessible to the stochastic gradient descent procedure. This is why, in a large-scale setting, the influence of unsupervised pre-training is still present, in contrast to the classical regularizers, when the effect disappears with more data.

Nevertheless, fine-tuning large pre-trained models is parameter inefficient, because each task requires an entirely new model. A compact and extensible model is needed in order to solve this shortcoming. For this purpose, (Houlsby et al., 2019) introduce adapter-based tuning, which achieves a mean GLUE score of 80.0 on several text classification tasks, compared to 80.4 achieved by full fine-tuning, using 1.3 task-specific parameters in total, compared to 9. This method also facilitates continual learning (training on a sequence of tasks) and multi-task learning (training on simultaneous tasks).

3 Model

The system used in this paper is a *semantic frames* system (Li et al., 2017). It represents the dialogue as a set of slot-value pairs and at each step t, given the user utterance u_t, the agent takes an action a_t, which can be either the final result or a request for a value of an empty slot. The architecture consists of two parts: a **User Simulator** module and a **Dialogue Manager** module.

The purpose of the **User Simulator** is to interact with the Dialogue Manager in order to train a policy for an agent. First, a user goal is chosen randomly from the goals' pool and is unknown for the agent, but it tries to help the user to accomplish it during the dialogue. The goal consists of two types of slots:

- *inform slots* - represent the constraints imposed by the user, hence their values are known (e.g. {movie_name: ”deadpool”, city: ”Madison Heights”, date: ”saturday”, number_of_people: ”5”}).
- *request slots* - represent the values that the agent should provide, hence they enclose unknown values to the user (e.g. {price, start_time, critic_rating}).

Then, the user utterance u_t is generated following the **Agenda-Based** model (Li et al., 2016): the user has an internal state s_u composed of a goal G and an agenda A. The goal consists of constraints C and requests R. At each step t, the user simulator generates the user action $a_{u,t}$ based on the current state $s_{u,t}$ and the last agent action $a_{a,t}$ and updates the current state $s^{'}_{u,t}$.

Natural Language Generation (NLG) is the module that generates natural language text for the user dialogue actions. For better results, a hybrid approach is used, including a model-based NLG and a template-based NLG. The model-based NLG is an LSTM decoder, which takes a dialogue action as input and generates a sentence with slot placeholders. If the sentence can be found in the predefined templates, the template-based NLG is applied for filling the slots, otherwise, the utterance generated by the model-based NLG is used.

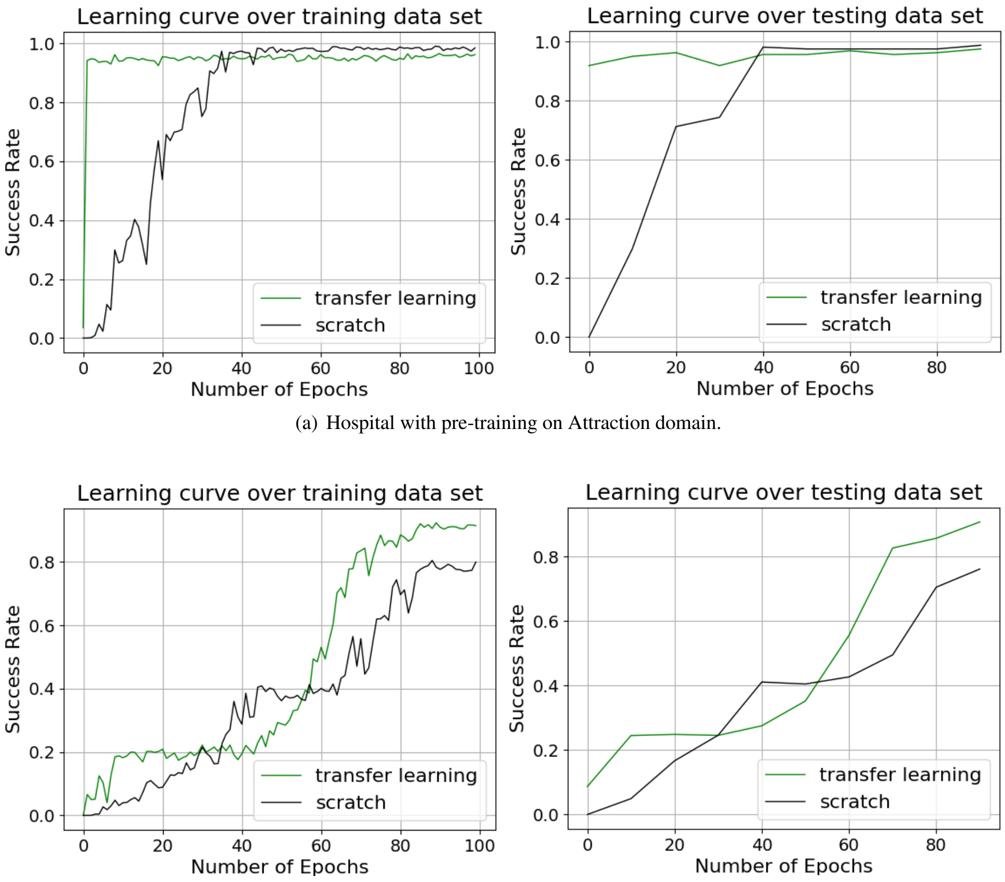
Natural Language Understanding (NLU) is the opposite to the NLG module: it takes as input an utterance and determines the user's *intent* and the set of *slots* associated with it (e.g. {movie_name: ”deadpool”, date: ”saturday”, number_of_people: ”5”}), in order to form a semantic frame. It is implemented with an LSTM and its objective is to maximize the conditional probability of the slots and the intent, given the utterance.

The **Dialogue Management (DM)** includes two submodules: **Dialogue State Tracker** and **Policy Learning** module. The goal of the **Dialogue State Tracker** is to build a representation of the current state for policy learning, using the semantic frame received from the NLU component. It keeps the history of the user utterances, system actions and the query results from the Knowledge Base.

Policy learning module generates the next action of the system a_t according to the policy $\pi = P(a|s)$, given the current state s_t, in order to accomplish the user goal in the smallest number of steps. The state s_t includes the latest user action, the latest agent action, turn information, history dialogue turns and the available database results. A DQN (Mnih et al., 2015) is used to approximate the state-action function $Q(s, a|\Theta)$ and contains the experience replay mechanism.

4 Dataset

In order to study the impact of transfer learning in multiple domains, we choose MultiWOZ



(a) Hospital with pre-training on Attraction domain.

(b) Taxi with pre-training on Train domain.

Figure 1: Small number of extra slots in target domain.

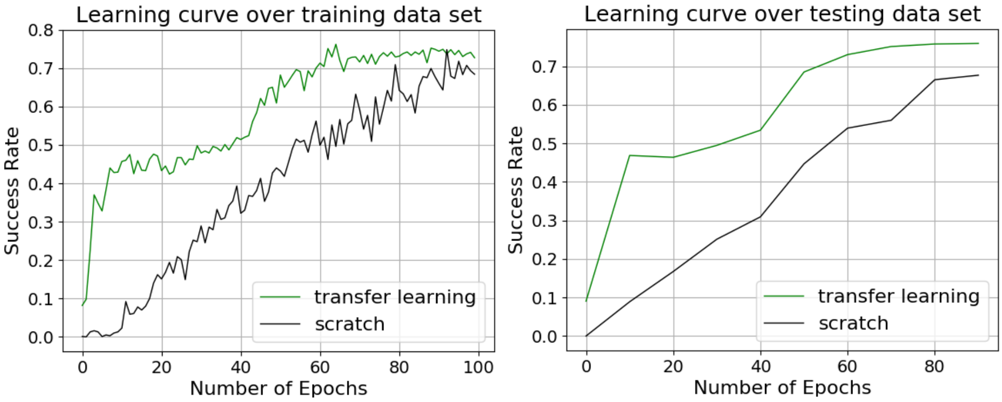
2.0 (Budzianowski et al., 2018), a large-scale multi-domain corpus of natural human-human conversations, collected through the Wizard-of-Oz framework (Kelley, 1984). It contains about 10000 samples from seven domains, with an average of turns per dialogue between 8.9 and 15, depending on the domain. From this dataset, we select the following five domains: **hotel**, **attraction**, **train**, **taxi**, **hospital**, and also keep **movie** and **tourist** domains used by Ilievski et al. (2018). These domains are grouped in source-target pairs according to their common slots, resulting five new opportunities for transfer learning. The total number of slots for source and target domains, respectively, as well as the amount of common slots is presented in Table1. We call *extra source/target slots* the difference between the total domain slots and the common ones.

The goals can be divided into two categories depending on whether they contain request slots or not. In the first case, the user sends a list of inform slots to the agent and the agent should accomplish the task respecting the constraints imposed by the user. In the second case, the user sends a list of request slots besides the list of inform slots, and the agent should accomplish the task and answer to the user's questions.

	Source Domain	Target Domain	Source Slots	Target Slots	Common Slots
1	movie	restaurant	6	9	3
2	restaurant	tourist	9	9	6
3	**hotel**	**attraction**	13	9	5
4	**train**	**taxi**	9	6	4
5	**movie**	**hotel**	17	13	5
6	**tourist**	**hotel**	9	13	6
7	**attraction**	**hospital**	9	4	3

Table 1: Number of slots per domain

This is a noteworthy detail, because it can influence the success rate through the experience accumulated in the warm start phase. In this phase, a fixed-size buffer is filled with experiences from positive-outcome conversations. Thus, the learning process gains a boost when having to self-calibrate based on an experience which will lead



(a) Attraction with pre-training on Hotel domain.

Figure 2: Medium number of extra slots in target domain.

to goal-achievement. We have noticed that the best results are obtained with warm start on no-request goals, following that the agent will manage to achieve request goals during training. As we increased the percentage of request goals in the warm start buffer, the overall success rate decreases and the learning curve becomes less smooth.

The total number of goals per domain is distributed as follows:

- 3000 training and 400 testing user goals in hotel, train and attraction;
- 2000 training and 200 testing user goals in taxi datasets;
- 80 training and 15 testing user goals in hospital dataset.

5 Experiments

The experiments are executed on overlapping domains, with the setup from (Ilievski et al., 2018) and running each experiment 10 times, with $n_{epochs} = 100$ epochs. The second set of experiments mimic testing on extension domain by restricting the slots in the source domain to the common ones. The warm start technique with experience replay buffer is used for both *transfer learning* and *scratch* agent (the same version, but without transfer learning), and the experience buffer is flushed when the agent reaches, for the first time, a success rate of 0.3. We also keep the maximal number of allowed turns per dialogue $n_{max\_turns} = 20$ in most experiments, except in the case of attraction domain with pre-training on hotel. In this situation, the number of turns is too small compared with the number of slots from the hotel domain, and the agent trained on the source domain is not able to learn. Consequently, we increased n_{max_turns} to 40 turns.

5.1 Different Domains

The first set of experiments aims to analyze the convergence for the agent with and without transfer learning on new domains. We group the experiments in three categories, according to the number of extra slots in the target domain, as follows: 1. small number of extra slots (less than 2 slots); 2. medium number of extra slots (between 3 and 6 slots); and 3. big number of extra slots (greater than 6 slots). We are interested in the improvement transfer learning brings to the success rate and the convergence pace.

Figure 1 presents the learning curve for the target domains with a small number of extra slots. For hospital domain with pre-training on attraction, both agents converge to a success rate greater than 95%, but the agent with transfer learning converges in a few epochs (<5), while the scratch agent needs 40 epochs to reach similar accuracy values. In the case of taxi domain with pre-training on train domain, the improvement of transfer learning is 15% for train dataset and 19% for test dataset. In absolute terms, the success rate increases from 80% to 91% for train dataset and from 76% to 91% on test dataset.

For the attraction domain with pre-training on hotel, presented in Figure 2, the model obtained with transfer learning has a success rate 7% higher than that of the scratch model on train dataset. This denotes an increase from 68% to 73% in absolute terms. For test dataset, transfer learning improves the success rate from 68% to 76% or with 12% in relative terms.

The last category registers the lowest overall success rate for both agents and we consider that these results stem from the large number of extra slots in the target domain. The learning curves are illustrated in Figure 3 and the success rate is similar for train and test datasets. Hotel domain with pre-training on movie has a success rate of 27% with transfer learning and 8.5% without transfer learning, with an improvement of 217%. While the same domain with tourist as source domain of transfer learning, registers a relative boost of 737%, from 4.3% to 36%.

5.2 Same Domains, Different Number of Slots

The second set of experiments targets the evolution of the success rate according to the number of extra slots, both in the source and target domain. The selected experiment evaluates the hotel domain with pre-training on tourist dataset, given they each have large number of slots with few common ones (see Table1). We keep the setup for the other parameters and only change the number of extra slots in one domain, while the other remains constant.

	Source Slots	Target Slots	Scratch Score	TL Score	Scratch Epochs	TL Epochs
1	6	6	0.81	0.88	100	40
2	7	6	0.81	0.86	100	70
3	8	6	0.81	0.84	100	70
4	9	6	0.77	0.83	100	50
5	6	9	0.55	0.72	100	100
6	7	9	0.57	0.63	100	100
7	8	9	0.54	0.70	80	100
8	9	9	0.56	0.70	100	100

Table 2: Source Slots number influence

The final success rate on the test dataset for constant slots in target domain is summarized in Table 2. When the target contains only the common slots, we observe a decrease of the success rate with less than 2% with each extra slot added in the source domain, for the model trained with transfer learning. However, it is still by 6.6% greater than the success rate of the agent trained with any other prior knowledge. An interesting fact is that the same test with the common slots plus three extra slots in target domain has the effect of diminishing the success rate by an average of 15% compared with the previous situation. At the same time, the improvement over the scratch agent is equal to 24%.

	Source Slots	Target Slots	Scratch Score	TL Score	Scratch Epochs	TL Epochs
1	6	6	0.81	0.88	100	40
2	6	7	0.75	0.85	90	40
3	6	8	0.63	0.79	70	100
4	6	9	0.55	0.72	100	100
5	6	10	0.53	0.59	100	100
6	6	11	0.09	0.52	100	90
7	6	12	0.01	0.44	90	100
8	6	13	0.0	0.39	10	100
9	9	6	0.77	0.83	100	50
10	9	7	0.71	0.82	90	100
11	9	8	0.65	0.76	100	100
12	9	9	0.56	0.70	100	100
13	9	10	0.54	0.59	100	100
14	9	11	0.11	0.52	100	100
15	9	12	0.04	0.42	90	100
16	9	13	0.04	0.36	100	60

Table 3: Target Slots number influence

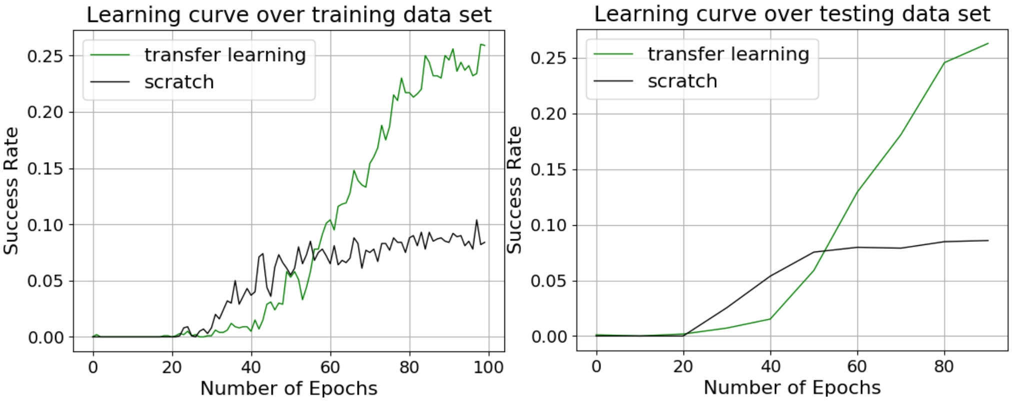
As expected, the number of extra slots in target domain has a bigger influence over the final results. Therefore, the relative average decrease of the success rate for the agent with transfer learning is 9.5% for each extra slot, while the source domain contains only common slots. Another three slots added to the source dataset generates an average decrease of 3.6%, relative to the previous test. In comparison with the scratch agent, the improvement increases from 79% in the first case, to 274% in the latter.

6 Conclusions

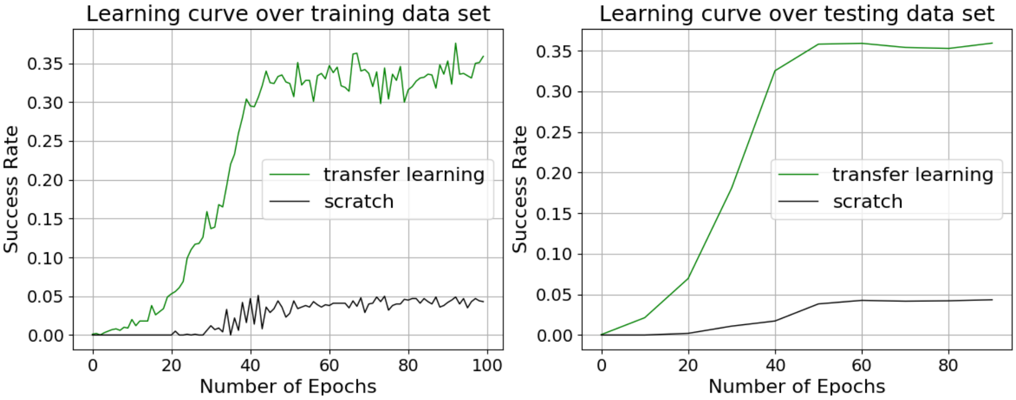
In this paper, we study the factors that influence the success of transfer learning approach in Reinforcement Learning-based Goal-Oriented Chatbots and demonstrate the results on five new cases of overlapping domains. We found that a big number of different slots between the source domain and the target domain leads to a smaller success rate. Even so, the transfer learning mechanism brings a betterment of over 79% over the agent trained with no prior knowledge.

The outcomes encourage the use of transfer learning with warm start on various cases of overlapping and extending source and target domains. However, the optimal selection in terms of hyperparameters of the system, such as the number of epochs or the number of maximum turns in a conversation, need to be determined for each particular scenario. They are, after all, directly influenced by the amount of data and its characteristics: number of slots, types and distribution of goals, and the degree of overlapping between source and target slots.

Further work involves developing wider experiment scenarios for hierarchical deep reinforce-



(a) Hotel with pre-training on Movie domain.



(b) Hotel with pre-training on Tourist domain.

Figure 3: Big number of extra slots in target domain.

ment learning system and introducing the transfer learning approach into this architecture when the sub-tasks share slots. Moreover, we can imagine other transfer learning setups such as sharing subtasks as the learnt common part, instead of slots, from one composite task to another. All these attempts have the objective of gaining more context information and better performance with less annotated data, which is onerous to obtain.

References

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. pages 5016–5026.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11:625–660.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*. PMLR, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. http://proceedings.mlr.press/v97/houlsby19a.html.

Vladimir Ilievski, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. 2018. Goal-oriented chatbot dialog management bootstrapping with transfer learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*. pages 4115–4121.

John F. Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems* 2:26–41. https://doi.org/10.1145/357417.357420.

Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng

Gao. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*. pages 733–743.

Xiujun Li, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *CoRR* abs/1612.05688.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu A., Joel Veness, and Marc G. Bellemare et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2231–2240.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*. pages 3776–3784.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *CoRR* abs/1606.02689.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Ashish Vaswani, Noam Shazeer, Niki Parmar, and Jakob Uszkoreit et al. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. pages 6000–6010.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 438–449.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR* abs/1901.08149.

Learning Sentence Embeddings for Coherence Modelling and Beyond

Tanner Bohn **Yining Hu** **Jinhang Zhang** **Charles X. Ling**
Department of Computer Science, Western University, London, ON, Canada
{tbohn,yhu534,jzha337,charles.ling}@uwo.ca

Abstract

We present a novel and effective technique for performing text coherence tasks while facilitating deeper insights into the data. Despite obtaining ever-increasing task performance, modern deep-learning approaches to NLP tasks often only provide users with the final network decision and no additional understanding of the data. In this work, we show that a new type of sentence embedding learned through self-supervision can be applied effectively to text coherence tasks while serving as a window through which deeper understanding of the data can be obtained. To produce these sentence embeddings, we train a recurrent neural network to take individual sentences and predict their location in a document in the form of a distribution over locations. We demonstrate that these embeddings, combined with simple visual heuristics, can be used to achieve performance competitive with state-of-the-art on multiple text coherence tasks, outperforming more complex and specialized approaches. Additionally, we demonstrate that these embeddings can provide insights useful to writers for improving writing quality and informing document structuring, and assisting readers in summarizing and locating information.

1 Introduction

A goal of much of NLP research is to create tools that not only assist in completing tasks, but help gain insights into the text being analyzed. This is especially true of text coherence tasks, as users are likely to wonder where efforts should be focused to improve writing or understand how text should be reorganized for improved coherence. By improving coherence, a text becomes easier to read and understand (Lapata and Barzilay, 2005), and in this work we particularly focus on measuring coherence in terms of sentence ordering.

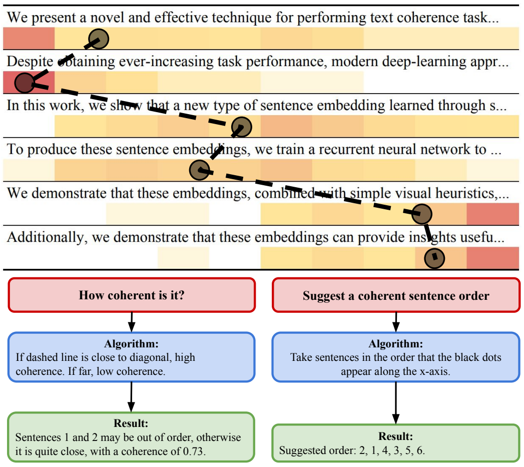

We present a novel and effective technique for performing text coherence task...
Despite obtaining ever-increasing task performance, modern deep-learning appr...
In this work, we show that a new type of sentence embedding learned through s...
To produce these sentence embeddings, we train a recurrent neural network to ...
We demonstrate that these embeddings, combined with simple visual heuristics,...
Additionally, we demonstrate that these embeddings can provide insights usefu...
How coherent is it?
Algorithm:
If dashed line is close to diagonal, high coherence. If far, low coherence.
Result:
Sentences 1 and 2 may be out of order, otherwise it is quite close, with a coherence of 0.73.
Suggest a coherent sentence order
Algorithm:
Take sentences in the order that the black dots appear along the x-axis.
Result:
Suggested order: 2, 1, 4, 3, 5, 6.

Figure 1: This paper abstract is analyzed by our sentence position model trained on academic abstracts. The sentence encodings (predicted position distributions) are shown below each sentence, where white is low probability and red is high. Position quantiles are ordered from left to right. The first sentence, for example, is typical of the first sentence of abstracts as reflected in the high first-quantile value. For two text coherence tasks, we show the how the sentence encodings can easily be used to solve them. The black dots indicate the weighted average predicted position for each sentence.

Many recent approaches to NLP tasks make use of end-to-end neural approaches which exhibit ever-increasing performance, but provide little value to end-users beyond a classification or regression value (Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018). This leaves open the question of whether we can achieve good performance on NLP tasks while simultaneously providing users with easily obtainable insights into the data. This is precisely what the work in this paper aims to do in the context of coherence analysis, by providing a tool with which users can quickly and visually gain insight into structural information about a text. To accomplish this, we rely on the surprising importance of sentence location in many areas of natural language processing. If a sentence does not appear to belong where it is located, it decreases the coherence and readability of the text (Lapata and Barzilay, 2005). If a sentence is located at the beginning of a document or news article, it is very likely to be a part of a high quality extractive summary (See et al., 2017). The location of a sentence in a scientific abstract is also an informative indicator of its rhetorical purpose (Teufel et al., 1999). It thus follows that the knowledge of where a sentence should be located in a text is valuable.

Tasks requiring knowledge of sentence position – both relative to neighboring sentences and globally – appear in text coherence modelling, with two important tasks being order discrimination (is a sequence of sentences in the correct order?) and sentence ordering (re-order a set of unordered sentences). Traditional methods in this area make use of manual feature engineering and established theory behind coherence (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008; Grosz et al., 1995). Modern deep-learning based approaches to these tasks tend to revolve around taking raw words and directly predicting local (Li and Hovy, 2014; Chen et al., 2016) or global (Cui et al., 2017; Li and Jurafsky, 2017) coherence scores or directly output a coherent sentence ordering (Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018). While new deep-learning based approaches in text coherence continue to achieve ever-increasing performance, their value in real-world applications is undermined by the lack of actionable insights made available to users.

In this paper, we introduce a self-supervised approach for learning sentence embeddings which can be used effectively for text coherence tasks (Section 3) while also facilitating deeper understanding of the data (Section 4). Figure 1 provides a taste of this, displaying the sentence embeddings for the abstract of this paper. The self-supervision task we employ is that of predicting the location of a sentence in a document given only the raw text. By training a neural network on this task, it is forced to learn how the location of a sentence in a structured text is related to its syntax and semantics. As a neural model, we use a bidirectional recurrent neural network, and train it to take sentences and predict a discrete distribution over possible locations in the source text. We demonstrate the effectiveness of predicted position distributions as an accurate way to assess document coherence by performing order discrimination and sentence reordering of scientific abstracts. We also demonstrate a few types of insights that these embeddings make available to users that the predicted location of a sentence in a news article can be used to formulate an effective heuristic for extractive document summarization – outperforming existing heuristic methods.

The primary contributions of this work are thus:

1. We propose a novel self-supervised approach to learn sentence embeddings which works by learning to map sentences to a distribution over positions in a document (Section 2.2).

2. We describe how these sentence embeddings can be applied to established coherence tasks using simple algorithms amenable to visual approximation (Section 2.3).

3. We demonstrate that these embeddings are competitive at solving text coherence tasks (Section 3) while quickly providing access to further insights into texts (Section 4).

2 Predicted Position Distributions

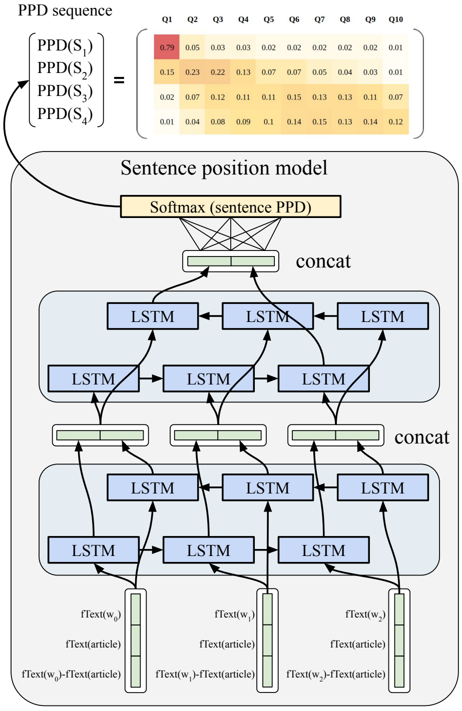
2.1 Overview

By training a machine learning model to predict the location of a sentence in a body of text (conditioned upon features not trivially indicative of position), we obtain a sentence position model such that sentences predicted to be at a particular location possess properties typical of sentences found at that position. For example, if a sentence is predicted to be at the beginning of a news article, it should resemble an introductory sentence.

In the remainder of this section we describe our neural sentence position model and then discuss how it can be applied to text coherence tasks.

2.2 Neural Position Model

The purpose of the position model is to produce sentence embeddings by predicting the position in



Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
0.79	0.05	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.01
0.15	0.23	0.22	0.13	0.07	0.07	0.05	0.04	0.03	0.01
0.02	0.07	0.12	0.11	0.11	0.15	0.13	0.13	0.11	0.07
0.01	0.04	0.08	0.09	0.1	0.14	0.15	0.13	0.14	0.12

Figure 2: Illustration of the sentence position model, consisting of stacked BiLSTMs. Sentences from a text are individually fed into the model to produce a PPD sequence. In this diagram we see a word sequence of length three fed into the model, which will output a single row in the PPD sequence.

a text of a given sentence. Training this model requires no manual labeling, needing only samples of text from the target domain. By discovering patterns in this data, the model produces sentence embeddings suitable for a variety of coherence-related NLP tasks.

2.2.1 Model Architecture

To implement the position model, we use stacked bi-directional LSTMs (Schuster and Paliwal, 1997) followed by a softmax output layer. Instead of predicting a single continuous value for the position of a sentence as the fraction of the way through a document, we frame sentence position prediction as a classification problem.

Framing the position prediction task as classification was initially motivated by the poor performance of regression models; since the task of position prediction is quite difficult, we observed that regression models would consistently make predictions very close to 0.5 (middle of the document), thus not providing much useful information. To convert the task to a classification problem, we aim to determine what quantile of the document a sentence resides in. Notationally, we will refer to the number of quantiles as Q. We can interpret the class probabilities behind a prediction as a discrete distribution over positions for a sentence, providing us with a predicted position distribution (PPD). When $Q = 2$ for example, we are predicting whether a sentence is in the first or last half of a document. When $Q = 4$, we are predicting which quarter of the document it is in. In Figure 2 is a visualization of the neural architecture which produces PPDs of $Q = 10$.

2.2.2 Features Used

The sentence position model receives an input sentence as a sequence of word encodings and outputs a single vector of dimension Q. Sentences are fed into the BiLSTM one at a time as a sequence of word encodings, where the encoding for each word consists of the concatenation of: (1) a pretrained word embedding, (2) the average of the pretrained word embedding for the entire document (which is constant for all words in a document), and (3) the difference of the first two components (although this information is learnable given the first two components, we found during early experimentation that it confers a small performance improvement). In addition to our own observations, the document-wide average component was also shown in (Logeswaran et al., 2018) to improve performance at sentence ordering, a task similar to sentence location prediction. For the pretrained word embeddings, we use 300 dimensional fastText embeddings[1], shown to have excellent cross-task performance (Joulin et al., 2016). In Figure 2, the notation $ftxt(token)$ represents converting a textual token (word or document) to its fastText embedding. The embedding for a document is the average of the embeddings for all words in it.

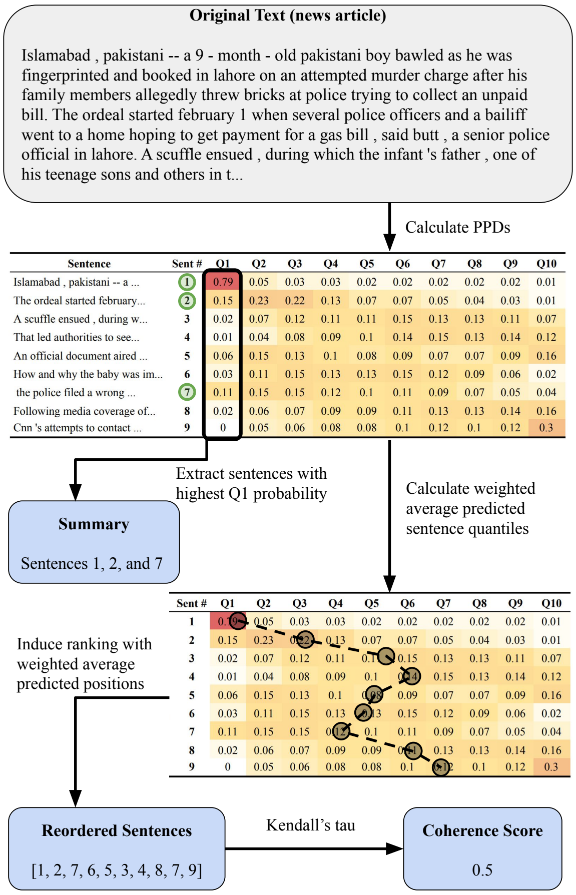
The features composing the sentence embeddings fed into the position model must be chosen carefully so that the order of the sentences does not directly affect the embeddings (i.e. the sentence embeddings should be the same whether the sentence ordering is permuted or not). This is because we want the predicted sentence positions to be independent of the true sentence position, and not every sentence embedding technique provides

[1] Available online at `https://fasttext.cc/docs/en/english-vectors.html`. We used the `wiki-news-300d-1M` vectors.

this. As a simple example, if we include the true location of a sentence in a text as a feature when training the position model, then instead of learning the connection between sentence meaning and position, the mapping would trivially exploit the known sentence position to perfectly predict the sentence quantile position. This would not allow us to observe where the sentence *seems* it should be located.

2.3 Application to Coherence Tasks

For the tasks of both sentence ordering and calculating coherence, PPDs can be combined with simple visually intuitive heuristics, as demonstrated in Figure 3.



Original Text (news article)

Islamabad , pakistani -- a 9 - month - old pakistani boy bawled as he was fingerprinted and booked in lahore on an attempted murder charge after his family members allegedly threw bricks at police trying to collect an unpaid bill. The ordeal started february 1 when several police officers and a bailiff went to a home hoping to get payment for a gas bill , said butt , a senior police official in lahore. A scuffle ensued , during which the infant 's father , one of his teenage sons and others in t...

Sentence	Sent #	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Islamabad , pakistani -- a ...	1	0.79	0.05	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.01
The ordeal started february...	2	0.15	0.23	0.22	0.13	0.07	0.07	0.05	0.04	0.03	0.01
A scuffle ensued , during w...	3	0.02	0.07	0.12	0.11	0.11	0.15	0.13	0.13	0.11	0.07
That led authorities to see...	4	0.01	0.04	0.08	0.09	0.1	0.14	0.15	0.13	0.14	0.12
An official document aired ...	5	0.06	0.15	0.13	0.1	0.08	0.09	0.07	0.07	0.09	0.16
How and why the baby was im...	6	0.03	0.11	0.15	0.13	0.13	0.15	0.12	0.09	0.06	0.02
the police filed a wrong ...	7	0.11	0.15	0.15	0.12	0.1	0.11	0.09	0.07	0.05	0.04
Following media coverage of...	8	0.02	0.06	0.07	0.09	0.09	0.11	0.13	0.13	0.14	0.16
Cnn 's attempts to contact ...	9	0	0.05	0.06	0.08	0.08	0.1	0.12	0.1	0.12	0.3

Extract sentences with highest Q1 probability
Calculate weighted average predicted sentence quantiles
Summary
Sentences 1, 2, and 7
Induce ranking with weighted average predicted positions

Sent #	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	0.79	0.05	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.01
2	0.15	0.23	0.22	0.13	0.07	0.07	0.05	0.04	0.03	0.01
3	0.02	0.07	0.12	0.11	0.11	0.15	0.13	0.13	0.11	0.07
4	0.01	0.04	0.08	0.09	0.1	0.14	0.15	0.13	0.14	0.12
5	0.06	0.15	0.13	0.1	0.08	0.09	0.07	0.07	0.09	0.16
6	0.03	0.11	0.15	0.13	0.13	0.15	0.12	0.09	0.06	0.02
7	0.11	0.15	0.15	0.12	0.1	0.11	0.09	0.07	0.05	0.04
8	0.02	0.06	0.07	0.09	0.09	0.11	0.13	0.13	0.14	0.16
9	0	0.05	0.06	0.08	0.08	0.1	0.12	0.1	0.12	0.3

Figure 3: A visualization of our NLP algorithms utilizing PPDs applied to a news article. To reorder sentences, we calculate average weighted positions (identified with black circles) to induce an ordering. Coherence is calculated with the Kendall's rank correlation coefficient between the true and induced ranking. We also show how PPDs can be used to perform summarization, as we will explore further in Section 4.

2.3.1 Sentence Ordering

To induce a new ordering on a sequence of sentences, S, we simply sort the sentence by their weighted average predicted quantile, $\hat{\mathcal{Q}}(s \in S)$, defined by:

$$\hat{\mathcal{Q}}(s) = \sum_{i=1}^{Q} i \times PPD(s)_i, \tag{1}$$

where $PPD(s)$ is the Q-dimensional predicted position distribution/sentence embedding for the sentence s.

2.3.2 Calculating coherence

To calculate the coherence of a text, we employ the following simple algorithm on top of the PPDs: use the Kendall's tau coefficient between the sentence ordering induced by the weighted average predicted sentence positions and the true sentence positions:

$$coh = \tau((\hat{\mathcal{Q}}(s), \text{ for } s = S_1, ..., S_{|S|}), (1, ..., |S|)). \tag{2}$$

3 Experiments

In this section, we evaluate our PPD-based approaches on two coherence tasks and demonstrate that only minimal performance is given up by our approach to providing more insightful sentence embeddings.

Task	Dataset	Q	Epochs	Layer dropouts	Layer widths
Order Discrim.	Accident	5	10	(0.4, 0.2)	(256, 256)
	Earthquake	10	5	(0.4, 0.2)	(256, 64)
Reordering	NeurIPS	15	20	(0.5, 0.25)	(256, 256)

Table 1: The neural sentence position model hyperparameters used in our coherence experiments. The following settings are used across all tasks: batch size of 32, sentence trimming/padding to a length of 25 words, the vocabulary is set to the 1000 most frequent words in the associated training set. The Adamax optimizer is used (Kingma and Ba, 2014) with default parameters supplied by Keras (Chollet et al., 2015).

Order discrimination setup. For order discrimination, we use the Accidents and Earthquakes datasets from (Barzilay and Lapata, 2008) which consists of aviation accident reports and news articles related to earthquakes respectively. The task is to determine which of a permuted

	Order discrimination		Reordering	
Model	**Accident**	**Earthquake**	**Acc**	τ
Random	50	50	15.6	0
Entiry Grid	90.4	87.2	20.1	0.09
Window network	-	-	41.7	0.59
LSTM_PtrNet	93.7	99.5	50.9	0.67
RNN Decoder	-	-	48.2	0.67
Varient-LSTM+PtrNet	94.4	99.7	51.6	**0.72**
ATTOrderNet	**96.2**	**99.8**	**56.1**	**0.72**
PPDs	94.4	99.3	54.9	**0.72**

Table 2: Results on the order discrimination and sentence reordering coherence tasks. Our approach trades only a small decrease in performance for improved utility of the sentence embeddings over other approaches, achieving close to or the same as the state-of-the-art.

ordering of the sentences and the original ordering is the most coherent (in the original order), for twenty such permutations. Since these datasets only contain training and testing partitions, we follow (Li and Hovy, 2014) and perform 10-fold cross-validation for hyperparameter tuning. Performance is measured with the accuracy with which the permuted sentences are identified. For example, the Entity Grid baseline in Table 2 gets 90.4% accuracy because given a shuffled report and original report, it correctly classifies them 90.4% of the time.

Sentence ordering setup. For sentence ordering, we use past NeurIPS abstracts to compare with previous works. While our validation and test partitions are nearly identical to those from (Logeswaran et al., 2018), we use a publicly available dataset[2] which is missing the years 2005, 2006, and 2007 from the training set ((Logeswaran et al., 2018) collected data from 2005 - 2013). Abstracts from 2014 are used for validation, and 2015 is used for testing. To measure performance, we report both reordered sentence position accuracy as well as Kendall's rank correlation coefficient. For example, the Random baseline correctly predicts the index of sentences 15.6% of the time, but there is no correlation between the predicted ordering and true ordering, so $\tau = 0$.

Training and tuning. Hyperparameter tuning for both tasks is done with a random search, choosing the hyperparameter set with the best validation score averaged across the 10 folds for order discrimination dataset and for three trials for the sentence reordering task. The final hyperparameters chosen are in Table 1.

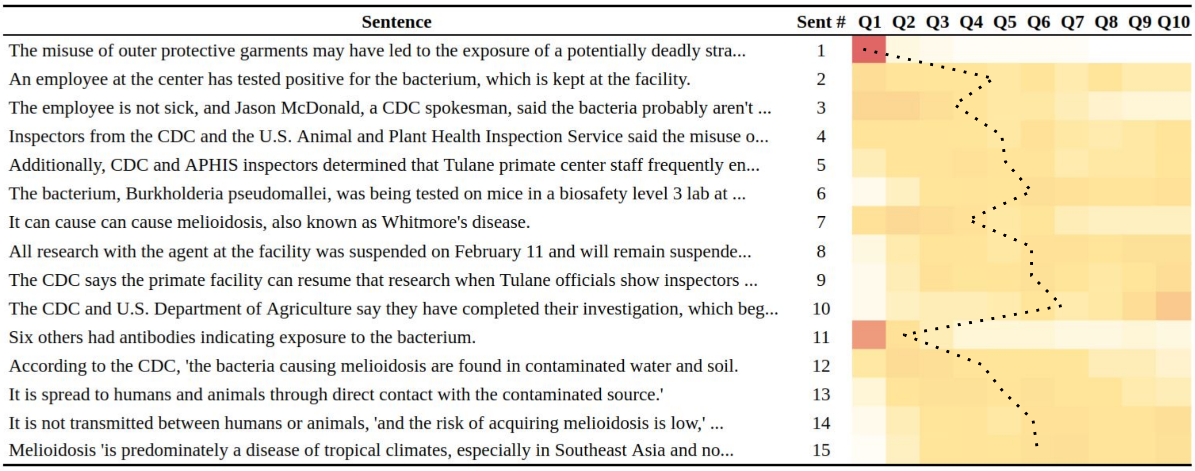
Baselines. We compare our results against a random baseline, the traditional Entity Grid approach from (Barzilay and Lapata, 2008), Window network (Li and Hovy, 2014), LSTM+PtrNet (Gong et al., 2016), RNN Decoder and Varient-LSTM+PtrNet from (Logeswaran et al., 2018), and the most recent state-of-the art ATTOrderNet (Cui et al., 2018).

Results. Results for both coherence tasks are collected in Table 2. For the order discrimination task, we find that on both datasets, our PPD-based approach only slightly underperforms ATTOrderNet (Cui et al., 2018), with performance similar to the LSTM+PtrNet approaches (Gong et al., 2016; Logeswaran et al., 2018). On the more difficult sentence reordering task, our approach exhibits performance closer to the state-of-the-art, achieving the same ranking correlation and only slightly lower positional accuracy. Given that the publicly available training set for the reordering task is slightly smaller than that used in previous work, it is possible that more data would allow our approach to achieve even better performance. In the next section we will discuss the real-world value offered by our approach that is largely missing from existing approaches.

4 Actionable Insights

A primary benefit of applying PPDs to coherence-related tasks is the ability to gain deeper insights into the data. In this section, we will demon-

[2] https://www.kaggle.com/benhamner/nips-papers



Sentence	Sent #
The misuse of outer protective garments may have led to the exposure of a potentially deadly stra...	1
An employee at the center has tested positive for the bacterium, which is kept at the facility.	2
The employee is not sick, and Jason McDonald, a CDC spokesman, said the bacteria probably aren't ...	3
Inspectors from the CDC and the U.S. Animal and Plant Health Inspection Service said the misuse o...	4
Additionally, CDC and APHIS inspectors determined that Tulane primate center staff frequently en...	5
The bacterium, Burkholderia pseudomallei, was being tested on mice in a biosafety level 3 lab at ...	6
It can cause can cause melioidosis, also known as Whitmore's disease.	7
All research with the agent at the facility was suspended on February 11 and will remain suspende...	8
The CDC says the primate facility can resume that research when Tulane officials show inspectors ...	9
The CDC and U.S. Department of Agriculture say they have completed their investigation, which beg...	10
Six others had antibodies indicating exposure to the bacterium.	11
According to the CDC, 'the bacteria causing melioidosis are found in contaminated water and soil.	12
It is spread to humans and animals through direct contact with the contaminated source.'	13
It is not transmitted between humans or animals, 'and the risk of acquiring melioidosis is low,' ...	14
Melioidosis 'is predominately a disease of tropical climates, especially in Southeast Asia and no...	15

Figure 4: The PPDs for a CNN article. (full text available at http://web.archive.org/web/20150801040019id_/http://www.cnn.com/2015/03/13/us/tulane-bacteria-exposure/). The dashed line shows the weighted average predicted sentence positions.

strate the following in particular: (1) how PPDs can quickly be used to understand how the coherence of a text may be improved, (2) how the existence of multiple coherence subsections may be identified, and (3) how PPDs can allow users to locate specific types of information without reading a single word, a specific case of which is extractive summarization. For demonstrations, we will use the news article presented in Figure 4.

4.1 Improving Coherence

For a writer to improve their work, understanding the incoherence present is important. Observing the PPD sequence for the article in Figure 4 makes it easy to spot areas of potential incoherence: they occur where consecutive PPDs are significantly different (from sentences 1 to 2, 6 to 7, and 10 to 11). In this case, the writer may determine that sentence 2 is perhaps not as introductory as it should be. The predicted incoherence between sentences 10 and 11 is more interesting, and as we will see next, the writer may realize that this incoherence may be okay to retain.

4.2 Identifying Subsections

In Figure 4, we see rough progressions of introductory-type sentences to conclusory-type sentences between sentences 1 and 10 and sentences 11 and 15. This may indicate that the article is actually composed of two coherent subsections, which means that the incoherence between sentences 10 and 11 is expected and natural. By being able to understand where subsections may occur in a document, a writer can make informed decisions on where to split a long text into more coherent chunks or paragraphs. Knowing where approximate borders between ideas in a document exist may also help readers skim the document to find desired information more quickly, as further discussed in the next subsection.

4.3 Locating Information and Summarization

When reading a new article, readers well-versed in the subject of the article may want to skip high-level introductory comments and jump straight to the details. For those unfamiliar with the content or triaging many articles, this introductory information is important to determine the subject matter. Using PPDs, locating these types of information quickly should be easy for readers, even when the document has multiple potential subsections. In Figure 4, sentences 1 and 11 likely contain introductory information (since the probability of occurring in the first quantiles is highest), the most conclusory-type information is in sentence 10, and lower-level details are likely spread among the remaining sentences.

Locating sentences with the high-level details of a document is reminiscent of the task of extractive summarization, where significant research has been performed (Nenkova et al., 2011; Nenkova and McKeown, 2012). It is thus natural to ask how well a simple PPD-based approach performs

Model (lead baseline source)	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3 (Nallapati et al., 2017)	39.2	15.7	35.5
Lead-3 (See et al., 2017)	40.3	17.7	36.6
Lead-3 (Ours)	35.8	15.9	33.5
SummaRuNNer (Nallapati et al., 2017) ((Nallapati et al., 2017))	39.6	16.2	35.3
Pointer-generator (See et al., 2017) ((See et al., 2017))	39.5	17.3	36.4
RL (Paulus et al., 2017) ((Nallapati et al., 2017))	41.2	15.8	39.1
TextRank (Mihalcea and Tarau, 2004) (ours)	26.2	11.1	24.3
Luhn (Luhn, 1958) (ours)	26.4	11.2	24.5
SumBasic (Nenkova and Vanderwende, 2005) (ours)	27.8	10.4	26.0
LexRank (Erkan and Radev, 2004) (ours)	28.4	11.6	26.3
PPDs (ours)	**30.1**	**12.6**	**28.2**

Table 3: ROUGE scores on the CNN/DailyMail summarization task. Our PPD-based heuristic outperforms the suite of established heuristic summarizers. However, the higher performance of the deep-learning models demonstrates that training explicitly for summarization is beneficial.

at summarization. To answer this question, the summarization algorithm we will use is: select the n sentences with the highest $PPD(s \in S)_0$ value, where S is the article being extractively summarized down to n sentences. For the article in Figure 4, sentences 1, 11, and 3 would be chosen since they have the highest first-quantile probabilities. This heuristic is conceptually similar to the Lead heuristic, where sentences that actually occur at the start of the document are chosen to be in the summary. Despite its simplicity, the Lead heuristic often achieves near state-of-the-art results (See et al., 2017).

We experiment on the non-anonymized CNN/DailyMail dataset (Hermann et al., 2015) and evaluate with full-length ROUGE-1, -2, and -L F1 scores (Lin and Hovy, 2003). For the neural position model, we choose four promising sets of hyperparameters identified during the hyperparameter search for the sentence ordering task in Section 3 and train each sentence position model on 10K of the 277K training articles (which provides our sentence position model with over 270K sentences to train on). Test results are reported for the model with the highest validation score. The final hyperparameters chosen for this sentence location model are: Q = 10, epochs = 10, layer dropouts = (0.4, 0.2), layer widths = (512, 64).

We compare our PPD-based approach to other heuristic approaches[3]. For completeness, we also include results of deep-learning based approaches and their associated Lead baselines evaluated using full-length ROUGE scores on the non-anonymized CNN/DailyMail dataset.

Table 3 contains the the comparison between our PPD-based summarizer and several established heuristic summarizers. We observe that our model has ROUGE scores superior to the other heuristic approaches by a margin of approximately 2 points for ROUGE-1 and -L and 1 point for ROUGE-2. In contrast, the deep-learning approaches trained explicitly for summarization achieve even higher scores, suggesting that there is more to a good summary than the sentences simply being introductory-like.

5 Related Work

Extensive research has been done on text coherence, motivated by downstream utility of coherence models. In addition to the applications we demonstrate in Section 4, established applications include determining the readability of a text (coherent texts are easier to read) (Barzilay and Lapata, 2008), refinement of multi-document summaries (Barzilay and Elhadad, 2002), and essay scoring (Farag et al., 2018).

Traditional methods to coherence modelling utilize established theory and handcrafted linguistic features (Grosz et al., 1995; Lapata, 2003). The Entity Grid model (Lapata and Barzilay, 2005; Barzilay and Lapata, 2008) is an influential traditional approach which works by first constructing a sentence × discourse entities (noun phrases) occurrence matrix, keeping track of the syntactic role of each entity in each sentence. Sentence transition probabilities are then calculated using this representation and used as a feature vector as in-

[3] Implementations provided by Sumy library, available at https://pypi.python.org/pypi/sumy.

put to a SVM classifier trained to rank sentences on coherence.

Newer methods utilizing neural networks and deep learning can be grouped together by whether they indirectly or directly produce an ordering given an unordered set of sentences.

Indirect ordering. Approaches in the indirect case include Window network (Li and Hovy, 2014), Pairwise Ranking Model (Chen et al., 2016), the deep coherence model from (Cui et al., 2017), and the discriminative model from (Li and Jurafsky, 2017). These approaches are trained to take a set of sentences (anywhere from two (Chen et al., 2016) or three (Li and Hovy, 2014) to the whole text (Cui et al., 2017; Li and Jurafsky, 2017)) and predict whether the component sentences are already in a coherent order. A final ordering of sentences is constructed by maximizing coherence of sentence subsequences.

Direct ordering. Approaches in the direct case include (Gong et al., 2016; Logeswaran et al., 2018; Cui et al., 2018). These model are trained to take a set of sentences, encode them using some technique, and with a recurrent neural network decoder, output the order in which the sentences would coherently occur.

Models in these two groups all use similar high-level architectures: a recurrent or convolutional sentence encoder, an optional paragraph encoder, and then either predicting coherence from that encoding or iteratively reconstructing the ordering of the sentences. The PPD-based approaches described in Section 2 take a novel route of directly predicting location information of each sentence. Our approaches are thus similar to the direct approaches in that position information is directly obtained (here, in the PPDs), however the position information produced by our model is much more rich than simply the index of the sentence in the new ordering. With the set of indirect ordering approaches, our model approach to coherence modelling shares the property that induction of an ordering upon the sentences is only done after examining all of the sentence embeddings and explicitly arranging them in the most coherent fashion.

6 Conclusions

The ability to facilitate deeper understanding of texts is an important, but recently ignored, property for coherence modelling approaches. In an effort to improve this situation, we present a self-supervised approach to learning sentence embeddings, which we call PPDs, that rely on the connection between the meaning of a sentence and its location in a text. We implement the new sentence embedding technique with a recurrent neural network trained to map a sentence to a discrete distribution indicating where in the text the sentence is likely located. These PPDs have the useful property that a high probability in a given quantile indicates that the sentence is typical of sentences that would occur at the corresponding location in the text.

We demonstrate how these PPDs can be applied to coherence tasks with algorithms simple enough such that they can be visually performed by users while achieving near state-of-the-art, outperforming more complex and specialized systems. We also demonstrate how PPDs can be used to obtain various insights into data, including how to go about improving the writing, how to identify potential subsections, and how to locate specific types of information, such as introductory or summary information. As a proof-of-concept, we additionally show that despite PPDs not being designed for the task, they can be used to create a heuristic summarizer which outperforms comparable heuristic summarizers.

In future work, it would be valuable to evaluate our approach on texts from a wider array of domains and with different sources of incoherence. In particular, examining raw texts identified by humans as lacking coherence could be performed, to determine how well our model correlates with human judgment. Exploring how the algorithms utilizing PPDs may be refined for improved performance on the wide variety of coherence-related tasks may also prove fruitful. We are also interested in examining how PPDs may assist with other NLP tasks such as text generation or author identification.

Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grants Program. NSERC invests annually over $1 billion in people, discovery and innovation.

References

Regina Barzilay and Noemie Elhadad. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research* .

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952* .

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2018. Deep attentive sentence ordering network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 4340–4349. http://aclweb.org/anthology/D18-1465.

Baiyun Cui, Yingming Li, Yaqing Zhang, and Zhongfei Zhang. 2017. Text coherence analysis based on deep neural network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, pages 2027–2030.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Youmna Farag, Helen Yannakoudakis, and Ted Briscoe. 2018. Neural automated essay scoring and coherence modeling for adversarially crafted input. *arXiv preprint arXiv:1804.06898* .

Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-end neural sentence ordering using pointer network. *arXiv preprint arXiv:1611.04953* .

Barbara J Grosz, Scott Weinstein, and Aravind K Joshi. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics* 21(2):203–225.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 545–552.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *IJCAI*. volume 5, pages 1085–1090.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 2039–2048.

Jiwei Li and Dan Jurafsky. 2017. Neural net models of open-domain discourse coherence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 198–209.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 71–78.

Lajanugen Logeswaran, Honglak Lee, and Dragomir Radev. 2018. Sentence ordering and coherence modeling using recurrent neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*. pages 3075–3081.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining text data*, Springer, pages 43–76.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval* 5(2–3):103–233.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005* 101.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1073–1083.

Simone Teufel et al. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, Citeseer.

Risk Factors Extraction from Clinical Texts based on Linked Open Data

Svetla Boytcheva[1] and **Galia Angelova**[1] and **Zhivko Angelov**[2]
[1] Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences, Sofia, Bulgaria
svetla.boytcheva@gmail.com, galia@lml.bas.bg
[2] ADISS Lab Ltd,
4 Hristo Botev blvd., 1463 Sofia, Bulgaria
angelov@adiss-bg.com

Abstract

This paper presents experiments in risk factors analysis based on clinical texts enhanced with Linked Open Data (LOD). The idea is to determine whether a patient has risk factors for a specific disease analyzing only his/her outpatient records. A semantic graph of "meta-knowledge" about a disease of interest is constructed, with integrated multilingual terms (labels) of symptoms, risk factors etc. coming from Wikidata, PubMed, Wikipedia and MESH, and linked to clinical records of individual patients via ICD–10 codes. Then a predictive model is trained to foretell whether patients are at risk to develop the disease of interest. The testing was done using outpatient records from a nation-wide repository available for the period 2011-2016. The results show improvement of the overall performance of all tested algorithms (kNN, Naïve Bayes, Tree, Logistic regression, ANN), when the clinical texts are enriched with LOD resources.

1 Motivation

Recently, with the improving quality of Natural Language Processing (NLP), it is increasingly recognized as the most useful tool to extract clinical information from the free text of scientific medical publications and clinical records. In this way NLP becomes an instrument supporting biomedical research and new application scenarios are sought to reveal patterns and dependencies expressed by medical texts. Open-source NLP software appears, tailored to clinical text, and this increases NLP dissemination and acceptance. The construction of language resources for biomedical NLP goes in parallel to technology development. The large variety of medical terminology systems is continuously transformed and integrated into standardized, structured repositories of Linked Open Data[1]; de-identified data sets of electronic health records (EHRs) are made available as open resources[2]. Current hype in open linked data and collective efforts for their generation allow to benefit from the multilingual versions of some encyclopedic datasets like Wikidata and Wikipedia. Still there is a lack of NLP tools and linguistic resources with sufficient quality for processing medical texts in languages other than English but the interest to process such texts increases too.

Our goal is to determine whether a patient has risk factors for a specified disease, according to the information in his/her outpatient record. We suggest to enrich patient-related clinical narratives with additional information sources in order to enable a deeper investigation of dependencies between diseases and risk factors. In general it is difficult to predict the risk of a certain disease from the text of a clinical record only. Patient history contains numerous facts that are documented within a series of records but most often the medical expert reads them in isolation. In addition, many symptoms might signal various diseases. We propose to construct semantic graphs of "meta-knowledge" about diseases of interest, to integrate there multilingual terms (labels) of symptoms, risk factors etc., and to link clinical records of individual patients to this construction with the hope to discover new hints and interrelations that are not contained in the primary documents.

In the experiments presented here, patient records in Bulgarian language are enhanced with

[1] https://lod-cloud.net/

[2] E.g. at BioPortal https://bioportal.bioontology.org/ and at DBMI Data Portal https://portal.dbmi.hms.harvard.edu/

semantic information provided by medical ontologies and other resources in English, like scientific publications and encyclopedic data. Several data mining experiments were run on datasets containing outpatient records of diabetic patients in Bulgarian linked to encyclopedic extracts and Life Sciences LOD in English. The results show that LOD infuse some relations that are not found by standard text mining techniques of clinical narratives, and thus enable the discovery of associations hinting to further risk factors for diabetes mellitus.

2 Related Work

Mining of inter-related collections of clinical texts and LOD is still rare. On the one hand, with hundreds of open biomedical ontologies and numerous biomedical datasets made available as LOD, there is a salient opportunity to integrate clinical and biomedical data to better interpret patient-related texts and to uncover associations of biomedical interest. On the other hand, such mining experiments require significant efforts to make clinical data interoperable with standardized health terminologies, biomedical ontologies and growing LOD repositories. One of the earliest papers in this direction is (Pathak et al., 2013) which describes how patient EHRs data at Mayo Clinic are represented as Resource Description Framework (RDF) in order to identify potential drug-drug interactions for widely prescribed cardiovascular and gastroenterology drugs. Some drug-drug interactions of interest were identified which suggest lack of consensus on practice guidelines and recommendations. The authors of (Odgers and Dumontier, 2015) describe how they transformed a de-identified version of the STRIDE[3] EHRs into a semantic clinical data warehouse containing among others annotated clinical notes. They showed the feasibility of using semantic web technologies to directly exploit existing biomedical ontologies and LOD. As far as NLP is concerned, an open-source tool (NegEx) is used in the EHR transformation to recognize negated terms. The integrated search in EHR data and LOD is not yet considered as a popular trend in the secondary use of clinical narratives (Meystre et al., 2017) and is still an emerging direction of research mostly due to the complex data preparation.

Information Extraction (IE) refers to the automatic extraction of concepts, entities and events as well as their relations and associated attributes from free text. A recent review of clinical IE applications (Wang et al., 2018) notes the increasing interest to NLP but lists only 25 IE systems which were used multiple times, outside the labs where they were created. Isolated attempts exist to apply IE in the context of EHR processing in frameworks for semantic search, for instance SemEHR deployed to identify contextualized mentions of biomedical concepts within EHRs in a number of UK hospitals (Wu et al., 2018). We mention the following research prototypes as experimental developments, based on some sort of IE: (Shi et al., 2017) reports about a system extracting textual medical knowledge from heterogeneous sources in order to integrate it into knowledge graphs; (Hassanpour and Langlotz, 2016) describes a machine learning system that annotates radiology reports and extracts concepts according to a model covering most clinically significant contents in radiology; (Jackson et al., 2018) presents the information extraction and retrieval architecture CogStack, deployed in the King's College Hospital. CogStack has functionality to transform records into de-identified text documents and applies generic clinical IE pipelines to derive additional structured data from free texts.

Most of the successful systems listed above work for clinical narratives in English. All major resources, ontologies and terminology classifications like UMLS[4] and MESH[5] are available in English. The comprehensive ontology SNOMED CT[6] was developed initially in English and then translated to other languages. Progress in biomedical NLP for languages other than English will catalyze the development of tools in the respective languages and will enable access to medical data presented in a variety of languages (Névéol et al., 2018). In Europe, the European commission supports the development of multilingual platforms like SEMCARE which performs queries on unstructured medical data in English, German, and Dutch (López-García et al., 2016).

Using Big Data (nowadays - millions of EHRs) to advance medical research and health care prac-

[3] Stanford Translational Research Integrated Database Environment including a repository for EHR data, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2815452/

[4] https://www.nlm.nih.gov/research/umls/

[5] https://meshb.nlm.nih.gov/search

[6] http://www.snomed.org/

tices is now on the rise (Kessel and Combs, 2016). Core NLP components are already embedded in general clinical platforms similar to CogStack (Jackson et al., 2018). Development of high quality corpora and terminology is a key factor for NLP progress in smaller languages. Here we employ English terminology in data mining tasks concerning EHRs in Bulgarian language.

3 Materials

The datasets used in this study are a blend between LOD and clinical texts in Bulgarian language that belong to the Repository underpinning the Bulgarian Diabetes Register.

The Register was automatically generated in 2015 from 260 million pseudonymized outpatient records (ORs) provided by the National Health Insurance Fund (NHIF) for the period 2011–2014 for more than 5 million citizens yearly, more than 7 million citizens in total (Boytcheva et al., 2017). Updated twice with data about 2015 and 2016, today the Register is maintained by the University Specialized Hospital for Active Treatment of Diabetes (USHATE) - Medical University Sofia. At present the Repository of ORs, which underpins the Register, contains about 262 million records. These are reimbursement requests submitted by General Practitioners and Specialists from Ambulatory Care after every contact with a patient. The average number of patients with Diabetes Mellitus Type 2 (T2DM) per year is about 450,000.

In the primary database, from where we extract our datasets, the ORs are stored as semi-structured files with predefined XML-format. Administrative information is structured: visit date and time; pseudonymized personal data and visit-related information, demographic data etc. All diagnoses are given by ICD–10[7] codes and location names are specified in Bulgarian according to a standard nomenclature. However much information is provided as free text: anamnesis (case history, previous treatments, often family history, risk factors), patient status (summary of patient state, height, weight, body-mass index, blood pressure etc.), clinical tests (values of clinical examinations and lab data listed in arbitrary order) as well as prescribed treatment (codes of drugs reimbursed by NHIF, free text descriptions of other drugs).

To enhance clinical information with semantic data related to diagnoses, risk factors and symptoms, the following open datasets are selected:

- Wikidata[8] - contains multilingual encyclopedic information. Wikidata is a trusted resource, providing multilingual terminologies, their association with MESH codes, and complex relations between diagnoses, risk factors, and symptoms. Currently Wikidata contains descriptions of 5,227 items included in ICD–10 and 10,517 descriptions of items included in ICD–10–CT. The main problem is that many duplicated entities exist. For instance, for ICD–10 code I20 there are two items "angina pectoris (Q180762)" and "ischaemic heart disease (Q1444550)". Using SPARQL[9] queries, from Wikidata we collect for a given diagnosis all risk factors related to it as well as the associated MESH codes. From the list of risk factors that is originally in English we produce also a list in Bulgarian for the corresponding terms.

- PubMed[10] – the largest collection of scientific publications in the area of biomedicine and life sciences. From Pubmed we automatically extract publication abstracts and related MESH terms via advanced queries[11] through API. The search is limited to 10,000 abstracts in order to keep balance between the amounts of clinical narratives and texts of scientific publications.

- Wikipedia – from Wikipedia we extract automatically Wikipedia pages' summaries for a specified query via MediaWiki RESTful web service API[12]. The information in Wikipedia is encyclopedic and more broader, thus the semantic information there is too vague and shallow, in contrast to PubMed abstracts.

- MESH ontology – this ontology is chosen because both Pubmed publications and Wikidata contain references to it. In addition a mapping between MESH and SNOMED CT is available.

[7] http://apps.who.int/classifications/icd10/browse/2016/en#/

[8] https://www.wikidata.org/wiki/Wikidata:Main_Page

[9] https://www.w3.org/TR/rdf-sparql-query/

[10] https://www.ncbi.nlm.nih.gov/pubmed/

[11] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2651214/

[12] https://www.mediawiki.org/wiki/API:Tutorial

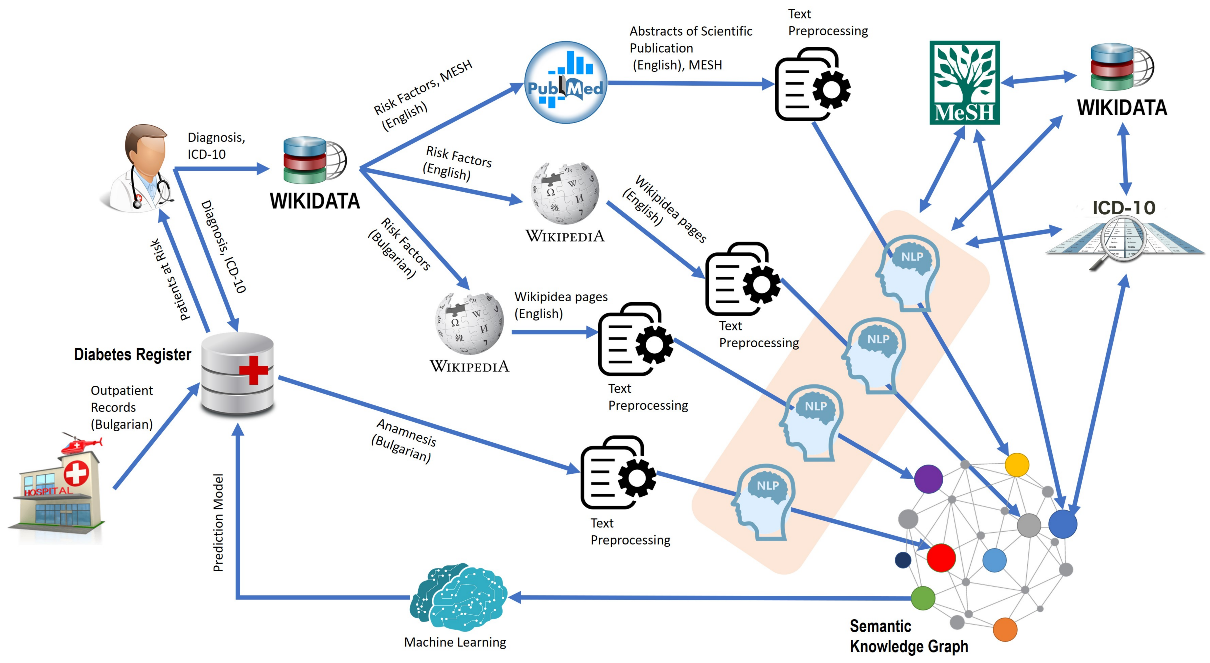


Figure 1: Pipeline for identification of patients at risk

4 Methods

The proposed method for risk factors identification is based on LOD and benefits from mapping multilingual data and using their vocabularies.

The data flow diagram is shown on Fig. 1. The process starts with selection of a diagnosis D, according to ICD–10 or ICD–10–CD by a medical expert who is looking for patients at risk in the Diabetes Register. The next step is to extract corresponding risk factors and symptoms for D in English $RE = \{re_1, re_2, ..., re_n\}$ from Wikidata, their equivalent terms in Bulgarian $RB = \{rb_1, rb_2, ..., rb_n\}$ and the MESH codes $M = \{m_1, m_2, ..., m_n\}$.

For each term re_i in English and its corresponding rb_i in Bulgarian, summaries of the respective Wikipedia pages are extracted automatically.

For each Mesh code m_i of risk factor r_i are extracted up to 10,000 abstracts of Pubmed publications and their annotations with MESH codes. For Pubmed the advanced search is done using an automatically generated query in the form:

```
(D/pc [majr] OR D/di [majr] OR
   D/ep [majr] D [mh]) AND
( re1 [mh] OR ...  OR ren [mh])
```

where the MeSH qualifiers for subheadings are: "pc" refers to "prevention and control"; "di" means "diagnosis"; "ep" is "epidemiology", "mh" - MeSH heading, and "majr" - to search MeSH heading that is a major topic of an article.

From the Bulgarian Diabetes Register a dataset is excerpted for patients with the diagnosis D. For those with recent D onset, ORs for previous periods are collected (only within 2011–2016).

4.1 Text Pre-Processing

The main transformations are done stepwise:

- tokenization - for Bulgarian language we used the UDPipe tokenizer[13].
- convertion of all words to lower case;
- removal of all punctuation marks;
- removal of all numbers;
- application of a stemmer and lemmatizer - for Bulgarian the UDPipe lemmatizer, for English Porter Stemmer (Porter, 2006);
- filtering stopwords - both for Bulgarian[14] and English;
- application of text vectorization based on TFIDF.

[13] https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-train.html#support_in_text_mining

[14] http://bultreebank.org/wp-content/uploads/2017/04/BTB-StopWordList.zip

4.2 Semantic Model

Semantic Knowledge Graphs are used recently as powerful representation of entities and relations between them (Paulheim, 2017). Often knowledge graphs are generated automatically from semi-structured resources or from the documents underpinning various ontologies, through terms/words they contain, by a combination of linguistic and statistical methods (Grainger et al., 2016).

In our experiment, all data are interlinked in a Semantic Knowledge graph via MESH codes and ICD–10 codes. Wikidata is the mediator between all resources providing cross-lingual ontology information and mapping between MESH and ICD-10 codes. Term mappings from MeSH to ICD-10 are 1,535 and to ICD-10-CM are 2,127. In addition Wikidata provides multilingual vocabulary for symptoms and diseases, and Pubmed publications are annotated by Mesh codes.

For each symptom and risk factor, related to the selected diagnosis D, the system identifies the most significant words related to D from the Wikipedia and Pubmed datasets respectively, using p-value as a measure for their significance. The knowledge graph is enriched with relations between these terms. The main relation between clinical texts and other resources is based on ICD-10 codes and some symptoms and risk factors that are presented in the anamnesis section of ORs.

4.3 Predictive Model

Two types of clinical texts are used as training datasets - for patients that have the diagnosis D, and for patients that do not have D. After text preprocessing, semantic hashing of all clinical texts in both datasets is done for predefined size of the hash. Two predictive models are applied:

- Based on the ORs information only
- Based on the ORs information enhanced with semantic data for symptoms and risk factors. In this case the vector space is extended; not only the dimensions of semantic hash vectors are used, but also additional dimensions for all symptoms, risk factors and the most significant terms related to them.

Several machine learning techniques were used to train the predictive model, including Naïve Bayes (NB) (McCallum et al., 1998), kNN, Tree, Logistic Regression and Artificial Neural Networks. (Dreiseitl and Ohno-Machado, 2002).

5 Experiments and Results

The diagnosis with ICD–10 code I20 "Ischaemic heart disease" is chosen for experiments because the Diabetes Register contains a plenty of clinical descriptions about this case. Patients with T2DM are at higher risk for developing I20 which is one of the T2DM complications. Sets of symptoms and risk factors for I20, both in Bulgarian and English, and English MeSH codes are automatically extracted by Wikidata queries. Seven symptoms are extracted: angina pectoris, nausea, dyspnea, lightheadedness, unstable angina, neck pain, fatigue. Only for three of them there are labels in Bulgarian, and for five of them there are MESH codes. In addition, there are 18 risk factors and for 11 of them labels in Bulgarian exist. Multiple MeSH codes are associated to some risk factors but there diagnoses without associated MeSH code. The final cardinality of the generated term set is $|RE| = 24$, $|RB| = 14$ and $|M| = 34$.

The Wikipedia API extracts 87 documents for a query with the RB terms in Bulgarian and we limit the set to the top 5 related documents. They contain 14,600 tokens from 3,627 types. Although only the top 5 most related documents are taken into consideration, some of the extracted Wikipedia pages are not directly related to the symptoms and risk factors, as they discuss e.g. herbs and medications for treatment. But some very related symptoms are included: for example for "nausea" the Wikipedia page about "vomiting" is extracted, and for "smoking" pages about "tobacco" and "pipe" are found. In addition, some barely related pages are extracted – mainly about some famous people, who suffer from the diseases in question and have related symptoms. Unfortunately the information in Wikipedia categories "Medical conditions" and "Diseases" for Bulgarian is too limited. For "Medical conditions" there are only 41 pages, and for many diseases the articles in the Bulgarian Wikipedia are stubs or some pages are not tagged in the respective categories.

For the query with RE terms 146 documents are identified that contain 40,099 tokens from 3,803 types. The extracted Wikipedia pages in English are also sometimes noisy and unrelated mainly due to the ambiguity e.g. pages about "Nausea(novel)" and "Nausea(band)", or "Insomnia (2002 film)" are extracted too. Other pages, indirectly related to the risk factors, contain information about Health organizations for treatment of the respective diseases, about diagnostic procedures, medication for the treatment etc.

Using the Pubmed advanced search, the generated query with RE terms identified 67,103 related scientific papers, from which we retrieved a subset of 2,000 abstracts only. The MeSH headings only contain 104,363 tokens from 2,509 types. Both MeSH heading and Pubmed abstracts contain 546,772 tokens from 12,684 types.

Despite the imperfection of all texts extracted from Wikipedia and PubMed, the most significant terms related to the predefined subsets of symptoms and risk factors are sound and correct. Their identification is based on bag–of–words and calculation of p-value ($p \leq 0.01$) and False Discovery Rate ($FD \leq 0.2$). For example, for тютюнопушене (tobacco smoking) the following words are identified as relevant: тютюнопушене (tobacco smoking), дим (smoke), пушене (smoking), тютюнев (tobacco), практикувам (practice), пристрастяване (addiction), пушач (smoker), цигара (cigar). These words were selected among 3,588 words in the text of related Wikipedia pages in Bulgarian. From those words 754 were filtered as relevant, and the final selection contain 9 words as most significant.

Two subsets of ORs (Anamnesis section) are extracted from the Repository behind the Diabetes Register: $S1$ for 36,580 patients with diagnosis I20 and $S2$ for 86,000 patients without diagnosis I20. All clinical texts are preprocessed. The total number of tokens in $S1$ and $S2$ is 123,258 from 25,086 types. For experiments are used kNN (5 neighbours, Mahalanobis metric), Tree (Pruning: at least 2 instances in leaves, at least 5 instances in internal nodes, maximum depth 100; Stop splitting when majority reaches 95%), Neural Network (30 hidden layers, Rectified Linear Activation Function (ReLu) (Nair and Hinton, 2010), stochastic gradient-based optimizer Adam (Kingma and Ba, 2014)), $\alpha = 0.0004$, Max iterations 200, replicable training) , NB and Logistic Regression (Ridge L2). The baseline results of the prediction model based on ORs only are presented in Table 1.

The results of prediction models trained with enhanced LOD data (Table 2) show improvement of the overall performance of all algorithms on this task, especially NB and Logistic Regression, when the clinical texts are enriched with additional information provided by open data resources and medical terminologies.

Model	F1	P	R
kNN	0.796	0.795	0.801
Tree	0.778	0.776	0.783
Neural Network	0.705	0.713	0.735
NB	0.588	0.615	0.701
Logistic Regression	0.581	0.640	0.703

Table 1: Baseline: Performance of employed ML algorithm using semantic hashing over ORs only.

Model	F1	P	R
kNN	0.819	0.752	0.899
Tree	0.893	0.858	0.932
Neural Network	0.743	0.746	0.760
NB	0.823	0.704	0.989
Logistic Regression	0.825	0.703	0.999

Table 2: Performance of employed ML algorithm using semantic hashing of ORs enhanced with semantic model data.

6 Conclusion and Further Work

The proposed approach shows how clinical texts can be enhanced with additional information about the diseases, their symptoms and risk factors. The experimental results show promising improvement of the risk factors prediction accuracy. Still there is a problem with Latin medical terminology that is often used in the Bulgarian clinical texts. Another issue is the imperfection of the additional terms provided by the LOD resources, due to many ambiguous terms included there. As future work we plan to apply word sense disambiguation to the texts extracted from open resources and more precise methods for constructing the relations in the semantic knowledge graphs. As future work we are planning to do deep analysis of the individual contribution of each new term added to the clinical texts. Another direction for future work is to use some transfer learning methods like UMLfit (Howard and Ruder, 2018), BERT (Devlin et al., 2018) and XLnet (Yang et al., 2019) to train models for word embedding on clinical texts in Bulgarian.

Acknowledgments

This research is partially funded by the Bulgarian Ministry of Education and Science, grant DO1-200/2018 ’Electronic health care in Bulgaria’ (e-Zdrave) and the Bulgarian National Science Fund, grant DN-02/4-2016 ’Specialized Data Mining Methods Based on Semantic Attributes’ (IZIDA). We are grateful to anonymous reviewers for useful comments and suggestions.

References

Svetla Boytcheva, Galia Angelova, Zhivko Angelov, and Dimitar Tcharaktchiev. 2017. Integrating data analysis tools for better treatment of diabetic patients. *CEUR Workshop Proceedings* 2022:229–236.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Stephan Dreiseitl and Lucila Ohno-Machado. 2002. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics* 35(5-6):352–359.

Trey Grainger, Khalifeh AlJadda, Mohammed Korayem, and Andries Smith. 2016. The semantic knowledge graph: A compact, auto-generated model for real-time traversal and ranking of any relationship within a domain. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, pages 420–429.

Saeed Hassanpour and Curtis Langlotz. 2016. Information extraction from multi-institutional radiology reports. *Artificial Intelligence in Medicine* 66:29–39.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

Richard Jackson, Ismail Kartoglu, Clive Stringer, et al. 2018. Cogstack - experiences of deploying integrated information retrieval and extraction services in a large national health service foundation trust hospital. *BMC Medical Informatics and Decision Makingvolume* 18.

Kerstin A Kessel and Stephanie E Combs. 2016. Review of developments in electronic, clinical data collection, and documentation systems over the last decade–are we ready for big data in routine health care? *Frontiers in oncology* 6:75.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Pablo López-García, Markus Kreuzthaler, Stefan Schulz, Daniel Scherr, Philipp Daumke, Kornél G Markó, Jan A Kors, Erik M van Mulligen, Xinkai Wang, Hanney Gonna, et al. 2016. Semcare: Multilingual semantic search in semi-structured clinical data. In *eHealth*. pages 93–99.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer, volume 752 (1), pages 41–48.

SM Meystre, Christian Lovis, T Bürkle, G Tognola, A Budrionis, and CU Lehmann. 2017. Clinical data reuse or secondary use: current status and potential future progress. *Yearbook of medical informatics* 26(01):38–52.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Aurélie Névéol, Hercules Dalianis, Sumithra Velupillai, Guergana Savova, and Pierre Zweigenbaum. 2018. Clinical natural language processing in languages other than english: opportunities and challenges. *Journal of biomedical semantics* 9(1):12.

David J Odgers and Michel Dumontier. 2015. Mining electronic health records using linked data. *AMIA Summits on Translational Science Proceedings* 2015:217.

Jyotishman Pathak, Richard C Kiefer, and Christopher G Chute. 2013. Using linked data for mining drug-drug interactions in electronic health records. *Studies in health technology and informatics* 192:682.

Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8(3):489–508.

Martin F Porter. 2006. An algorithm for suffix stripping. *Program* .

Longxiang Shi, Shijian Li, Xiaoran Yang, Jiaheng Qi, Gang Pan, and Binbin Zhou. 2017. Semantic health knowledge graph: Semantic integration of heterogeneous medical knowledge and services. *BioMed research international* 2017.

Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. 2018. Clinical information extraction applications: a literature review. *Journal of biomedical informatics* 77:34–49.

Honghan Wu, Giulia Toti, Katherine I Morley, Zina M Ibrahim, Amos Folarin, Richard Jackson, Ismail Kartoglu, Asha Agrawal, Clive Stringer, Darren Gale, et al. 2018. Semehr: A general-purpose semantic search system to surface semantic data from clinical notes for tailored care, trial recruitment, and clinical research. *Journal of the American Medical Informatics Association* 25(5):530–537.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* .

Parallel Sentence Retrieval From Comparable Corpora for Biomedical Text Simplification

Rémi Cardon
UMR CNRS 8163 – STL
F-59000 Lille, France
remi.cardon@univ-lille.fr

Natalia Grabar
UMR CNRS 8163 – STL
F-59000 Lille, France
natalia.grabar@univ-lille.fr

Abstract

Parallel sentences provide semantically similar information which can vary on a given dimension, such as language or register. Parallel sentences with register variation (like expert and non-expert documents) can be exploited for the automatic text simplification. The aim of automatic text simplification is to better access and understand a given information. In the biomedical field, simplification may permit patients to understand medical and health texts. Yet, there is currently no such available resources. We propose to exploit comparable corpora which are distinguished by their registers (specialized and simplified versions) to detect and align parallel sentences. These corpora are in French and are related to the biomedical area. Manually created reference data show 0.76 inter-annotator agreement. Our purpose is to state whether a given pair of specialized and simplified sentences is parallel and can be aligned or not. We treat this task as binary classification (alignment/non-alignment). We perform experiments with a controlled ratio of imbalance and on the highly unbalanced real data. Our results show that the method we present here can be used to automatically generate a corpus of parallel sentences from our comparable corpus.

1 Introduction

Parallel sentences provide semantically similar information which can vary on a given dimension. Typically, parallel sentences are collected in two languages and correspond to mutual translations. In the general language, the Europarl (Koehn, 2005) corpus provides such sentences in several pairs of languages. Yet, the dimension on which the parallelism is positioned can come from other levels, such as expert and non-expert register of language. The following pair of sentences (first in expert and second in non-expert languages) illustrates this:

- *Drugs that inhibit the peristalsis are contraindicated in that situation*
- *In that case, do not take drugs intended for blocking or slowing down the intestinal transit*

Pairs of parallel sentences provide useful information on lexicon used, syntactic structures, stylistic features, etc., as well as the correspondences between the languages or registers. Hence, pairs built from different languages are widely used in machine translation, while pairs differentiated by the register of language can be used for the text simplification. The purpose of text simplification is to provide simplified versions of texts, in order to remove or replace difficult words or information. Simplification can be concerned with different linguistic aspects, such as lexicon, syntax, semantics, pragmatics and even document structure.

Automatic text simplification can be used as a preprocessing step for NLP applications or for producing suitable versions of texts for humans. In this second case, simplified documents are typically created for children (Vu et al., 2014), for people with low literacy or foreigners (Paetzold and Specia, 2016), for people with mental or neurodegenerative disorders (Chen et al., 2016), or for laypeople who face specialized documents (Leroy et al., 2013). Our work is related to the creation of simplified medical documents for laypeople, such as patients and their relatives. It has indeed been noticed that medical and health documents contain information that is difficult to understand by patients and their relatives, mainly because of the presence of technical and specialized terms and notions. This situation has a negative effect on the healthcare process (AMA, 1999; Mcgray, 2005; Rudd, 2013). Hence, helping patients to better understand medical and health information is an important issue, which motivates our work.

In order to perform biomedical text simplification, we propose to collect parallel sentences, which align difficult and simple information, as they provide crucial and necessary indicators for automatic systems for text simplification. Indeed, such pairs of sentences contain cues on transformations which are suitable for the simplification, such as lexical substitutes and syntactic modifications. Yet, this kind of resources is seldom available, especially in languages other than English. As a matter of fact, it is easier to access comparable corpora: they cover the same topics but are differentiated by their registers (documents created for medical professionals and documents created for patients). More precisely, we can exploit an existing monolingual comparable corpus with medical documents in French (Grabar and Cardon, 2018). The purpose of our work is to detect and align parallel sentences from this comparable corpus. We also propose to test what is the impact of imbalance on categorization results: imbalance of categories is indeed the natural characteristics in textual data.

The existing work on searching parallel sentences in monolingual comparable corpora indicates that the main difficulty is that such sentences may show low lexical overlap but be nevertheless parallel. Recently, this task gained in popularity in general-language domain thanks to the semantic text similarity (STS) initiative. Dedicated *SemEval* competitions have been proposed for several years (Agirre et al., 2013, 2015, 2016). The objective, for a given pair of sentences, is to predict whether they are semantically similar and to assign a similarity score going from 0 (independent semantics) to 5 (semantic equivalence). This task is usually explored in general-language corpora (Coster and Kauchak, 2011; Hwang et al., 2015; Kajiwara and Komachi, 2016; Brunato et al., 2016). Among the exploited methods, we can notice:

- lexicon-based methods which rely on similarity of subwords or words from the processed texts or on machine translation (Madnani et al., 2012). The features exploited can be: lexical overlap, sentence length, string edition distance, numbers, named entities, the longest common substring (Clough et al., 2002; Zhang and Patrick, 2005; Qiu et al., 2006; Nelken and Shieber, 2006; Zhu et al., 2010);

- knowledge-based methods which exploit external resources, such as WordNet (Miller et al., 1993) or PPDB (Ganitkevitch et al., 2013). The features exploited can be: overlap with external resources, distance between the synsets, intersection of synsets, semantic similarity of resource graphs, presence of synonyms, hyperonyms or antonyms (Mihalcea et al., 2006; Fernando and Stevenson, 2008; Lai and Hockenmaier, 2014);

- syntax-based methods which exploit the syntactic modelling of sentences. The features often exploited are: syntactic categories, syntactic overlap, syntactic dependencies and constituents, predicat-argument relations, edition distance between syntactic trees (Wan et al., 2006; Severyn et al., 2013; Tai et al., 2015; Tsubaki et al., 2016);

- corpus-based methods which exploit distributional methods, latent semantic analysis (LSA), topics modelling, word embeddings, etc. (Barzilay and Elhadad, 2003; Guo and Diab, 2012; Zhao et al., 2014; Kiros et al., 2015; He et al., 2015; Mueller and Thyagarajan, 2016).

There has been work for detection of paraphrases in French comparable biomedical corpora (Deléger and Zweigenbaum, 2009), but there is no work on building a corpus for text simplification in the biomedical domain. Our work is positioned in this area.

In what follows, we first present the linguistic material used, and the methods proposed. We then present and discuss the results obtained, and conclude with directions of future work.

2 Method

We use the CLEAR comparable medical corpus (Grabar and Cardon, 2018) available online[1] which contains three comparable sub-corpora in French. Documents within these sub-corpora are contrasted by the degree of technicality of the information they contain with typically specialized

[1] http://natalia.grabar.free.fr/resources.php#clear

and simplified versions of a given text. These corpora cover three genres: drug information, summaries of scientific articles, and encyclopedia articles. We also exploit a reference dataset with sentences manually aligned by two annotators.

2.1 Comparable Corpora

Table 1 indicates the size of the comparable corpus in French: number of documents, number of words (occurrences and lemmas) in specialized and simplified versions. This information is detailed for each sub-corpus: drug information (*Drugs*), summaries of scientific articles (*Scient.*), and encyclopedia articles (*Encyc.*).

The *Drugs* corpus contains drug information such as provided to health professionals and patients. Indeed, two distinct sets of documents exist, each of which contains common and specific information. This corpus is built from the public drug database[2] of the French Health ministry. Specialized versions of documents provide more word occurrences than simplified versions.

The *Scientific* corpus contains summaries of meta-reviews of high evidence health-related articles, such as proposed by the Cochrane collaboration (Sackett et al., 1996). These reviews have been first intended for health professionals but recently the collaborators started to create simplified versions of the reviews (*Plain language summary*) so that they can be read and understood by the whole population. This corpus has been built from the online library of the Cochrane collaboration[3]. Here again, specialized version of summaries is larger than the simplified version, although the difference is not very important.

The *Encyclopedia* corpus contains encyclopedia articles from Wikipedia[4] and Vikidia[5]. Wikipedia articles are considered as technical texts while Vikidia articles are considered as their simplified versions (they are created for children from 8 to 13 year old). Similarly to the works done in English, we associate Vikidia with Simple Wikipedia[6]. Only articles indexed in the medical portal are exploited in this work. From Table 1, we can see that specialized versions (from Wikipedia) are also longer than simplified versions.

Those three corpora have different degrees of parallelism: Wikipedia and Vikidia articles are written independently from each other, drug information documents are related to the same drugs but the types of information presented for experts and laypeople vary, while simplified summaries from the *Scientific* corpus are created starting from the expert summaries.

2.2 Reference Data

The reference data with aligned sentence pairs, which associate technical and simplified contents, are created manually. We have randomly selected 2x14 *Encyclopedia* articles, 2x12 *Drugs* documents, and 2x13 *Scientific* summaries. The sentence alignment is done by two annotators following these guidelines:

1. exclude identical sentences or sentences with only punctuation and stopword difference ;

2. include sentence pairs with morphological variations (e.g. *Ne pas dépasser la posologie recommandée.* and *Ne dépassez pas la posologie recommandée.* – both examples can be translated by *Do not take more than the recommended dose.*);

3. exclude sentence pairs with overlapping semantics, when each sentence brings own content, in addition to the common semantics;

4. include sentence pairs in which one sentence is included in the other, which enables many-to-one matching (e.g. *C'est un organe fait de tissus membraneux et musculaires, d'environ 10 à 15 mm de long, qui pend à la partie moyenne du voile du palais.* and *Elle est constituée d' un tissu membraneux et musculaire.* – *It is an organ made of membranous and muscular tissues, approximately 10 to 15 mm long, that hangs from the medium part of the soft palate.* and *It is made of a membranous and muscular tissue.*);

5. include sentence pairs with equivalent semantics – other than semantic intersection and inclusion (e.g. *Les médicaments inhibant le péristaltisme sont contre-indiqués dans cette situation.* and *Dans ce cas, ne prenez pas de médicaments destinés à bloquer ou ralentir le transit intestinal.* – *Drugs that inhibit peristalsis are contraindicated in*

[2] http://base-donnees-publique.medicaments.gouv.fr/

[3] http://www.cochranelibrary.com/

[4] https://fr.wikipedia.org

[5] https://fr.vikidia.org

[6] http://simple.wikipedia.org

corpus	*# docs*	*# occ$_{sp}$*	*# occ$_{simpl}$*	*# lemmas$_{sp}$*	*# lemmas$_{simpl}$*
Drugs	11,800x2	52,313,126	33,682,889	43,515	25,725
Scient.	3,815x2	2,840,003	1,515,051	11,558	7,567
Encyc.	575x2	2,293,078	197,672	19,287	3,117

Table 1: Size of the three source corpora. Column headers: number of documents, total number of occurrences (specialized and simple), total number of unique lemmas (specialized and simple)

		Specialized				*Simplified*				*Alignment*	
		source		*aligned*		*source*		*aligned*		*rate (%)*	
	# doc.	*# sent.*	*# occ.*	*# pairs.*	*# occ.*	*# sent.*	*# occ.*	*# pairs.*	*# occ.*	*sp.*	*simp.*
D	12x2	4,416	44,709	502	5,751	2,736	27,820	502	10,398	18	11
S	13x2	553	8,854	112	3,166	263	4,688	112	3,306	20	43
E	14x2	2,494	36,002	49	1,100	238	2,659	49	853	2	21

Table 2: Size of the reference data with consensual alignment of sentences. Column headers: number of documents, sentences and word occurrences for each subset, alignment rate

that situation. and *In that case, do not take drugs intended for blocking or slowing down the intestinal transit.*).

The judgement on semantic closeness may vary according to the annotators. For this reason, the alignments provided by each annotator undergo consensus discussions. This alignment process provides a set of 663 aligned sentence pairs. The inter-annotator agreement is 0.76 (Cohen, 1960). It is computed within the two sets of sentences proposed for alignment by the two annotators.

Table 2 shows the details of the manually aligned set of sentences. Because the three corpora vary in their capacity to provide parallel sentences, we compute their *alignment rate*. The alignment rate for a given corpus is the number of sentences that are part of an aligned pair relative to the total number of sentences. As expected, only a tiny fraction of all possible pairs corresponds to aligned sentences. We can observe that the *Scientific* corpus is the most parallel with the highest alignment rate of sentences, while the two other corpora (*Drugs* and *Encylopedia*) contain proportionally less parallel sentences. Sentences from simplified documents in the *Scientific* and *drugs* corpora are longer than sentences from specialized documents because they often add explanations for technical notions, like in this example: *We considered studies involving bulking agents (a fibre supplement), antispasmodics (smooth muscle relaxants) or antidepressants (drugs used to treat depression that can also change pain perceptions) that used outcome measures including improvement of abdominal pain, global assessment (overall relief of IBS symptoms) or symptom score.* In the *Encylopedia* corpus such notions are replaced by simpler words, or removed. Finally, in all corpora, we observe frequent substitutions by synonyms, like {*nutrition*, *food*}, {*enteral*, *directly in the stomach*}, or {*hypersensitivity*, *allergy*}. Notice that with such substitutions, lexical similarity between sentences is reduced.

The documents are pre-processed. They are segmented into sentences using strong punctuation (*i.e.* *.?!;:*). We removed, from each subcorpus, the sentences that are found in at least half of the documents of a given corpus. Those sentences are typically legal notices, section titles, and remainders from the conversion of the HTML versions of the documents. The lines that contain no alphabetic characters have also been removed. That reduces the total number of possible pairs for each document pair approximately from 940,000 to 590,000.

2.3 Automatic Detection and Alignment of Parallel Sentences

Automatic detection and alignment of parallel sentences is the main step of our work. The unity processed is a pair of sentences. The objective is to categorize the pairs of sentences in one of the two categories:

- alignment: the sentences are parallel and can be aligned;
- non-alignment: the sentences are non-parallel and cannot be aligned.

The reference data provide 663 positive examples (parallel sentence pairs). In order to perform the automatic categorization, we also need negative examples, which are obtained by randomly pairing all sentences from all the document pairs except the sentence pairs that are already found to be parallel. Approximately 590,000 non-parallel sentences pairs are created in this way. That high degree of imbalance is the main challenge in our work and we address it in the experimental design (sec 2.4).

For the automatic alignment of parallel sentences, we first use a binary classification model that relies on the random forests algorithm (Breiman, 2001). The implementation we use is the one that is available in scikit-learn (Pedregosa et al., 2011). Our goal is to propose features that can work on textual data in different languages and registers. We use several features which are mainly lexicon-based and corpus-based, so that they can be easily applied to textual data in other corpora, speacialized areas and languages or transposed on them. The features are computed on word forms (occurrences). The features are the following:

1. *Number of common non-stopwords*. This feature permits to compute the basic lexical overlap between specialized and simplified versions of sentences (Barzilay and Elhadad, 2003). It concentrates on non-lexical content of sentences;

2. *Percentage of words from one sentence included in the other sentence, computed in both directions*. This features represents possible lexical and semantic inclusion relations between the sentences;

3. *Sentence length difference between specialized and simplified sentences*. This feature assumes that simplification may imply stable association with the sentence length;

4. *Average length difference in words between specialized and simplified sentences*. This feature is similar to the previous one but takes into account average difference in sentence length;

5. *Total number of common bigrams and trigrams*. This feature is computed on character ngrams. The assumption is that, at the sub-word level, some sequences of characters may be meaningful for the alignment of sentences if they are shared by them;

6. *Word-based similarity measure exploits three scores (cosine, Dice and Jaccard)*. This feature provides a more sophisticated indication on word overlap between two sentences. Weight assigned to each word is set to 1;

7. *Character-based minimal edit distance* (Levenshtein, 1966). This is a classical acception of edit distance. It takes into account basic edit operations (insertion, deletion and substitution) at the level of characters. The cost of each operation is set to 1;

8. *Word-based minimal edit distance* (Levenshtein, 1966). This feature is computed with words as units within sentence. It takes into account the same three edit operations with the same cost set to 1. This feature permits to compute the cost of lexical transformation of one sentence into another;

9. *WAVG*. This features uses word embeddings. The word vectors of each sentence are averaged, and the similarity score is calculated by comparing the two resulting sentence vectors (Stajner et al., 2018);

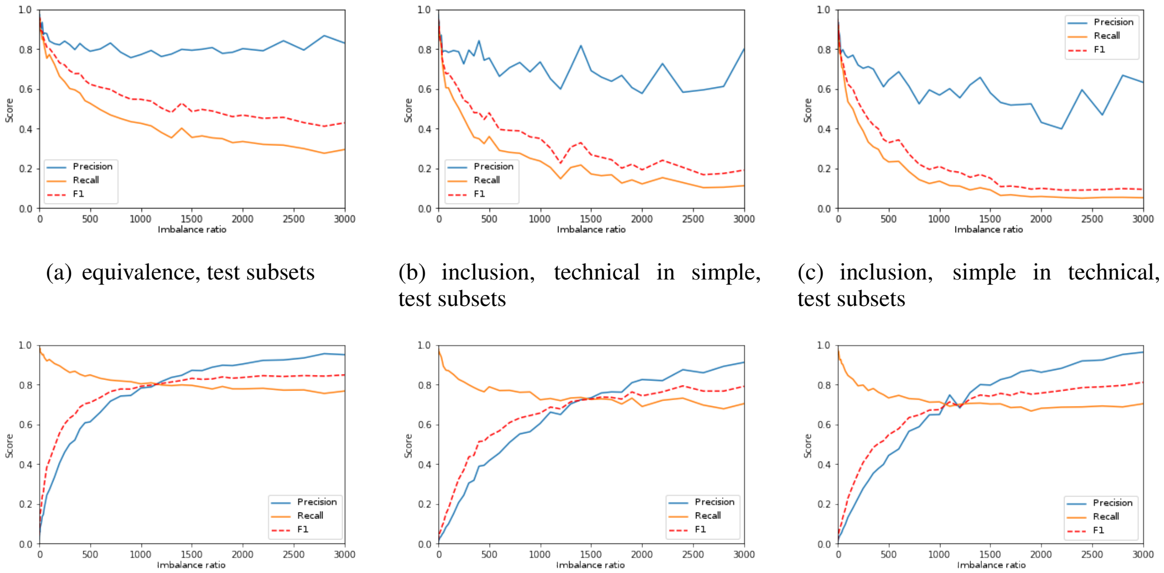
10. *CWASA*. This feature is the continuous word alignment-based similarity analysis, as described in (Franco-Salvador et al., 2016).

For the last two features, we trained the embeddings on the CLEAR corpus using word2vec (Mikolov et al., 2013), and the scores are computed using the CATS tool (Stajner et al., 2018).

2.4 Experimental Design

The set with manually aligned pairs is divided into three subsets:

- *equivalence*: 238 pairs with equivalent semantics,
- *tech in simp*: 237 pairs with inclusion where the content of technical sentence is fully included in simplified sentence, and simplified sentence provides additional content,
- *simp in tech*: 112 pairs with inclusion where the content of simplified sentence is fully included in technical sentence, and technical sentence provides additional content.



(a) equivalence, test subsets

(b) inclusion, technical in simple, test subsets

(c) inclusion, simple in technical, test subsets

(d) equivalence, real data

(e) inclusion, technical in simple, real data

(f) inclusion, simple in technical, real data

Figure 1: Precision, Recall and F-1 for the various experiments and subsets

For each subset, we perform two sets of experiments:

1. We train and test the model with balanced data (we randomly select as many non-aligned pairs as aligned pairs), and then we progressively increase the number of non-aligned pairs until we reach a ratio of 3000:1, which is close to the real data (∼4000:1).

2. Then, for each ratio, we apply the obtained model to the whole dataset and evaluate the results. Note that the training data is included in the whole dataset, we proceed this way because of the low volume of available data.

As there is some degree of variability coming with the subset of non-aligned pairs that are randomly selected for the imbalance ratio, every single one of those experiments has been performed fifty times: the results that are presented correspond to the mean values over the fifty runs.

2.5 Evaluation

For evaluating the results, in each experiment we divide the indicated datasets in two parts: two thirds for training and one third for testing. The metrics we use are Recall, Precision and F1 scores. As we are primarily focused on detection of the aligned pairs, we only report scores for that class. Another reason to exclude the negative class and the global score from the observations is that when the data are imbalanced (negative class is growing progressively), misclassifying the positive data has little influence over the global scores, which thus always appear to be high (metrics above 0.99).

Finally, we apply the best model for equivalent pairs on another 30 randomly selected documents and evaluate the output.

3 Presentation and Discussion of Results

We present the results in Figure 1: The x axis represents the growing of imbalance (the first position is 1 and corresponds to balanced data), while the y axis represents the values of Precision, Recall and F-measure. The results for the three subsets are presented: equivalence (Figures 1(a) and 1(d)), inclusion of technical sentence in simple sentence (Figures 1(b) and 1(e)), and inclusion of simple sentence in technical sentence (Figures 1(c) and 1(f)). Besides, Figures 1(a), 1(b) and 1(c) present the results obtained by training and testing the model on the dataset with the same imbalance ratio (first set of experiments described in section 2.4). As for Figures 1(d), 1(e) and 1(f), they present the results obtained by the models mentioned above that are applied on the whole set of manually annotated data (second set of experiments described in section 2.4).

	equivalence	simp. in tech.	tech. in simp.	intersection	false positives
nb. of pairs	56	10	4	9	1
ratio	70%	12.5%	5%	11.25%	1.25%

Table 3: Breakdown by pair types of the output of the model trained on equivalent pairs with an imbalance ratio of 1200:1 and applied to 30 randomly chosen pairs of documents

The most visible conclusion we can draw from those experiments is that equivalent pairs (Figures 1(a) and 1(d)) are easier to classify than inclusion pairs (the rest of the Figures). Values of both, Precision and Recall, are higher on the equivalence dataset at different imbalance points. For instance, with training on the equally balanced dataset (position 1 on Figure 1(a)), the scores for Precision (0.98) and Recall (0.95) are higher than the scores obtained by the technical in simple dataset (0.96 Precision and 0.94 Recall) and the simple in technical dataset (0.95 Precision and 0.93 Recall) at the same point. For the application to the real data, for ratio 1200:1 – the point where Precision and Recall meet for equivalent pairs, see Figure 1(d) – we obtain 0.81 Precision and 0.81 Recall. At that same ratio, for the technical in simple pairs the scores are 0.65 Precision and 0.73 Recall, and for the simple in technical pairs Precision is 0.73 and Recall is 0.70. This result is positive because the equivalence dataset usually provides the main and the most complete information on transformations required for the simplification. As for the inclusion relations, they cover a large variety of situations which do not necessarily correspond to the searched information. This is illustrated by the unstability of the curves in Figures 1(b) and 1(c), whereas they are smooth in Figure 1(a). The negative examples subset seems to have a quite high influence on the results, which indicates that it is more difficult to draw a clear separation between positive and negative examples. We need to design additional processing steps to be able to classify those pairs in a more efficient way.

We can also observe from Figures 1(a), 1(b) and 1(c) that the use of balanced data provides very high results, both for Precision and Recall, which are very close to the reference data (> 0.90 performance). This is true for the three subsets tested (equivalence and inclusions). These good results in an artifical setting cannot be applied to the real dataset, as is indicated by the starting point in Figures 1(d), 1(e) and 1(f). Yet, the imbalance has greater effect on the inclusion datasets, while again the equivalence dataset resists better. An interesting fact is that, when the model is learned on a substantial degree of imbalance, the Precision score is high when that model is applied to the real data, which has a ratio of about 4,000:1. This is interesting because it shows that the model is particularly good at discriminating counter-examples. The recall value is also high, but since two thirds of the real data examples have been used for training, that good score should be considered cautiously. We are planning to evaluate the models on a separate set of manually annotated documents. This is still a good result, as during the tests that we performed with other classification algorithms, the models did not successfully recognize the examples they had seen during training.

For further evaluation, we randomly selected 30 pairs of documents to evaluate the performances of the models. We used the model that was trained at a ratio of 1200:1 on equivalent pairs. In terms of precision, the model shows 98.75% on all the sentence pairs aligned (80 sentence pairs), including equivalence, inclusions and intersection. Table 3 shows the breakdown of this output in terms of pair types: 70% (56 pairs) are equivalent pairs, 29% (23 pairs) are examples of inclusion (10 simple in technical, 4 technical in simple) and intersection (9), and one pair contains two unrelated sentences. Those results show that we have a model that can be used to automatically generate a parallel corpus with reduced noise, from highly imbalanced comparable corpus, for text simplification purposes.

4 Conclusion and Future Work

We addressed the task of detection and alignment of parallel sentences from a monolingual comparable French corpus. The comparable aspect is on the technicality axis, as the corpus contrasts technical and simplified versions of information on the same subjects. We use the CLEAR corpus, that is related to the biomedical area.

Several experiments were performed. We divide the task in three subtasks – equivalent pairs, and inclusion on both directions – and make observations on the effect of imbalance during training on the performance on the real data. We show that increasing the imbalance during training increases the Precision of the model while still maintaining a stable value for Recall. We also find that the task is easier to perform on sentence pairs that have the same meaning, than on sentence pairs where one is included in the other.

We will use that model to generate a corpus of parallel sentences in order to work on the development of methods for biomedical text simplification in French. We will also perform experiments on the general language. Another task we will explore addresses the question on how that model performs with the cross-lingual transfer of descriptors and models.

5 Acknowledgements

This work was funded by the French National Agency for Research (ANR) as part of the *CLEAR* project (*Communication*, *Literacy*, *Education*, *Accessibility*, *Readability*), ANR-17-CE19-0016-01. The authors would like to thank the reviewers for their helpful comments.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval 2015*, pages 252–263.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval 2016*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and WeiWei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In **SEM*, pages 32–43.

AMA. 1999. Health literacy: report of the council on scientific affairs. Ad hoc committee on health literacy for the council on scientific affairs, American Medical Association. *JAMA*, 281(6):552–7.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *EMNLP*, pages 25–32.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Dominique Brunato, Andrea Cimino, Felice Dell'Orletta, and Giulia Venturi. 2016. PaCCSS-IT: A parallel corpus of complex-simple sentences for automatic text simplification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 351–361, Austin, Texas. Association for Computational Linguistics.

Ping Chen, John Rochford, David N. Kennedy, Soussan Djamasbi, Peter Fay, and Will Scott. 2016. Automatic text simplification for people with intellectual disabilities. In *AIST*, pages 1–9.

Paul Clough, Robert Gaizauskas, Scott S.L. Piao, and Yorick Wilks. 2002. METER: Measuring text reuse. In *ACL*, pages 152–159.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA. Association for Computational Linguistics.

Louise Deléger and Pierre Zweigenbaum. 2009. Extracting lay paraphrases of specialized expressions from monolingual comparable medical corpora. In *Proceedings of the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora (BUCC)*, pages 2–10, Singapore. Association for Computational Linguistics.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Comp Ling UK*, pages 1–7.

Marc Franco-Salvador, Parth Gupta, Paolo Rosso, and Rafael E. Banchs. 2016. Cross-language plagiarism detection over continuous-space- and knowledge graph-based representations of language. *Knowledge-Based Systems*, 111:87 – 99.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL-HLT*, pages 758–764.

Natalia Grabar and Rémi Cardon. 2018. CLEAR – Simple Corpus for Medical French. In *Workshop on Automatic Text Adaption (ATA)*, pages 1–11.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *ACL*, pages 864–872.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586, Lisbon, Portugal.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard Wikipedia to simple Wikipedia. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 211–217, Denver, Colorado. Association for Computational Linguistics.

Tomoyuki Kajiwara and Mamoru Komachi. 2016. Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1147–1158, Osaka, Japan. The COLING 2016 Organizing Committee.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Neural Information Processing Systems (NIPS)*, pages 3294–3302.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Workshop on Semantic Evaluation (SemEval 2014)*, pages 239–334, Dublin, Ireland.

Gondy Leroy, David Kauchak, and Obay Mouradi. 2013. A user-study measuring the effects of lexical simplification and coherence enhancement on perceived and actual text difficulty. *Int J Med Inform*, 82(8):717–730.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics. Doklady*, 707(10).

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *NAACL-HLT*, pages 182–190.

A Mcgray. 2005. Promoting health literacy. *J of Am Med Infor Ass*, 12:152–163.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, pages 1–6.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1993. Introduction to wordnet: An on-line lexical database. Technical report, WordNet.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI Conference on Artificial Intelligence*, pages 2786–2792.

Rani Nelken and Stuart M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *EACL*, pages 161–168.

Gustavo H. Paetzold and Lucia Specia. 2016. Benchmarking lexical simplification systems. In *LREC*, pages 3074–3080.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Empirical Methods in Natural Language Processing*, pages 18–26, Sydney, Australia.

ED Rudd. 2013. Needed action in health literacy. *J Health Psychol*, 18(8):1004–10.

David L. Sackett, William M. C. Rosenberg, Jeffrey A. MuirGray, R. Brian Haynes, and W. Scott Richardson. 1996. Evidence based medicine: what it is and what it isn't. *BMJ*, 312(7023):71–2.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *Annual Meeting of the Association for Computational Linguistics*, pages 714–718.

Sanja Stajner, Marc Franco-Salvador, Simone Paolo Ponzetto, and Paolo Rosso. 2018. Cats: A tool for customised alignment of text simplification corpora. In *Proceedings of the 11th Language Resources and Evaluation Conference, LREC 2018, Miyazaki, Japan, May 7-12*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566, Beijing, China.

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2016. Non-linear similarity learning for compositionality. In *AAAI Conference on Artificial Intelligence*, pages 2828–2834.

Tu Thanh Vu, Giang Binh Tran, and Son Bao Pham. 2014. Learning to simplify children stories with limited data. In *Intelligent Information and Database Systems*, pages 31–41.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using dependency-based features to take the para-farce out of paraphrase. In *Australasian Language Technology Workshop*, pages 131–138.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *Australasian Language Technology Workshop*, pages 160–166.

Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *COLING 2010*, pages 1353–1361.

Classifying Author Intention for Writer Feedback in Related Work

Arlene Casey
School of Informatics
University of Edinburgh
Edinburgh, UK
a.j.casey@sms.ed.ac.uk

Bonnie Webber
School of Informatics
University of Edinburgh
Edinburgh, UK
bonnie@inf.ed.ac.uk

Dorota Głowacka
Dept. of Computer Science
University of Helsinki
Helsinki, Finland
glowacka@cs.helsinki.fi

Abstract

The ability to produce high-quality publishable material is critical to academic success but many Post-Graduate students struggle to learn to do so. While recent years have seen an increase in tools designed to provide feedback on aspects of writing, one aspect that has so far been neglected is the *Related Work* section of academic research papers. To address this, we have trained a supervised classifier on a corpus of 94 *Related Work* sections and evaluated it against a manually annotated gold standard. The classifier uses novel features pertaining to citation types and co-reference, along with patterns found from studying *Related Works*. We show that these novel features contribute to classifier performance with performance being favourable compared to other similar works that classify author intentions and consider feedback for academic writing.

1 Introduction

Argument structures are key in allowing an author to construct a persuasive message that realizes the author's intention. The automatic identification of such intentions has been shown to be a valuable resource in areas such as summarising information (Teufel and Moens, 2002; Cohan and Goharian, 2015), and understanding citation function and sentiment (Teufel et al., 2006; Jurgens et al., 2018). Recent years have seen more academic writing tools focused on content that use an understanding of expected author intentions to assist in feedback. This is an important resource for Post-Graduate (PG) students who struggle to gain the necessary skills in academic writing that are critical to their success (Aitchison et al., 2012; Paltridge and Starfield, 2007). Automating understanding of author intentions is challenging as research articles, whilst classed in their own genre, are known to differ in content and linguistic style across disciplines (Hyland, 2008).

Despite these challenges, previous work using author intentions has been successful in automating writing feedback though largely focused on the Abstract (Feltrim et al., 2006) or the Introduction (Cotos and Pendar, 2016; Abel, 2018). One reason for this focus on a single section of a research paper is that each section has its own purpose, which encourages different linguistic practices. Existing tools may concentrate on the Introduction due to formative work done by Swales (1990). Swales was one of the first to recognise these intentions in academic writing calling them *rhetoric intentions*. Swales showed how linguistic patterns in the Introduction could be matched to intentions, such as *establishing a territory* or *defining the problem*.

One section of academic papers that has not yet been explored for automated content feedback is the *Related Work* section. One particular challenge students have when learning to write is to find and project their own viewpoint (Kamler and Thomson, 2006). Often their lack of experience and confidence in projecting their own voice amongst established scholars results in students making bland statements about others' work. Such bland statements in the *Related Work* section do nothing more than provide a list of work with no real critical commentary or attempt to relate it to the author's own work. We address this gap of content feedback for *Related Work* by building a model of author intentions that one expects to find in *Related Work*.

We show how the labels of an annotated corpus for author intention, designed for writer feedback in *Related Work*, can be identified reliably. We build on existing methods for feature representation of author intentions and show that the novel features we introduce contribute significantly to the classifier performance, improving on performance of existing writer feedback tools.

2 Related Work

Automating Author Intentions – Previous models of author intentions in research articles have been successfully automated. One of the first and widely used is Teufel (1999) who proposed Argument Zoning (AZ) which labels sentences with zones representing the rhetoric purpose (author intent) within the global context of a document e.g. background, aim or conclusion. Further work has applied this schema to biology papers (Mizuta and Collier, 2004), with a modified, finer grained approach applied to papers on chemistry (Teufel et al., 2009). Liakata et al. (2012) took a different approach to labelling author intentions, studying the conceptual structure of biology articles treating the article as an investigation. Fisas et al. (2015) develop a schema based on both Liakata and Teufel's work to represent scientific concepts that appear in computer graphics articles. These works have successfully identified author intentions, but they differ from our work by seeking intentions in a global context across a whole article. For example, AZ was developed to support summarisation and information access. The author intentions that these activities would be associated with are rarely found in a *Related Work* section and are unlikely to be helpful in writing feedback for this section. They are nevertheless useful in supporting writing feedback on Abstracts and summaries of PhDs (Feltrim et al., 2006).

Related Work does have in common with other sections the fact that it should contain citations. Understanding the motivations or function of a citation can help determine an author intention (Teufel et al., 2006). Work on citation function has been an area of research for several decades (Weinstock, 1971; Oppenheim and Renn, 1978; Teufel et al., 2006; Angrosh et al., 2012), with more recent work considering how this recognition can be automated. Jurgens et al. (2018) investigates the framing of citations and how this can be used to study the evolution of a field. Teufel et al. (2006) work on automated recognition of citation function and show a strong relationship between function and sentiment. One work that specifically looks at context identification of sentences in *Related Work* is (Angrosh et al., 2010). This work focuses on sentences in terms of their ability to support intelligent information retrieval in digital library services. While aspects of this work and the previous works on citation function are relevant, what is missing is the need to identify where an author talks about their own work in context to other work, showing why it is different or how it fills a gap. As discussed in the Introduction, one of the problems with poor writing in *Related Work* is bland statements that provide lists of citations. Cited works should be ones that have implications for the author's work (Maxwell, 2006). To provide such feedback, we must capture this context in addition to citation function.

Recognising Author Intentions – Specific phrasing has been shown to function in structuring discourse by guiding readers through a text (Hyland, 2012) and can be found to align to sections, such as the Introduction or Results. Most previous work in automating author intentions have utilised these patterns as part of their feature set. The early work of Teufel (1999) (in the domain of computational linguistics) uses cue phrases and lexical patterns that involve parts of speech and citation markers as features. Jurgens et al. (2018) shows how applying bootstrapping to Tuefel's lexicon improves citation function recognition.

Verbs have been shown to have a role in understanding citation function by determining rhetorical and semantic levels. Verbs used to report can show positive and negative aspects of evaluation in cited works and differentiate between intentions e.g in Angrosh et al. (2010) they use reporting verbs that describe (examine, propose), refer to an outcome (develop, show) or show a strength (improve). Citation forms (Swales, 1990) of integral and non-integral have been shown to be a contributing feature to author intention recognition, with studies of novice writers showing that they use a limited range of citation types (Thompson and Tribble, 2001). Our approach also uses linguistic patterns, verb types and citation types to support building our feature set, and we do this within one domain, computational linguistics (cf. Section 4).

Automated Assessment of Writing – Existing, academic writing tools have focused on identifying author intentions, such as those described by Swales (1990), that can be found in an Introduction (Cotos and Pendar, 2016; Anthony and

V. Lashkia, 2003; Abel, 2018). The Criterion online writing service, focuses on automated persuasive essay evaluation and uses recognition of discourse elements based on aspects such as supporting ideas, introductions and conclusion (Burstein et al., 2003, 2004). Several other works have focused on identifying argument components and relations and how these relate to essay scores (Ghosh et al., 2016; Song et al., 2014). Recognizing argument components in this case focuses on premises and claims largely based on the Toulmin model of argumentation (Toulmin, 2003) which is a different approach to ours. In addition, all this work focuses on feedback for persuasive essays which will differ in linguistic practices found in scientific papers and from the author intention structure of a *Related Work*. Overall, whilst aspects may be relevant in general, these methods would not facilitate the kind of content feedback that would help a writer with *Related Work*.

3 Author Intentions to Support Feedback

3.1 Annotation Schema for Data

The need for annotated data is something that previous methods have in common, each using an annotation schema that supports the intentions they seek. It is known that annotation schemas benefit from being task-orientated (Guo et al., 2010). We use an annotation schema developed to recognise author intentions in *Related Work* sections and provide authors with useful feedback (Casey et al., 2019). This schema uses qualities that should be present in *Related Work* sections, following (Kamler and Thomson, 2006). Qualities group into four areas: **Background** – helps the author locate their work in the field, demonstrating they understand their field and its history through indicating seminal works and relevant research fields; **Cited works** – in addition to generally identifying the field, the author should demonstrate specifically (i) which works are most pertinent to their work, (ii) how these works have influenced them and (iii) if and how the current works build on or use these methods; **Gap** – make clear what the gap is and what has specifically not been addressed; **Contribution** – having exposed the gap, the author should identify their contribution. This schema tries to isolate neutral citations that provide mere description, compared to those that highlight gaps or problems, along with identifying where an author talks about their own work and how this relates to the cited work or background in general. The sentence label schema we use can be found in Table 1, and we indicate which of the qualities each label falls into.

3.2 Annotated Dataset

The annotated dataset in (Casey et al., 2019) is composed of papers from the ACL anthology (Bird et al., 2008) that have been pre-annotated for citations and co-reference to the author's own work by (Schäfer et al., 2012). We use 94 papers with *Related Work* sections after removing one due to OCR issues. All papers were conference papers 6-8 pages in length. The authors report annotator agreement, based on Cohen Kappa (Cohen, 1960) at 0.77, which increased to 0.85 following a round of discussion.

3.3 Challenges and Changes to the Schema to Support Automation

Previous works have largely been based on annotating at a sentence level but some works have considered a smaller discourse unit such as (Shatkay et al., 2008). This smaller discourse unit does allow for instances where an author may encode two intentions in one sentence. The data we use is labelled on a sentence basis. The authors in (Casey et al., 2019) acknowledge that this may not be satisfactory as some sentences could be multi-labelled, such as where an author highlights a gap then state their contribution. Just as this is challenging for an annotator to label, it may be more so for an automated classifier.

From the annotated data, some categories were rare and were collapsed into more frequent categories. The distinction these rare labels offered was not necessary. In particular, the two labels that denoted the author said something positive about a citation/field were merged into the appropriate cited/field evaluation categories. We merged the two categories (author's work builds on/adapts/uses X, and author's work is similar to X) into one category. Finally, comparison of two cited works was merged to cited work description. Table 2 shows the distribution of the labels in the 94 *Related Work* sections. We abbreviate some of the labels from the original schema.

Quality	Label	Description
Background	BG-NE	Description of the state of the field, describing/listing known methods or common knowledge. No evidence i.e. citation is not included
	BG-EP	As above but evidence provided i.e.citation included
	BG-EV	Positive, shortcoming, problem or gap in the field
Cited Works	CW-D	Describes cited work, this could be specific details, or very high level details or nothing more than a reference for further information
	A-CW-U	Author's work uses/builds/similar to a cited work
	CW-EV	Positive, shortcoming, problem or gap about the cited work
Gap and Contribution	A-D	Author describes their work with no linguistic marking to other's work or being different
	A-GAP	Author specifically says they address a gap or highlights the novelty of their work
	A-CW-D	Author's highlights how their work is different to cited work
	TXT	Sentence provides information about what will be discussed in the next section

Table 1: Sentence Labels

Sentence Label	Count
BG-EV	90
BG-NE	257
BG-EP	171
CW-EV	133
CW-D	707
A-CW-U	59
A-D	107
TXT	21
A-CW-D	151
A-GAP	59
Total	1755

Table 2: Label class distribution

4 Methods

4.1 Classifiers

All models are trained using LibSVM (Chang and Lin, 2011) with a linear kernel and default settings. SVM's are known to be robust to over-fitting, and perform well in document classification tasks when features are sparse and the set of them is large. SVM does not assume statistical independence, making it a more suitable method when features may be overlapping or interdependent. Initially, we experimented with decision trees. However, when we tested multiple iterations for reliability, both Random Forest (Breiman, 2001) and C4.5 (Sumner et al., 2005) were not only consistently lower in performance (12%) but rare categories showed large variation (15%) between iterations and in some instances labels would not classify. We believe this was due to feature overlap and the problem previously discussed with some labels being multi-class. Due to the unreliability of its performance, we did not pursue the decision tree approach further.

4.2 Features

Features were motivated by other works that classify author intention and citation function, and were extracted on a sentence basis. We use a vector of sentence features as the input to our classifier. The following list summarises the features used in this work.

- **Structural** Positional information, such as relative sentence position, has been useful for identifying background sentences, as these are more likely to occur in an Introduction or *Related Work* than a Results section (Teufel, 1999; Jurgens et al., 2018; Liakata et al., 2012). We do not include relative sentence position but instead use a binary indicator for paragraph start and end sentence, manually added from the original PDF. This is similar to the feature in (Teufel and Moens, 2002) of paragraph structure. We expect this to work like a sentence relative position, as many background statements will come at the start of paragraphs, and towards the end of paragraphs authors will be more likely to relate their own work.

- **Citations Type Features** Authorial and parenthetical citations (Swales, 1990) have been shown to be useful in determining author intention. We build a parser to identify three types of citation: (i) those that form part of the syntax of the sentence (authorial); (ii) those that refer to the name of a system or known algorithm; and (iii) those that provide supporting evidence, found in parenthetical with no syntax e.g **in (Smith, 1990)** although in parenthesis would be of type (i). This slightly differing approach we believe will help to discriminate between background sentences with citation evidence and citation description sentences. We also take a count of type 1 and 2 citations and a separate count of type 3 citations.

- **N-grams** Work based on a much larger corpus than ours show that n-grams contribute significantly to the performance of their classifier. Liakata et al. (2012) show a 40% contribution and Cotos and Pendar (2016) work is mainly based on n-gram features of 650 Introductions. While our corpus is much smaller (Related Work from 94 articles), we nevertheless include binary values for bigrams and trigrams occurring with a frequency of ≥5. We do not remove stop words.

- **Co-referencing Features** Often discussion about a work or the author's work will be carried out over several sentences. The later sentences can have co-references to the original citation such as *'this paper' 'this model'* However, as Teufel (1999) shows, determining what she calls agents (e.g US_AGENT - 'our paper'), these co-reference phrases can be ambiguous. For example does *'this paper'* mean the previously cited paper or it is referencing the author's work. We take a different approach and use the annotations in our corpus for (i) references to the authors own work, (ii) cited work. In addition, we manually mark co-reference to multiple works in background sentences e.g *'previous work has been done in the area of ..'* and co-reference to previously cited work e.g. *'these previously mentioned works above'*

- **Verb Features** We use part of speech (POS) tags to identify verbs, treating the six possible VB tags (VB, VBD, VBG, VPN, VBP, VBZ) as binary features of being present or not in a sentence. Having substituted the co-references, described above, in a sentence we then parse for dependency extracting subject and object verb pairs in every sentence.

- **Linguistic Patterns** Teufel (1999) makes available a list of patterns containing cue phrases/words, lexical categories, constrained by PoS tags, developed on computational linguistic literature. Like (Jurgens et al., 2018) we use this list and adapt it manually using patterns we observe in *Related Work*. For example, one aspect we consider is contrasts that occur at the beginning of a sentence and those that happen mid sentence, creating lexical expressions to capture these. We also produce finer grained lexicon patterns for discourse connectives as these are indicative of a continuation sentence. Within those patterns we include citation types and co-references as described above.

- **Sentiment** We use our adapted version of Teufel's list to identify positive and negative words (e.g *advantageous* - positive adjective, *inaccurate* - negative adjective). In addition, we use the 82 polar phrases found in (Athar, 2014). We parse each sentence and count the positive and negative words.

- **Counts** Counts of sentence words, nouns, adverbs, discourse connectives.

- **Subject** We assign a sentence subject label before assigning a sentence label to decide if the sentence is about a citation, background or field information, author's work, or a combination of author's work and cited work. This subject feature is based on rules of sentence and previous sentence features e.g our finer grained approach to discourse connectives in conjunction with co-reference markers help us to understand subject.

5 Experimental Setup and Evaluation

5.1 Baseline

We provide two baselines, one with n-gram features only and one with all features based on the majority class.

5.2 Evaluation

Our work is similar to other automated classifications but not directly comparable as schemas and experimental settings differ. Our results are more comparable to the works of (Teufel, 1999; Jurgens et al., 2018; Teufel and Kan, 2011) as we use the same pattern list from (Teufel, 1999) as a starting point. These works use Naive Bayes, Random Forest and Maximum Entropy as classifier methods. We report their published Macro F1 scores, range of F1 scores for labels and the number of labels in the schema for comparison (Table

Features	Prec	Recall	F1	Acc%
ALL	.69 (.005)	.7 (.004)	.7 (.005)	70 (.48)
Cotos(2016)	.686	.549	.61	72.9%
Teufel(2011)	.478	.376	.4142	66.8%

Table 3: Classifier Performance, Mean scores after 10 iterations, Variance in brackets

System	Macro F1/Range	Label No.
(Teufel and Kan, 2011)	0.41 (0.19-0.81)	8
(Jurgens et al., 2018)	0.53	6
(Teufel, 1999)	0.68 (0.28-0.86)	12
(Cotos and Pendar, 2016)	0.61 (0.36-0.85)	17
Our Work		
-all feat	0.70 (.25 -0.88)	10
- no novel feat	0.54 (.15-.87)	
Baseline		
Ngram(B,T)	0.39 (.02-.68)	
Majority	0.57 (-)	

Table 4: Classifier Comparison, * significant 0.01

4). Also included is the work of (Cotos and Pendar, 2016) which focuses on writer feedback for Introductions. This is a much larger corpus using 650 annotated Introductions but fewer features, focusing on unigram and trigrams. However, it also uses SVM for classification. Where available, we also report Precision, Recall and Accuracy from these works to compare against our best performing model in Table 3.

Reliability of our model is important to ensure consistent results. Therefore, in addition to 10-fold cross validation, we carry out 10 iterations of the All features model, reporting on mean precision, recall, F1, Accuracy and variance in Table 3. None of our iterations produced significantly different results, demonstrating reliability and low variation. Significance, where noted, is tested with corrected t-test, p <0.01, (Nadeau and Bengio, 1999).

We also look at the features in our model and how they influence the label F1 scores with leave one out (LOO), which highlights the performance decrease when a single feature is omitted and single features (SF), which highlights the contribution of a single feature to performance. Looking at individual label features is important as having just one label perform poorly, such as being able to recognise an author gap sentence or where an author says how their work is different, will impact our goal of giving writer feedback.

6 Results

6.1 Classifier Performance

We compare our results to those mentioned in Section 5, Table 4. Comparing F1 scores overall, we outperform the other systems by a reasonable margin. In addition, the range of F1 scores for our labels are also similar to other systems. We outperform the work of Cotos and Pendar (2016), who looks at classification for Introduction feedback, despite their work being based on a bigger annotated corpus. We significantly outperform both our baselines of n-grams and majority class. We rerun our classification (no novel feat) removing the manual additions we added to the original pattern list of Teufel (1999), removing co-references and subject labels. This results in lower performance, significant (p <0.01) than our all features and our majority baseline.

6.2 Feature Contribution

Here we examine feature contributions by single feature and leave one out cross validation runs (Table 5). For each category, we highlight the lowest score in bold, which corresponds to the feature being left out. We place in brackets any scores higher than the All features model.

More frequently occurring categories, cited work description (CW-D), background sentences with and without evidence (BG-NE, BG-EP) are more robust to feature omissions. Features are not independent, so many of the patterns cover the n-gram features which may be why leaving out n-grams has less impact than expected. In the lower part of the table, n-grams as a single feature contributes most to labels TEXT and CW-DESC. Compared to other works that have used n-grams, our size is much smaller at <3000, whereas Liakata et al. (2012) used ∼42000 and Cotos and Pendar (2016) had ∼27000. It would be expected in a much larger corpus that n-grams will contribute more as a feature.

The removal of the paragraph start and end markers makes relatively little difference, with the exception of the author gap category (A-GAP): Being a rare category, this addition although small is important. Sentiment contributes in a small way to performance but particularly in the evaluation labels (BG-EV, CW-EV) as expected. Surprisingly, sentiment contributes to the text label. However, within text-labelled sentences, both these counts are zero, which may explain why it con-

Features	BG-EV	BG-NE	BG-EP	CW-EV	CW-D	A-CW-U	A-D	TXT	A-CW-D	A-GAP
ALL	.39	.72	.73	.53	.84	.48	.47	.88	.63	.25
Feat-(LOO)										
-subject	**.33**	.62	.71	.51	**.81**	.49	.41	.85	.64	**.22**
-n-grams	.33	**.70**	**.70**	.53	.84	(.50)	**.39**	.83	.62	.25
-verbtense	.35	.71	.72	.51	.84	.48	.46	.88	**.66**	.32
-sentiment	.34	.71	.71	**.50**	.84	**.46**	.43	**.67**	**.61**	(.28)
-counts	(.40)	.72	.73	.52	.84	(.50)	.46	.87	(.64)	.26
-Tot cit count	.38	.71	(.74)	(.54)	(.85)	.49	(.48)	.88	.64	.26
-paragraph	(.40)	.71	.73	(.54)	.84	.49	.47	.87	.62	**.22**

Features	BG-EV	BG-NE	BG-EP	CW-EV	CW-D	A-CW-U	A-D	TXT	A-DIFF	A-GAP
ALL	.39	.72	.73	.53	.84	.48	.47	.88	.63	.25
Feat-(SF)										
-patterns	.30	.54	(.74)	.41	.77	(.57)	(.48)	.80	(.65)	.26
-subject	-	.58	-	-	.80	-	.45	.75	.46	-
-sub+patt+dep	.31	.72	.73	.47	.83	(.55)	.46	.84	.63	(.27)
-n-grams	.11	.31	.21	.24	.62	.23	.04	.68	.39	.02

Table 5: F-Measures for Features and Labels,10 cross validation

tributes here. Neither of our evaluations labels (BG-EV, CW-EV) perform as well as expected. These two labels are merged from the annotation schema, positive and shortcoming/problem into one evaluation label. These original labels are both quite different linguistically and we speculate that this might prove difficult for the classifier.

Total citation counts and counts of adverbs, words, nouns and discourse connectives seem to actually make the performance of the classifier worse on many of the labels, although not significantly so. There is an overlap in total citation counts with the count of our citation types perhaps indicating this feature could be omitted.

We note that the features we add to the pattern list, dependencies and subject label show very close to performance of the All features model. We observe better performance on the rare label author gap (A-GAP) with just these features alone.

Most categories are negatively impacted by the removal of the subject label with the exception of author uses/build/similar to cited work (A-CW-U) and authors work differs from cited work (A-CW-D). As a single feature we see that subject is important to the classifier performance and contributes to several of the labels - background with no evidence (BG-NE), cited work description (CW-D), author description (A-D) and author and cited work differ (A-CW-D). Leaving out subject label was the only feature to cause a drop in classifier performance that was significant. In Table 6 and Table 7 experiments from using a gold subject label and using a history feature of previous label are presented. History label was previously shown by (Liakata et al., 2012) to contribute to sentence classification. Our gold subject label was determined from the annotated label. We see that determining this label accurately has an almost 15% increase in the performance of our classifier and an increase in F1 score for all label categories. Including a previous label also increases the classifier performance, but this increase was not significant.

7 Discussion and Conclusions

We use a manually annotated data-set designed for support of writer feedback of a *Related Work* section and show that we can outperform existing similar methods. We describe our feature set, proposing some novel features such as co-reference specific to *Related Works*, citation types and include these in our adapted pattern set. We show the introduction of our features over and above the original pattern features (Teufel, 1999) was a contributing factor to the performance of our classifier. This highlights the importance of understanding the author intentions of interest and looking for patterns that are specific to these. This major contribution of patterns though is also a limitation in that this is built on a study of patterns that occur within the computational linguistic domain and how it would perform in another domain remains to be investigated. Recent work of (Asadi et al., 2019) show that using WordNet roots for Nouns, e.g where nouns are taken to their more general form (e.g., *mm* and *cm* become *quantity*, is a useful feature for author intention identification. The application of WordNet is one possible

Features	BG-EV	BG-NE	BG-EP	CW-EV	CW-D	A-CW-U	A-D	TXT	A-D	A-G
ALL	.39	.72	.73	.53	.84	.48	.47	.88	.63	.25
+Plabel	.50	.60	.70	.60	.86	.51	.46	.63	.61	.27
+GoldSubject	.61	.86	.88	.67	.94	.68	.72	1	.88	.4

Table 6: Mean F-Measures for Labels All features and All with Gold Subject and Previous label

Features	Prec	Recall	F1	Acc%
ALL	.69	.7	.7	70
+Plabel	.71	.72	.71	71.72
+GoldSubject	.84	.85	.84	84.6

Table 7: Classifier Performance, Mean scores after 10 iterations

avenue that may assist in transitioning our pattern list to another domain.

In future work, we intend to investigate augmenting our pattern set further. Jurgens et al. (2018) implement a bootstrapping pattern that identifies over four times the manually curated patterns, identifying new patterns that apply in a citing sentence, the preceding or following sentence. Bootstrapping to expand seed cue phrases based on rhetorical relations (Abdalla and Teufel, 2006) has also been successful. Incorporating more information from a preceding or following sentence we believe could help classify sentences where there is no linguistic clue as to the subject e.g. those that carry on describing a cited work but their is no co-reference to signal the subject. Understanding sentence subject is important, currently it contributes to the classifier performance but we show an almost 15% increase in performance that could occur using a gold sentence subject label. Having a way to improve our current implementation of sentence subject assignment would be beneficial.

Our overall intention for this work is to support writer feedback and so we intend to investigate how well our current level of automatic recognition of author intentions can support feedback and how useful this is to novice writers.

References

Rashid M. Abdalla and Simone Teufel. 2006. A bootstrapping approach to unsupervised detection of cue phrase variants. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL-44, pages 921–928. https://doi.org/10.3115/1220175.1220291.

Sophie Kitto Kirsty Knight Simon Buckingham Shum Simon Abel. 2018. Designing personalised, automated feedback to develop students research writing skills. In *Proceedings of 2018 Australasian Society for Computers in Learning in Tertiary Education*. pages 15–24.

Claire Aitchison, Janice Catterall, Pauline Ross, and Shelley Burgin. 2012. 'Tough love and tears': learning doctoral writing in the sciences. *Higher Education Research & Development* 31(4):435–447. https://doi.org/10.1080/07294360.2011.559195.

Mandya A. Angrosh, Stephen Cranefield, and Nigel Stanger. 2010. Context identification of sentences in related work sections using a conditional random field: Towards intelligent digital libraries. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*. ACM, New York, NY, USA, JCDL '10, pages 293–302. https://doi.org/10.1145/1816123.1816168.

Mandya A. Angrosh, Stephen Cranefield, and Nigel Stanger. 2012. A citation centric annotation scheme for scientific articles. In *Proceedings of the Australasian Language Technology Association Workshop 2012*. Dunedin, New Zealand, pages 5–14. https://www.aclweb.org/anthology/U12-1003.

Laurence Anthony and George V. Lashkia. 2003. Mover: A Machine Learning Tool to Assist in the Reading and Writing of Technical Papers. *Professional Communication, IEEE Transactions on* 46:185 – 193. https://doi.org/10.1109/TPC.2003.816789.

Nasrin Asadi, Kambiz Badie, and Maryam Tayefeh Mahmoudi. 2019. Automatic zone identification in scientific papers via fusion techniques. *Scientometrics* 119(2):845–862. https://doi.org/10.1007/s11192-019-03060-9.

Awais Athar. 2014. Sentiment analysis of scientific citations. Technical Report UCAM-CL-TR-856, University of Cambridge, Computer Laboratory. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-856.pdf.

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In *LREC 2008*.

Leo Breiman. 2001. Random forests. *Machine learning* 45(1):5–32.

J Burstein, D Marcu, and K Knight. 2003. Finding the WRITE stuff: Automatic identification of discourse structure in student essays. *IEEE Intelligent Systems* .

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2004. Automated Essay Evaluation: The Criterion Online Writing Service. *AI Magazine* 25:27–36.

Arlene J Casey, Bonnie Webber, and Dorota Głowacka. 2019. A Framework for Annotating 'Related Works', to Support Feedback to Novice Writers. In *LAW '13: Proceedings of the Linguistic Annotation Workshop*. Association for Computational Linguistics, Stroudsburg, PA, USA.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 390–400. https://doi.org/10.18653/v1/D15-1045.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20(1):37–46.

Elena Cotos and Nick Pendar. 2016. Discourse classification into rhetorical functions for awe feedback. *calico journal* 33(1):92–116.

Valéria D Feltrim, Simone Teufel, Maria Graças V das Nunes, and Sandra M Aluísio. 2006. Argumentative zoning applied to critiquing novices scientific abstracts. In *Computing Attitude and Affect in Text: Theory and Applications*, Springer, pages 233–246.

Beatriz Fisas, Horacio Saggion, and Francesco Ronzano. 2015. On the discoursive structure of computer graphics research papers. In *Proceedings of the 9th linguistic annotation workshop*. Association for Computational Linguistics, Denver, Colorado, USA, pages 42–51. https://doi.org/10.3115/v1/W15-1605.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained argumentation features for scoring persuasive essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 549–554. https://doi.org/10.18653/v1/P16-2089.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: an investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*. Association for Computational Linguistics, pages 99–107.

Ken Hyland. 2008. As can be seen: Lexical bundles and disciplinary variation. *English for Specific Purposes* 27(1):4–21.

Ken Hyland. 2012. Bundles in academic discourse. *Annual Review of Applied Linguistics* 32:150–169. https://doi.org/10.1017/S0267190512000037.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the Evolution of a Scientific Field through Citation Frames. *Transactions of the Association of Computational Linguistics* 6:391–406.

Barbara Kamler and Pat Thomson. 2006. *Helping doctoral students write: Pedagogies for supervision*. Routledge. https://doi.org/10.4324/9780203969816.

Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin Batchelor, and Dietrich Rebholz-Schuhmann. 2012. Automatic recognition of conceptualization zones in scientific articles and two life science applications. *Bioinformatics* 28(7):991–1000.

Joseph A. Maxwell. 2006. Literature Reviews of, and for, Educational Research: A Commentary on Boote and Beile's "Scholars Before Researchers". *Educational Researcher* 35(9):28–31. https://doi.org/10.3102/0013189X035009028.

Yoko Mizuta and Nigel Collier. 2004. Zone identification in biology articles as a basis for information extraction. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*. Association for Computational Linguistics, pages 29–35. https://www.aclweb.org/anthology/W04-1205.

Claude Nadeau and Yoshua Bengio. 1999. Inference for the Generalization Error. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'99, pages 307–313. http://dl.acm.org/citation.cfm?id=3009657.3009701.

Charles Oppenheim and Susan P Renn. 1978. Highly cited old papers and the reasons why they continue to be cited. *Journal of the American Society for Information Science* 29(5):225–231.

Brian Paltridge and Sue Starfield. 2007. *Thesis and Dissertation Writing in a Second Language*. Routledge. https://doi.org/10.4324/9780203960813.

Ulrich Schäfer, Christian Spurk, and Jörg Steffen. 2012. A fully coreference-annotated corpus of scholarly papers from the ACL anthology. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, Mumbai, India, pages 1059–1070. https://www.aclweb.org/anthology/C12-2103.

Hagit Shatkay, Fengxia Pan, Andrey Rzhetsky, and W John Wilbur. 2008. Multi-dimensional classification of biomedical text: toward automated, practical provision of high-utility text to diverse users. *Bioinformatics* 24(18):2086–2093. https://doi.org/10.1093/bioinformatics/btn381.

Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, Baltimore, Maryland, pages 69–78. https://doi.org/10.3115/v1/W14-2110.

Marc Sumner, Eibe Frank, and Mark A Hall. 2005. Speeding up logistic model tree induction. *PKDD* LNCS 3721:675–683. https://hdl.handle.net/10289/1446.

John M Swales. 1990. *Genre Analysis: English in academic and research settings*. Cambridge University Press.

Simone Teufel. 1999. *Argumentative zoning: Information extraction from scientific text*. Ph.D. thesis, University of Edinburgh.

Simone Teufel and Minyen Kan. 2011. Robust Argumentative Zoning for Sensemaking in Scholarly Documents. In *In Advanced Language Technologies for Digital Libraries*. Springer, pages 154–170.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics* 28(4):409–445. https://doi.org/10.1162/089120102762671936.

Simone Teufel, Advaith Siddharthan, and Colin Batchelor. 2009. Towards domain-independent argumentative zoning: Evidence from chemistry and computational linguistics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 1493–1502. https://www.aclweb.org/anthology/D09-1155.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sydney, Australia, pages 103–110. https://www.aclweb.org/anthology/W06-1613.

Paul Thompson and Chris Tribble. 2001. Looking at citations: Using corpora in english for academic purposes. *Language Learning Technology* 5(3):91 – 105.

Stephen E Toulmin. 2003. *The Uses of Argument*. Cambridge University Press.

Melvin Weinstock. 1971. Citation indexes. encyclopedia of library and information science. volume 5. eds. a. kent & h. lancour.

Sparse Victory – A Large Scale Systematic Comparison of Count-Based and Prediction-Based Vectorizers for Text Classification

Rupak Chakraborty
Adobe Inc, India
rupak97.4@gmail.com

Ashima Elhence
Adobe Inc, India
elhenceashima@gmail.com

Kapil Arora
Adobe Inc, India
karora@adobe.com

Abstract

In this paper we study the performance of several text vectorization algorithms on a diverse collection of 73 publicly available datasets. Traditional sparse vectorizers like Tf-Idf and Feature Hashing have been systematically compared with the latest state of the art neural word embeddings like Word2Vec, GloVe, FastText and character embeddings like ELMo, Flair. We have carried out an extensive analysis of the performance of these vectorizers across different dimensions like classification metrics (.i.e. precision, recall, accuracy), dataset-size, and imbalanced data (in terms of the distribution of the number of class labels).

Our experiments reveal that the sparse vectorizers beat the neural word and character embedding models on 61 of the 73 datasets by an average margin of 3-5% (in terms of macro f1 score) and this performance is consistent across the different dimensions of comparison.

1 Introduction

The use of text vectorization for NLP applications has its roots in information retrieval and allied fields for measuring semantic similarity as enshrined by Jones (1972). Traditional methods for converting text into a fixed length vector include a bag of words representation (Zelling, 1954), where each word in the vocabulary is represented by a unique index, Tf-Idf builds upon this by weighting the frequency of each word by the inverse count of its document occurrence thereby mitigating the noise induced by Zipfian distribution of words in natural language. These vector space models are often referred to as sparse discrete representations owing to the large number of zeros that predominate their vector representations. Building on this foundation, research direction was aimed at generating continuous distributional semantics of text using factorization of word co-occurrence matrix as evinced in Latent Semantic Analysis (Dumais et. al, 1988, 2004). These SVD (Singular Value Decomposition) based approaches form the precursors of modern topic modeling (Blei et. al, 2003, 2002). Feature hashing often referred to as the hashing trick (analogy to the kernel trick) involves using a non-cryptographic hash function to convert text (i.e. word tokens) to a corresponding numerical representation, these representations are made to be uniformly distributed by including a secondary hashing function which alters the sign bit of the output of the first hashing function. These have been shown to have provable error bounds (Weinberger et al, 2009) and have been previously used for collaborative spam filtering and large scale multi-task learning (Attenberg et al, 2009, Weinberger et al, 2009).

The use of the word neural word embeddings was first coined by the authors (Bengio et al, 2003) in their landmark paper which showed the efficacy of using hidden layer representations for measuring semantic similarity between words. Building upon this, it was further demonstrated by (Collobert et al, 2011) that unsupervised pre-training of word vectors preserved their syntactic and semantic similarities which lead to state of the art results on many downstream tasks. But it wasn't until introduction of Word2Vec (Mikolov et. al, 2013) that neural word embeddings became mainstream, this in a sense opened the flood gates of research into these models. GloVe (Pennington et. al, 2014) uses a log-bilinear regression model that combines the advantages of the two major model families in the literature - global matrix factorization and local context window methods. Enriching word vectors with subword information has proven to be effective as can be seen in fastText(Bojanowski et.al, 2017). Recent embedding models like ELMo

Proceedings of Recent Advances in Natural Language Processing, pages 188–197,
Varna, Bulgaria, Sep 2–4 2019.

(Peters et. al, 2018) use masked language modeling and textual entailment tasks to generate context-sensitive character-level representations. In the same vein, Flair embeddings (Akbik et. al, 2018) leverage the internal states of a trained character language model to produce a novel type of word embedding which the authors allude to as contextual string embedding. Moving from individual word representations to document and phrase level representation, we observe a less spectacular retinue of research work. Notable among these are Skip-Thought (Kiros et al, 2015) and InferSent (Conneau et al, 2017). Recently proposed Universal Sentence Encoder (Cer et. al, 2018) which uses multi-task transfer learning based on the transformer architecture (Vaswani et. al, 2017) to deliver promising results on several natural language inference tasks.

In light of these prolific advances made in the field of text vectorization, it becomes necessary to evaluate the different algorithms on downstream tasks and juxtapose their performance with the traditional non-neural counterparts. Existing evaluations (Baroni et. al, 2014) have only focused on the semantic aspect of these representations while ignoring tasks like text classification. Even when comparisons are made on benchmarks similar to the GLUE benchmark (Wang et. al, 2018), they are almost always made with state of art deep neural network based classifiers, the non-neural classifiers like Random Forests, SVMs and GradientBoosting are left out. To the best of our knowledge there is no existing research which comprehensively evaluates the performance of modern text vectorizers on text classification tasks, it is this research gap which we want to bridge in the present study. The main contributions of the paper are the following – 1. We have collected, curated and standardized a set of 73 different datasets which cover all aspects of text classification in particular and language modeling in general. 2. We have extensively analyzed the performance of neural vectorizers like Word2Vec, GloVe, FastText, ELMo and Flair on these datasets across many dimensions like dataset-size, class imbalance, classification metrics and juxtaposed it with their count-based non-neural alternatives like Feature Hashing and Tf-Idf. 3. We have also reported results on the performance of traditional ML classifiers, since our main aim is to study the efficacy of vectorization algorithms we haven't included any neural network based classifiers in the present study. 4. Our benchmark contains 73 datasets in comparison to GLUE which has only 10, thereby making it more diverse and challenging. Finally, we have made our source code[++], datasets[**] (including train and test splits), result files and all other necessary information publicly available so that, researchers can reproduce our results and further the progress in the field. While not central to the study we have also carried out an interpretation analysis on the predictions of these vectorizers by using model agnostic, locally interpretable explanations (Riberio et. al, 2016), the results are not included in the paper, however interested readers are encouraged to refer to Appendix A for more details.

The paper is organized as follows – Section 1 introduces the paper and gives an overview of the prior research work. Section-2 provides details of our datasets and the models used. Section-3 elucidates the approach we have taken for our experiments. Section-4 presents the results of our experiments, including an extensive analysis. Section-5 concludes the paper and provides useful future research directions.

2 Data & Model Details

We have collected the datasets from a variety of online sites like Kaggle, Crowdflower (now known as FigureEight), DataTurks, UCI repository and others. They have been grouped into 8 categories for ease of analysis, these are – emotion, sentiment, reviews, medical, general classification, news, spam-fake-hate-ironic and other. The general classification category set includes things like gender classification, website categorization weather and disaster detection from tweets etc. The other category set includes a set of language tasks like natural language inference, duplicate question detection, objectivity-subjectivity analysis which have been recast in a classification framework to promote uniformity. Details about the metadata of each category is present in table 1. All the end tasks are different text categorizations ranging from classification of sentiments, emotions, news articles, reviews, gender, hate speech detection etc. All the datasets have been standardized in a common format, this format contains only two fields one for the text data other for the class label. Refer to Appendix B for necessary details about the data standardization process. As can be inferred from table 1, all the categories contain more than a

[++] Source Code - `https://tinyurl.com/y23j3ygd`
[**] Datasets - `http://tinyurl.com/yyofx77r`

Category	# Datasets	# Avg Tokens	# Avg Sentences
Sentiment	16	1.1 $*10^8$	1.2 $* 10^7$
Emotion	2	1.6 $* 10^7$	2.1 $* 10^6$
Reviews	7	1.2 $* 10^9$	2.8 $* 10^7$
News	8	3.3 $* 10^8$	4.2 $* 10^7$
General Classification	17	1.9 $* 10^8$	4.8 $* 10^6$
Spam-Fake-Hate-Ironic	10	8.6 $* 10^7$	2.0 $* 10^6$
Medical	6	2.9 $* 10^8$	1.8 $* 10^7$
Other	7	1.8 $* 10^8$	1.4 $* 10^7$

Table 1: Details of datasets on a category basis

million sentences on an average. Out of the 73 datasets 17 contain more than 50K data samples, 39 contain less than 10K data points and 17 contain between 10K to 50K rows. On a per category basis we observe that the general classification category contains the greatest number of datasets whose size is greater than 50K, while the sentiment category contains the maximum concentration of datasets of size less than 10K. To get an insight into the distribution of the number of rows per category refer to the box and whisker plot in figure 1, the y-axis contains the number of rows on a logarithmic scale (base 10).

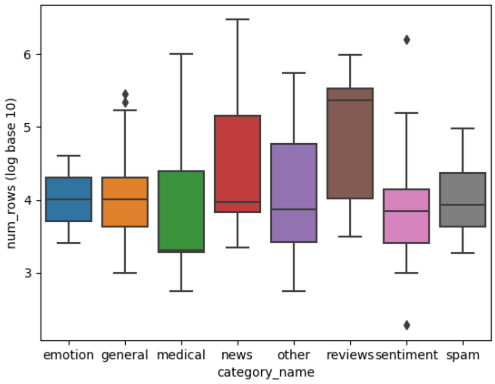


Figure 1: Box plot of distribution of rows per category

The neural embedding models have been pre-trained** in the following way - the Word2Vec model (of dimension 300) and has been trained on Google News Corpus (100 billion words). For greater ease of comparison both the GloVe and fastText models have a dimension of 300 and have been trained on the Common Crawl Corpus (640 billion words). The ELMo embedding has also been trained on Google News Corpus and as for the Flair embeddings we have used the original model provided by the authors which has been trained on English Wikipedia text. To provide a level-ground of comparison with the neural counterparts both the Tf-Idf and Feature Hashing vectorizers have a dimension of 300. For the hashing vectorizer, a variant of Murmurhash3(Appleby, 2015) has been used to project the word tokens in a lower dimensional embedding space.

3 Approach

A systematic and comprehensive comparison of the vectorizers entails evaluating them across several dimensions, reporting the results using relevant metrics and then interpreting the results. The dimensions considered in the present study are the following – 1. Dataset-size, we consider 3 mutually exclusive and exhaustive ranges: less than 10K, greater than 50K and between 10K and 50K, these ranges have been chosen because they provide the most coverage across the selected datasets. For each of these ranges we analyze the performance on a per category basis. 2. Imbalance measure as reflected in the distribution of number of class labels (using equation 1).

$$Imbalance = \sum_{\forall\, n1,n2} \frac{|n1-n2|}{n1+n2} \qquad (1)$$

In equation 1, n1 and n2 are the number of samples belonging to classes C_i and C_j. For a given dataset containing N different classes we find the ratio of the absolute difference of the number of data-points in the two classes to the total number of data points in the two classes, we calculate this for all pairs of classes. For a perfectly balanced dataset this value will be zero, higher the value more will be the imbalance measure, there is no upper bound on the value. For the imbalance measure we calculate the terciles and divide the datasets into three parts based on the interval of these tercile values they are – [0, 1.03), [1.03, 4.46), [4.46, ∞], where ∞ denotes the max value across all the datasets.

As mentioned in section 1, we have only considered non-neural classifiers as our main aim is to study the performance of vectorizers. The classifiers included in the present study are Random Forests, GradientBoost, AdaBoost, SVM (Linear Kernel) and Logistic Regression. These have been included because they represent a healthy mix of both bagging and boosting approaches along with linear models. For each dataset we measure its performance across all combinations of vectorizers and classifiers.

** Pre-trained Models Download Link- https://tinyurl.com/y2mlnhdf

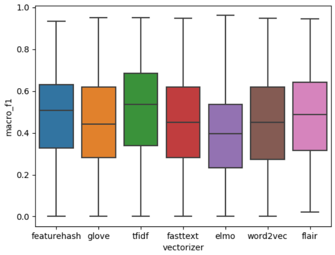


Figure 2: Vectorizer f1-score (global)

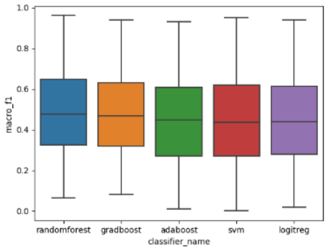


Figure 3: Classifier f1-score (global)

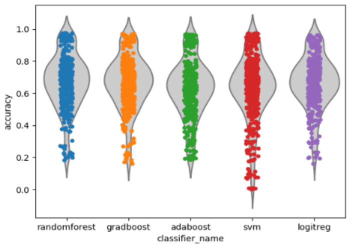


Figure 4: Classifier accuracy (global)

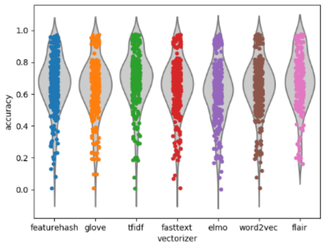


Figure 5: Vectorizer accuracy (global)

4 Experiments & Results

All the experiments have been performed on a 32 GB Intel i7 processor, with a clock rate of 3.40 GHz. Since all the embedding models have been pre-trained, using a dedicated GPU doesn't result in a significant speedup, as we have noticed an increase of only 1.5x - 2x while using a Nvidia GTX Geforce 1080 Ti, 11 GB graphics processor. The total training time for all combinations of datasets, vectorizers and classifiers is more than 3 weeks.

We have carried out basic pre-processing of the text data like – case normalization, stopword removal, punctuation and special character removal followed by word tokenization, though it will be interesting to see the effects of more sophisticated pre-processing like lemmatization on the results. The hyperparameter settings of all the classifiers have been set to default values as used in the scikit-learn library except for number of trees in the Random Forest model which has been set to 51. Figures 2 and 3 provide a global view of the macro f1 score of the vectorizers and classifiers averaged across all the datasets. For a given vectorizer we have calculated the mean performance metric (precision, recall, accuracy) across all classifiers and datasets. As can be inferred from figure 2, tf-idf and feature hashing consistently outperform their heavy weight neural counterparts, among the neural vectorizers flair embeddings demonstrate competitive performance on almost all datasets. The violin plots shown in figures 4 and 5 elucidate the performance of the classifiers and vectorizers (based on accuracy) under the same conditions as figures 2 and 3. We can observe the same trends in these figures as we have previously seen in figures 2, 3. With respect to classifiers, Random Forests, Gradient Boost and Logistic Regression are always among the top performing trio. Apart from this, we have also seen that our results[*] conform to widely established trends like the negative correlation between the increase in number of classes and classifier performance metrics, we will expand upon this more in the section on analyzing performance metrics based on class imbalance.

4.1 For Size Less Than 10K

Tables 2 and 5, illustrate the performance of vectorizers and classifiers for all datasets whose size is less than 10K. The results have been grouped on a per category basis, in the category column the number inside the brackets denotes the number of datasets which fall into that category for the given dataset size range. The mean values of Precision Recall and Accuracy have been juxtaposed by following the notation Pr./Rec./Acc. We notice a wide variance in the performance metrics across the categories especially for reviews and emotion. The reason for this is that the emotion category has a dataset which has 18 classes while only containing 2524 samples, same is the case for reviews which has a dataset containing 41 classes. It is this small sample size and sparse data problem which reflects in the suboptimal performance of the vectorizers and classifiers. The number of class labels for all the other datasets in this size range lies between 2-5. Again, we observe that, tf-idf and feature hash come out on top consistently beating the neural counterparts (except for Flair) by a margin of 10% (in terms of accuracy). On the classifier front again Random Forests, Gradient Boost and Logistic Regression edge out SVMs and AdaBoost. In context of the vectorizers we would like to make a case for feature hashing, extolling its many virtues which include – low computational footprint, the absence of a fixed vocabulary, theoretical error bounds and competitive performance, which serve to make it an ideal candidate for establishing strong baselines.

[*] All result files can be accessed from - https://tinyurl.com/y5e4hftt

Category Name	**GloVe** **(Pr./Rec./Acc.)**	**FastText** **(Pr./Rec./Acc.)**	**Word2Vec** **(Pr./Rec./Acc.)**	**ELMo** **(Pr./Rec./Acc.)**	**Tf-Idf** **(Pr./Rec./Acc.)**	**FeatureHash** **(Pr./Rec./Acc.)**	**Flair** **(Pr./Rec./Acc.)**
Sentiment (10)	41.6/38.1/59.5	42.9/38.9/59.9	42.9/38.2/59.4	36.1/35.1/57.1	**47.0/42.2/63.3**	45.0/41.3/61.8	43.3/38.9/60.0
Emotion (1)	**14.3/10.3/21.2**	12.5/9.1/20.4	11.7/9.6/20.8	7.9/7.0/19.0	**14.2/10.2/19.1**	**15.0/10.6/18.3**	8.6/8.2/18.6
General Classification (8)	56.8/49.5/64.8	55.9/49.2/64.6	54.3/48.6/64.0	46.8/44.9/61.5	**60.7/55.3/68.3**	58.2/51.8/65.1	56.5/52.2/65.0
Other (5)	59.7/56.8/67.8	59.7/56.4/67.4	59.1/56.6/67.6	52.9/52.1/65.5	**61.5/55.6/69.8**	**57.1/53.3/68.6**	59.1/52.8/67.0
Reviews (2)	52.1/37.6/83.4	44.2/37.5/83.2	52.1/37.6/83.2	45.6/37.7/83.1	**57.4/43.9/85.4**	50.0/43.6/84.1	55.8/42.2/84.0
Spam-Fake-Ironic-Hate (5)	75.9/71.0/82.6	78.0/72.4/83.7	77.8/72.4/83.6	70.7/64.8/81.0	**84.3/79.3/87.6**	80.0/74.9/84.5	79.9/76.3/85.4
Medical (4)	45.2/40.2/70.3	42.9/40.3/70.1	45.6/40.8/70.3	40.6/36.9/68.7	**53.8/45.9/73.8**	47.3/42.2/70.6	49.3/42.2/71.3
News (4)	50.6/49.4/66.6	48.6/48.3/66.2	48.9/48.7/66.1	35.9/36.6/54.3	**63.0/60.0/77.6**	58.1/55.8/73.2	**63.2/60.9/78.4**

Table 2. Mean Performance Metrics on a category basis for all Vectorizers (dataset size less than 10K)

Category Name	**GloVe** **(Pr./Rec./Acc.)**	**FastText** **(Pr./Rec./Acc.)**	**Word2Vec** **(Pr./Rec./Acc.)**	**ELMo** **(Pr./Rec./Acc.)**	**Tf-Idf** **(Pr./Rec./Acc.)**	**FeatureHash** **(Pr./Rec./Acc.)**	**Flair** **(Pr./Rec./Acc.)**
Sentiment (4)	54.5/45.5/60.8	55.8/46.7/61.7	55.5/46.4/61.5	52.9/42.2/59.2	**64.0/57.0/68.6**	**60.1/52.7/65.2**	57.6/49.9/63.0
Emotion (1)	14.9/11.9/27.5	13.9/12.4/28.3	14.2/12.3/28.0	13.7/10.7/25.7	**23.1/16.0/31.5**	15.8/13.8/28.4	14.8/12.6/28.6
General Classification (6)	47.4/41.9/58.9	48.5/43.1/59.9	48.8/42.9/59.6	41.8/37.6/54.7	**60.4/56.7/68.5**	**57.4/52.0/65.1**	52.3/46.1/63.1
Reviews (1)	35.9/24.3/56.6	33.9/24.3/56.6	34.7/24.4/56.6	30.9/23.1/54.9	**44.1/33.2/60.9**	**43.4/29.6/58.7**	36.1/25.2/55.4
Spam-Fake-Ironic-Hate(4)	61.4/51.8/76.7	63.4/53.2/77.6	63.5/53.0/77.4	58.5/47.4/74.8	61.5/54.5/79.0	60.7/51.4/76.9	**67.0/54.2/78.3**
News (1)	15.9/9.2/50.5	15.7/9.0/49.8	16.2/9.5/51.7	14.8/9.0/46.6	37.1/29.3/75.4	**44.9/36.7/74.6**	23.3/16.7/59.2

Table 3. Mean Performance Metrics on a category basis for all Vectorizers (dataset size between 10K and 50K)

Category Name	**GloVe** **(Pr./Rec./Acc.)**	**FastText** **(Pr./Rec./Acc.)**	**Word2Vec** **(Pr./Rec./Acc.)**	**ELMo** **(Pr./Rec./Acc.)**	**Tf-Idf** **(Pr./Rec./Acc.)**	**FeatureHash** **(Pr./Rec./Acc.)**	**Flair** **(Pr./Rec./Acc.)**
Sentiment (2)	58.5/52.0/61.7	57.3/50.7/62.2	55.6/50.5/61.9	54.0/46.1/56.6	56.0/48.6/59.3	54.6/49.2/59.3	**64.1/55.2/62.0**
General Classification (3)	34.5/29.3/45.4	**34.9/30.6/45.6**	34.1/29.0/44.3	29.0/26.7/42.6	**34.6/31.7/46.0**	**35.1/29.7/44.9**	34.4/29.8/44.5
Other (2)	53.7/48.2/59.6	**55.2/49.2/60.5**	**54.9/49.2/60.5**	48.3/46.7/57.8	48.3/44.4/54.0	49.7/46.9/55.9	54.3/47.2/57.6
Reviews (4)	33.7/22.0/44.0	37.0/25.0/48.1	34.8/24.2/45.8	30.6/21.8/46.1	**38.0/28.4/54.0**	**38.5/28.4/54.3**	37.2/27.4/52.2
Spam-Fake-Ironic-Hate(1)	89.2/63.4/92.5	**90.5/65.9/93.0**	**90.8/64.8/92.5**	76.1/55.2/91.3	82.1/62.6/92.3	80.9/58.4/90.3	83.0/63.5/91.7
Medical (2)	64.4/61.7/68.5	64.5/62.0/69.7	62.0/59.9/69.5	60.9/56.8/65.0	**67.3/65.5/70.1**	64.7/62.9/70.2	**65.4/63.6/68.5**
News (3)	40.0/35.3/42.8	42.2/38.1/44.4	42.1/38.2/45.3	36.8/29.3/34.0	**42.4/40.7/47.7**	**42.0/39.4/46.1**	**41.7/37.7/47.6**

Table 4. Mean Performance Metrics on a category basis for all Vectorizers (dataset size greater than 50K)

Category Name	**RandomForest (Pr./Rec./Acc.)**	**GradientBoost (Pr./Rec./Acc.)**	**AdaBoost (Pr./Rec./Acc)**	**Logit Regression (Pr./Rec./Acc.)**	**SVM (Linear) (Pr./Rec./Acc.)**
Sentiment (10)	**46.5/40.2/60.8**	44.7/39.3/60.6	38.4/37.4/57.7	41.1/38.4/60.9	42.7/39.4/60.7
Emotion (1)	**15.9/11.2/20.5**	13.9/9.9/18.5	5.4/6.2/19.2	11.2/8.0/20.5	14.0/11.3/19.5
General Classification (8)	**58.7/51.7/66.0**	**58.2/52.0/66.2**	50.2/45.9/59.3	**53.8/50.3/66.4**	57.1/51.2/65.8
Other (5)	**61.6/57.1/68.6**	60.1/55.6/68.0	58.0/54.8/66.2	54.0/51.8/68.2	58.2/55.0/67.6
Reviews (2)	**69.8/51.9/87.1**	64.1/47.6/85.8	38.8/33.5/81.1	36.8/30.9/82.3	45.7/36.2/82.7
Spam-Fake-Ironic-Hate (5)	**80.6/73.5/84.8**	**80.8/73.9/85.0**	76.1/73.4/82.7	74.0/70.7/83.9	78.2/73.6/84.0
Medical (4)	**49.1/42.3/71.9**	46.5/41.7/71.6	42.9/39.8/67.7	46.1/40.1/71.6	**47.4/42.0/71.0**
News (4)	53.6/51.6/69.5	**56.5/53.1/70.7**	47.9/46.8/63.4	51.8/52.4/70.4	53.3/52.7/70.5

Table 5. Mean Performance Metrics on a category basis for all Classifiers (dataset size less than 10K)

Category Name	**RandomForest (Pr./Rec./Acc.)**	**GradientBoost (Pr./Rec./Acc.)**	**AdaBoost (Pr./Rec./Acc)**	**Logit Regression (Pr./Rec./Acc.)**	**SVM (Linear) (Pr./Rec./Acc.)**
Sentiment (4)	**58.7/48.3/62.8**	**59.1/47.5/63.0**	52.6/46.8/60.5	50.0/46.2/62.1	**57.6/48.5/63.1**
Emotion (1)	15.5/12.7/27.3	16.9/13.6/29.1	12.9/10.6/24.9	**18.1/13.5/29.8**	**16.4/13.9/30.2**
General Classification (6)	**54.9/46.3/61.9**	51.6/46.8/62.1	43.8/41.7/56.6	49.9/46.2/62.1	**53.0/47.5/63.0**
Reviews (1)	**39.5/25.8/56.9**	36.9/25.7/57.2	35.2/26.8/56.6	35.4/26.7/58.0	**38.8/27.4/58.1**
Spam-Fake-Ironic-Hate (4)	**76.9/60.1/82.9**	65.1/52.2/76.6	51.2/48.4/73.9	55.7/48.6/75.8	58.6/50.2/76.2
News (1)	**41.5/22.8/64.9**	27.5/21.6/59.8	1.7/2.3/41.5	21.6/17.2/60.8	28.1/21.7/63.6

Table 6. Mean Performance Metrics on a category basis for all Classifiers (dataset size between 10K and 50K)

Category Name	**RandomForest (Pr./Rec./Acc.)**	**GradientBoost (Pr./Rec./Acc.)**	**AdaBoost (Pr./Rec./Acc)**	**Logit Regression (Pr./Rec./Acc.)**	**SVM (Linear) (Pr./Rec./Acc.)**
Sentiment (2)	**57.1/52.3/61.4**	58.0/46.8/60.1	55.9/44.1/61.2	**62.4/62.5/62.4**	53.3/47.0/58.8
General Classification (3)	**41.5/34.2/51.3**	39.5/32.8/51.9	38.5/31.4/45.9	37.0/33.5/50.9	23.2/22.2/32.5
Other (2)	**55.0/52.0/61.3**	**53.9/47.4/58.8**	48.8/45.2/54.7	46.2/43.7/53.6	48.4/44.4/55.6
Reviews (4)	**47.7/30.4/53.0**	**45.1/27.7/61.4**	28.6/22.4/44.1	36.0/27.1/53.2	25.7/20.2/43.3
Spam-Fake-Ironic-Hate (1)	**89.7/62.1/92.3**	**88.8/63.7/92.5**	84.0/64.5/92.0	75.6/56.0/90.7	79.1/62.4/91.4
Medical (2)	**69.3/67.7/73.8**	**63.1/61.0/70.0**	33.7/36.3/46.7	59.7/57.0/65.4	63.1/59.5/65.5
News (3)	48.8/43.1/50.8	47.8/46.1/52.5	46.2/45.0/51.4	**53.3/51.6/55.8**	33.0/27.9//34.6

Table 7. Mean Performance Metrics on a category basis for all Classifiers (dataset size greater than 50K)

Category	**GloVe**	**FastText**	**Word2Vec**	**ELMo**	**Tf-Idf**	**FeatureHash**	**Flair**
Name	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**
Sentiment (4)	69.0/69.0/69.0	69.7/69.7/69.7	69.4/69.3/69.3	62.1/61.7/61.6	**74.9/74.1/74.1**	**71.7/71.4/71.4**	69.0/68.2/68.2
General Classification (7)	59.4/57.1/65.0	60.5/58.5/66.1	59.6/57.9/65.6	51.4/52.0/60.8	**63.4/60.9/67.3**	59.2/56.5/63.7	**61.2/61.2/67.5**
Other (5)	65.1/61.0/71.7	65.5/61.1/71.7	**64.8/61.2/71.8**	57.6/56.4/69.8	**65.2/58.4/71.8**	62.2/57.3/71.5	**64.1/57.9/72.6**
Reviews (1)	80.3/54.8/91.9	64.9/54.2/91.4	79.2/54.3/91.6	69.0/54.5/91.7	**82.6/58.1/92.0**	73.8/63.1/91.5	**84.4/59.5/92.1**
Spam-Fake-Ironic-Hate (8)	75.3/69.1/80.2	76.7/70.7/80.9	76.5/70.4/80.8	69.6/63.2/77.7	**80.8/75.0/83.9**	76.2/70.8/80.9	77.2/72.8/81.7
Medical (2)	72.2/65.9/84.6	69.1/65.3/87.0	72.5/65.6/86.9	63.4/62.0/85.9	**69.4/64.4/83.3**	**71.1/64.0/85.4**	**64.5/58.9/93.7**
News (3)	64.4/64.1/64.5	62.9/62.6/63.2	63.5/63.3/63.7	43.3/41.6/42.4	**80.9/80.1/80.3**	71.7/71.5/71.6	**83.8/83.6/83.8**

Table 8. Mean Performance Metrics on a category basis for all Vectorizers (for imbalance measure between 0.0 and 1.03)

Category	**GloVe**	**FastText**	**Word2Vec**	**ELMo**	**Tf-Idf**	**FeatureHash**	**Flair**
Name	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**
Sentiment (8)	44.8/35.9/58.5	46.9/36.9/59.2	47.2/36.7/58.9	41.5/33.7/58.1	**49.7/42.0/62.0**	48.5/40.3/60.1	49.4/39.5/60.4
General Classification (6)	51.1/42.9/60.9	49.6/42.5/60.6	48.0/41.8/60.1	42.4/38.9/57.7	**53.9/48.7/65.6**	52.2/45.8/62.2	**52.8/47.3/63.2**
Other (2)	**40.0/38.0/50.1**	39.5/37.7/49.9	**39.6/38.1/50.4**	36.0/35.6/47.6	**39.9/38.2/50.4**	37.0/37.1/49.4	**44.3/37.5/50.2**
Reviews (2)	36.0/24.2/37.5	36.2/25.9/35.9	37.9/25.7/34.2	29.4/22.2/36.2	35.9/31.6/41.2	**37.9/31.7/42.0**	36.3/30.2/39.6
Spam-Fake-Ironic-Hate (1)	62.0/42.6/79.2	61.9/44.6/79.8	63.2/44.1/79.9	63.4/38.5/77.7	**70.5/63.8/88.0**	67.2/55.0/84.7	67.9/43.4/79.5
Medical (1)	54.2/54.0/57.8	55.8/55.8/55.7	56.4/56.3/56.4	52.7/48.0/52.6	**67.7/64.8/65.8**	61.7/61.9/62.0	59.5/59.5/59.6
News (2)	38.9/37.0/70.8	36.6/36.4/71.4	36.8/36.5/70.7	32.7/34.3/68.7	**43.9/39.1/73.9**	**42.3/37.8/72.6**	**43.1/38.7/73.5**

Table 9. Mean Performance Metrics on a category basis for all Vectorizers (for imbalance measure between 1.03 and 4.46)

Category	**GloVe**	**FastText**	**Word2Vec**	**ELMo**	**Tf-Idf**	**FeatureHash**	**Flair**
Name	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**	**(Pr./Rec./Acc.)**
Sentiment (4)	26.5/24.1/54.1	26.1/25.2/54.3	25.0/23.4/53.8	22.6/23.1/52.6	**33.8/27.9/59.2**	29.6/27.3/58.1	23.8/22.6/54.0
Emotion (2)	14.6/11.1/24.4	13.2/10.7/24.3	13.1/11.0/24.4	10.8/8.9/22.3	**18.7/13.1/25.3**	**15.4/12.3/23.3**	12.1/10.5/22.9
General Classification (3)	32.4/19.4/52.1	32.8/19.7/51.8	35.9/20.6/53.1	27.8/17.8/50.8	**55.7/47.5/67.7**	52.6/40.4/63.5	40.7/23.4/53.8
Reviews (4)	30.0/19.3/56.2	29.3/21.0/60.3	27.6/20.8/60.1	24.5/20.3/59.6	**36.4/27.8/65.8**	**33.7/24.7/63.3**	31.7/25.0/61.4
Spam-Fake-Ironic-Hate (1)	47.1/31.7/89.3	**53.0/32.5/90.2**	**52.6/33.4/90.1**	41.0/28.0/90.6	32.5/16.5/85.6	41.0/19.8/87.0	**58.8/34.9/91.1**
Medical (3)	35.6/31.0/64.5	34.2/31.3/65.4	34.6/30.5/65.4	33.4/28.4/62.7	**46.5/38.4/68.5**	36.8/33.1/63.9	36.6/25.3/66.0
News (3)	15.9/8.4/35.8	15.8/8.3/36.3	15.7/8.4/38.2	14.6/7.0/34.6	25.4/20.1/49.5	**30.8/24.2/49.8**	**25.0/17.0/54.8**

Table 10. Mean Performance Metrics on a category basis for all Vectorizers (for imbalance measure between 4.46 and ∞)

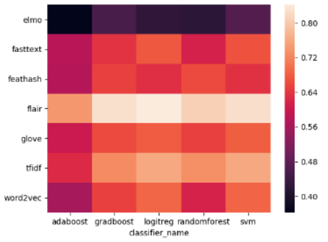


Figure 6: Heatmap(accuracy) for a news classification dataset

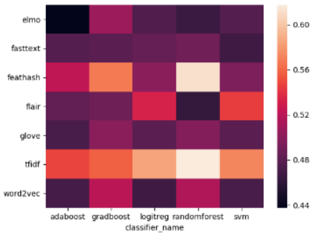


Figure 7: Heatmap(accuracy) for a inference dataset

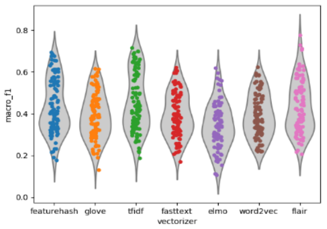


Figure 8: Performance (f1-score) for imbalance range [1.03, 4.46)

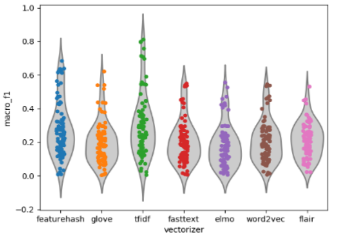


Figure 9: Performance (f1-score) for imbalance range [4.46, ∞)

4.2 For Size Between 10K – 50K

Tables 3 and 6 provide an overview of the performance metrics of the vectorizers and classifiers under the same conditions as tables 2 and 5 the only difference is that the range of datasets is between 10K and 50K, since Medical and Other categories have no datasets in this range, they haven't been included in the tables. The trends observed here are consistent with the ones we have observed in section 4.1. The average number of class labels in this range is less than 10 for all categories except for news, which has a dataset containing 75 classes and emotion where the number of class labels is 13 for a given dataset.

4.3 For Size Greater Than 50K

Tables 4 and 7 illustrate the performance metrics for 17 datasets whose size is greater than 50K, the emotion category is missing because it has no datasets in this range. The presence of a dataset with 27995 class labels in the general classification category skews the results and leads to the observed performance metrics, same is the case for news which has a dataset containing 756 classes. For all the experiments we have used a train-test split of 80-20, the seed used for random split is kept same so the results will be consistent while reproducing. Again, the trends noticed here are faithful to the observed trends in section 4.1 and 4.2. Here we would briefly like to mention that our intention is not to undermine the spectacular advances of deep learning and state of the art results it has produced in NLP, we are aware of the fact that only deep models are capable of scaling in performance with the increase in data-size, however it might seem like an overkill in situations where simpler models do an equally good job. Refer to figure 6 (dataset used is of a news classification task), for a case where a neural embedding (Flair) beats every other non-neural and neural counterpart (in terms of accuracy). Figure 7 also illustrates an accuracy heatmap where the results are more in tune with the general trends of the study, the dataset used here is one of agreement-disagreement between sentence pairs.

4.4 Metrics Under Class Imbalance

Tables 8, 9 and10 demonstrate the performance of the vectorizers across the three selected strata of imbalance measure. Tf-Idf and Feature Hash shines in all three cases. As a general trend we have noticed that the accuracy of the vectorizers increase when the text is more verbose (i.e. news) in comparison to limited character content. (i.e. tweets). Flair embeddings show a competitive performance (when the imbalance measure is less than 4.46), sometimes even outperforming the sparse vectorizers, as per the results aggregated it is clearly our neural vectorizer of choice. The violin plots in figures 8 and 9, illustrate the performance of the vectorizers (macro f1 score averaged across all datasets and classifiers) under different ranges of class imbalance (as calculated using equation 1). In all these cases feature hashing has the highest median performance as can inferred from the greatest density of points in the center. The tf-idf vectorizer has the highest variance in its performance because of a skew in class distribution, which in turn skews the performance of this count-based vectorizer. Word2Vec, FastText and GloVe have almost similar performance.

5 Conclusion & Future Work

"*Neural Embedding models are not a silver bullet*", it is in this spirit that we have carried out the present study and reported the results. The analysis presented here might serve as a starting point for researchers new to this field, who might be overwhelmed by the plethora of alternate text vectorizers available. It will also serve as a strong baseline for future research direction in the domain of text classification. In the future we would like to use neural classifiers (CNNs and RNNs) in place of traditional ones presently used, so that we can provide a complete picture of classifier performance across the entire spectrum. Fine-tuning models like ELMo, ULMFit (Howard and Ruder, 2018) and observing the change in results will be an interesting line of future work. Additionally, we would also like to explore other embedding models like BERT (Devlin et. al, 2018) and CoVe (McCann et.al, 2017).

Detailed Interpretation Visualizations - https://tinyurl.com/yxgf2vuj

References

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28(1):11–2.

Zellig S Harris. "Distributional structure. Word, 10 (2-3): 146–162. Reprinted in Fodor, J. A and Katz, JJ." *Readings in the Philosophy of Language* (1954): 33-49.

Susan T Dumais. "Latent semantic analysis." *Annual review of information science and technology* 38, no. 1 (2004): 188-230.

Susan T Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. "Using latent semantic analysis to improve access to textual information." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 281-285. Acm, 1988.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3, no. Jan (2003): 993-1022.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." In *Advances in neural information processing systems*, pp. 601-608. 2002.

Josh Attenberg, Kilian Weinberger, Anirban Dasgupta, Alex Smola, and Martin Zinkevich. "Collaborative email-spam filtering with the hashing trick." In *Proceedings of the Sixth Conference on Email and Anti-Spam*. 2009.

Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola. "Feature hashing for large scale multitask learning." *arXiv preprint arXiv:0902.2206* (2009).

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. "Natural language processing (almost) from scratch." *Journal of machine learning research* 12, no. Aug (2011): 2493-2537.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A neural probabilistic language model." *Journal of machine learning research* 3, no. Feb (2003): 1137-1155.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.

Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP),* pp. 1532-1543. 2014.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." *Transactions of the Association for Computational Linguistics5* (2017): 135-146.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Skip-thought vectors." *In Advances in neural information processing systems*, pp. 3294-3302. 2015.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. "Supervised learning of universal sentence representations from natural language inference data." *arXiv preprint* arXiv:1705.02364 (2017).

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant et al. "Universal sentence encoder." *arXiv preprint* arXiv:1803.11175 (2018).

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." *arXiv preprint arXiv:1802.05365* (2018).

Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual string embeddings for sequence labeling." In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638-1649. 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *In Advances in Neural Information Processing Systems*, pp. 5998-6008. 2017.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. "Dont count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors." *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), vol. 1, pp. 238-247. 2014.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. "Glue: A multi-task benchmark and analysis platform for natural language understanding." *arXiv preprint* arXiv:1804.07461 (2018).

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier." *In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135-1144. ACM, 2016.

Austin Appleby. *Murmurhash3 64-bit finalizer*. Version 19/02/15. https://code. google. com/p/smhasher/wiki/MurmurHash3.

Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 328-339. 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXivpreprint arXiv:1810.04805* (2018).

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. "Learned in translation: Contextualized word vectors." *In Advances in Neural Information Processing Systems*, pp. 6294-6305. 2017.

Ajay Patel, Alexander Sands, Chris Callison-Burch, and Marianna Apidianaki. "Magnitude: A Fast, Efficient Universal Vector Embedding Utility Package." In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 120-126. 2018.

A Fine-Grained Annotated Multi-Dialectal Arabic Corpus

Anis Charfi
Information Systems
Carnegie Mellon University in Qatar
acharfi@andrew.cmu.edu

Wajdi Zaghouani
College of Humanities and Social Sciences
Hamad Bin Khalifa University Qatar
wzaghouani@hbku.edu.qa

Syed Hassan Mehdi
Information Systems
Carnegie Mellon University in Qatar
smehdi@andrew.cmu.edu

Esraa Mohamed
Information Systems
Carnegie Mellon University in Qatar
emohamad@andrew.cmu.edu

Abstract

We present ARAP-Tweet 2.0, a corpus of 5 million dialectal Arabic tweets and 50 million words of about 3000 Twitter users from 17 Arab countries. Compared to the first version, the new corpus has significant improvements in terms of the data volume and the annotation quality. It is fully balanced with respect to dialect, gender, and three age groups: under 25 years, between 25 and 34, and 35 years and above. This paper describes the process of creating the corpus starting from gathering the dialectal phrases to find the users, to annotating their accounts and retrieving their tweets. We also report on the evaluation of the annotation quality using the inter-annotator agreement measures which were applied to the whole corpus and not just a subset. The obtained results were substantial with average Cohens Kappa values of 0.99, 0.92, and 0.88 for the annotation of gender, dialect, and age respectively. We also discuss some challenges encountered when developing this corpus.

1 Introduction

As the popularity of natural language processing (NLP) based tools increases, there is a rising need for language resources such as annotated corpora to develop NLP tools. In this regard, Arabic lags behind other languages due to several reasons. First, Arabic is complex at various levels of linguistic representation (phonology, orthography, morphology, and syntax). Even though native Arabic speakers prefer using Modern Standard Arabic (MSA) for official communications, we are witnessing with the popularity of social media a rise in the use of dialectal Arabic for informal online interactions such as those found in blogs, forums, chats, tweets, posts, etc. (Mubarak and Darwish, 2014; Bouamor et al., 2018).

The content written by the users in social media sites can reveal some characteristics and attributes about them, which is the main focus of author profiling research. However, the lack of Arabic language resources limits that research on author profiling for the Arabic language in particular dialectal Arabic.

We present in this paper ARAP-Tweet 2.0, which is a large-scale manually-annotated multi-dialectal Arabic corpus. A first version of this corpus was presented in (Zaghouani and Charfi, 2018a,b) and it was extended significantly in terms of data volume, number of users, and annotation quality. ARAP-Tweet 2.0 covers dialects from 17 Arab countries and 15 regions. The number of users per region is 198, including a total of about 3000 users and approximately 5 million tweets. All users' accounts were manually annotated with respect to the dialect, gender, and age. Thereby, we distinguished three age groups: under 25 years, between 25 years and 34 years, and 35 years and above. Moreover, significant effort was put in checking and improving the annotation quality.

The rest of this paper is organized as follows. Section 2 presents related work and Section 3 describes the methodology used in creating and annotating our corpus. Section 4 reports on the verification of the annotation as well as the evaluation. Section 5 discusses some challenges that we encountered when developing this corpus. Finally, Section 6 concludes the paper and outlines possible directions for future work.

Proceedings of Recent Advances in Natural Language Processing, pages 198–204,
Varna, Bulgaria, Sep 2–4 2019.

2 Related Work

Most research on Arabic NLP has focused on Modern Standard Arabic (MSA) (Habash, 2010). There are many parallel and monolingual annotated data collections with syntactic and semantic information such as the different iterations of Penn Arabic Probanks (Palmer et al., 2008; Zaghouani et al., 2010, 2012) and treebanks (Maamouri et al., 2010). Based on such resources, various tools were developed for syntactic parsing and morphological analysis (Habash, 2010).

Even though there are relatively many resources for MSA, Dialectal Arabic (DA) lags behind in terms of available resources. There have been some limited efforts toward creating resources for the most popular dialects such as the Egyptian and Levantine dialects which were presented in (Habash et al., 2013; Diab and Habash, 2007; Pasha et al., 2014). For example, Habash et al. created resources for morphological analysis of Egyptian dialect (Habash et al., 2013). For their work on machine translation, Zbib et al. (2012) created Levantine-English and Egyptian-English parallel corpora using crowd sourcing. Khalifa et al. (2018) created a morphologically annotated data corpus of Emirati Dialect.

Khalifa et al. (2016) created a corpus of 100M words covering various Arabic dialects. Other related projects worth to be mentioned are: the Egyptian Arabic Treebank (Maamouri et al., 2014); the Levantine Arabic Treebank (Maamouri et al., 2006), The Curras Palestinian Arabic annotated corpus with more than 70,000 words of various genres (Jarrar et al., 2014).

Furthermore, AlShargi et al. (2016) created a Yemeni (Sanaa Dialect) dataset and also a Moroccan Arabic corpus, while Al-Twairesh et al. (2018) built SUAR, a Najdi Arabic corpus annotated with the morphological analyzer MADAMIRA (Pasha et al., 2014). Finally, Voss et al. (2014) presented a Moroccan Arabic corpus annotated for code-switching (French, Berber and Morrocan Arabic).

Moreover, there have been some efforts towards creating Dialectal Arabic corpora by either translating existing corpora to dialects or by crowd sourcing annotation for data collected through various sources such as microblogs (e.g., Twitter). Along these lines, DART was developed as a Twitter based data set of dialectal Arabic covering five Arabic dialects: Egyptian, Levantine, Gulf, and Iraqi (Alsarsour et al., 2018). The annotation of this corpus was done through crowd sourcing. Bouamor et al. (2014) presented a multi-dialectal parallel corpus with 2,000 sentences translated to MSA, Tunisian, Jordanian, Palestinian and Syrian Arabic. Later on, MADAR was developed as a Multi-dialectal large scale corpus that provides parallel translation for 25 Arabic city dialects (Bouamor et al., 2018). All these efforts on creating Dialectal Arabic corpora either targeted some specific dialects only or did not provide the necessary annotation for author profiling such as annotation about age and gender.

3 Methodology

In the following, we report on the methodology and process followed for the creation, annotation, and validation of our corpus.

3.1 Corpus Overview

Twitter is an increasingly popular social media platform among the Arabic speaking people who tend to frequently use their Arabic dialects when sharing their stories and opinions. Therefore, we focused on Twitter to collect the data for our corpus. ARAP-Tweet 2.0 was developed in the context of the ARAP research project[1] and it includes about 5.3 million Arabic tweets of approximately 3000 users from the following 15 Arabic speaking regions: Qatar, Kuwait, United Arab Emirate (UAE), Kingdom of Saudi Arabia (KSA), Oman, Yemen, Iraq, Tunisia, Algeria, Libya, Morocco, Lebanon-Syria, Palestine-Jordan, Egypt, and Sudan. To the best of our knowledge, there is no other corpus that covers as many Arab regions as ours. For every region, we collected the tweets of 198 users that were equally-balanced over gender and three age groups: under 25, 25 until 34, and 35 and up. Compared to the first version of our corpus (Zaghouani and Charfi, 2018a) the number of users per region doubled in this second version.

3.2 Users

We used Twitter as our source for finding the users. In the beginning, we got a number of accounts by the geographical location of tweets and by searching on Twitter using specific words. We collected the users who posted tweets in a specific geographical location as defined in the tweet's object. As this information was not available for all

[1] arap.qatar.cmu.edu

tweets, we searched for users who had tweets that include some unique words, which are specific to a certain Arab region. Table 1 shows some of these unique words from every region covered by our corpus along with a tweet example for each word.

After that, we found further accounts using the followers of the initially identified users. Once the users were found, we hired experienced annotators to manually annotate their age, gender, and dialect. The annotators followed well-defined annotation guidelines, which are based on an extended version of the guidelines published in (Zaghouani and Charfi, 2018b). We continuously monitored the annotators' work and updated the guidelines when needed. For each region, we retrieved the tweets of the users from their Twitter profiles. The selected users had to have at least 100 Arabic tweets and at most 3200 tweets. Moreover, the tweets had to be the original tweets, which means that we did not include retweets in our corpus. The average number of words per tweet is 10.

3.3 Annotation

In the following, we report on the annotation of the users with respect to gender, age and dialect.

3.3.1 Gender Annotation

The gender was manually annotated for users and the number of users is balanced with respect to the gender across all regions, which means that we have 99 male users and 99 female users for each region. The annotation was done based on guidelines and criteria that were used to distinguish male and female users. For every user, the annotator determined the gender by analyzing the username, profile picture, and indicative words. The annotator checked the username if it is denoting a male or female name. Some users put their real photos in their Twitter profile, which also helped in determining their gender. In some cases, the username and the profile picture did not give sufficient information to identify their gender. Therefore, the annotators were looking for some indicative words in the user' s tweets. For example, the adjectives that describe feminine in Arabic usually end with ـة *(Taa' Marbootah)*. Therefore, finding words that end with the Arabic letter Taa' Marbootah such as بردانة *(Feeling cold)*, نعسانة *(Feeling sleepy)*, or جعانة *(Feeling hungry)* in the tweets of a user indicates that the writer is a female. However, if the writer is a male, the adjectives used in his tweets would not have that letter. The annotators used this Arabic rule to determine the user's gender when it was not possible to do so using the username or the profile photo. It is noteworthy that users for which the gender could not determined were not included in the corpus and were replaced by other users for which the gender could be determined as explained above.

3.3.2 Age Annotation

We annotated the age of the selected Twitter users using three age groups: under 25, 25 until 34, and 35 and above. In many cases, we were able to find the exact ages whereas in some other cases we were able to determine the correct age group without the exact age. The annotators went through the following steps to determine the age of a user:

Getting the exact age: Several Twitter users include their birth year as a part of their usernames. For example, in the username *Omda1981m* the birth year is most likely 1981, which indicates that this user is 38 years old. Some other users put their date of birth in their Twitter biography or in their other social media accounts such as Facebook, Instagram, or LinkedIn. Other users put their exact age in some of their tweets and we were able to search for those tweets using some relevant keywords in different languages as shown in Table 2.

Getting the estimated age: There were several cases in which we were not able to determine the exact age even after following the abovementioned steps. In these cases, we opted to determine the approximate ages, so that we could annotate the user with the correct age group. This was done by searching for the user on other social media networks such as LinkedIn and Facebook. We often found either their exact ages or other age related hints such as the year of their graduation from university, which helped us in making an educated guess towards the user's age. Furthermore, as a last resort for some few users, two annotators guessed the age separately using the Twitter photo. Then, we used Microsoft's online tool[2] that predicts the age by analyzing the facial features of a user through machine learning. Only users were included for which the age group guessed by Microsoft's age determination tool.

3.3.3 Dialect Annotation

As mentioned above, we selected the users for our corpus by searching for tweets with dialect spe-

[2] www.how-old.net

Region	Unique Word	User's Tweet
Lebanon-Syria	هيدا	عملت غوغل و **هيدا** يلي طلع معي
Tunisia	برشا	ذقت أنواع كسكسي متاع **برشا** دول يبقى الألذ والافضل التونسي
Iraq	اهوايه	اي والله صحيح **اهوايه** اكو هيجي نماذج مع الاسف
KSA	**جحفله**	صدقوني الموضوع فيه إنّ وفيه **جحفله**
Palestine-Jordan	هاظ	طبعاً **هاظ** اشي مفروغ منه
Qatar	احاتي	انا لعبتي **احاتي** الدراسة بس ما ادرس
Kuwait	طنطل	عاد انتي دلو مفروض تكونين **طنطل** شفيكم يا جماعه
Sudan	زول	تاني ما هعمل ريتويت لي اي **زول**
UAE	مايو ب	تسلمين يالوافيه يا راعيه الذوق يالراقيه دوم راعيه **مايوب**
Yemen	معادبش	يعني معقوله **معادبش** ولا شي كملوها السرق
Morocco	دابا	كنت ديما كنتسائل أش كاين فالكتاب **دابا** لقيت الجواب الشافي
Algeria	زاوش	كنا انا وخويا وولد عمي نصيدو **زاوش** ونسلخوه ونديروه مشوي
Libya	شهدره	عليش **شهدره** اسمي في ريبلي
Egypt	دلوقتي	سيبوني **دلوقتي** انا كويس
Oman	شقبه	انزين عادي م خسرانه بطلعي **شقبه** صدقيني

Table 1: Examples of unique words from each dialectal region with tweets that include these words.

Indicative word	Language	Tweet Example
(My birthday) ميلادي	Arabic	اكتوبر ١٩٩٠ عيد **ميلادي**
(My age) عمري	Arabic	الليله اخر يوم لي وأنا **عمري** ٢٢
(Year) سنة	Arabic	بكون ٢٣ **سنة** ان شاء الله في مونديال ٢٠٢٢
Turn	English	Hey Sosoo I will **turn** 20 in few days
(Years) Ans	French	16 **ans**?? J'ai 35 ans ma chérie

Table 2: Examples of keywords from different languages used to determine the users age.

cific words and then retrieving the users of those tweets and their followers. In addition, once we retrieved the user's tweets, the annotator(s) manually went through them and checked that at least half of the tweets are in the user's respective dialect. When the users' tweets were not mostly in their originally identified dialect, the annotators were instructed to replace them with other users from the same dialect, the same gender, and the same age group.

3.4 Tweets Retrieval

Twitter offers an Application Programming Interface (API) that allows developers to interact with Twitter's web services. We used the Python programming language with the *tweepy.py*[3] library to interact with the Twitter API and retrieve each user's timeline (tweets). Due to restrictions by Twitter, we were only allowed to retrieve at most 3,200 of the users most recent Tweets. The number of tweets of every user in our corpus ranges from a required minimum of 100 tweets to a maximum of 3,200 tweets. After collecting the maximum possible number of tweets for each user, we filtered them. Specifically, we removed the non-Arabic tweets, retweets, and short tweets that contain less than 3 words. Eventually, we kept only the Twitter accounts who had at least 100 original users tweets that are in Arabic and that have

[3] http://docs.tweepy.org/en/v3.5.0/index.html

three words at least. At the end of this process, we were able to have a well-balanced multi-dialect corpus of 198 users for 15 regions spanning 17 Arab countries. The corpus is equally-balanced with respect to gender and age.

4 Verification and Evaluation

Our main objective was to produce a multi-dialectal corpus of tweets with a very good quality of annotation so that it can be used to promote Arabic NLP research including research on author profiling. This is a major improvement compared to the previous version of the corpus. To ensure a good quality annotation, we performed two rounds of annotation for each Arab region, which means that each region was annotated by two different annotators. Next, we report on the verification and evaluation process for the annotation of dialect, age, and gender. Moreover, we present the evaluation results for each annotation category.

4.1 Verification of Dialect Annotation

The annotation was done by experienced Arab annotators who were asked to go through each file of retrieved users tweets and check the dialect used in these tweets. This step was necessary to verify the overall dialect of all tweets for every user, and not only the last tweets in a users timeline. Based on this dialect verification, we did the following:

Removing the users who post tweets in multiple dialects: many Arab users do not live in their original countries, and consequently, their dialects are affected by the dialects of the countries they live in. We opted to drop such users from our corpus and we replaced them by users from the same gender and age group and which used one dialect only.

Removing the users who have many tweets in MSA: some Arab users post few tweets in their own dialects, and they write their tweets mostly in MSA. After checking all retrieved tweets of a given user, we included in our corpus only those who have more than half of their tweets written in their dialect.

4.2 Verification of Age and Gender Annotation

Although the age was determined manually, the corpus included initially some users for which neither the exact age nor the age group could be determined with high confidence. These cases occurred for example when the age of the user was determined mainly by using their Twitter profile photo.

Moreover, the annotators found initially some difficulties with determining the gender for some users for instance when the profile did not include a real name nor a profile photo. For the latter cases, the accounts were replaced by accounts from the same gender and age group for which the age could be determined with higher confidence. For all accounts, a second round of annotation was performed by a different annotator. At the end of this phase, we were able to have two annotations from different contributors for each user account. Eventually, variations were checked and resolved by a member of the research team together with the annotators by reviewing and discussing the provided justifications for a certain annotation of age, gender, or dialect.

4.3 Annotation Evaluation

In order to evaluate the quality of the annotation, we used the inter-annotator agreement measures. We were able to have two rounds of annotation by different annotators for each user and this applies for the gender, dialect and age group for the whole corpus unlike in the previous version (Zaghouani and Charfi, 2018a) in which the inter-annotator agreement was based on a 10 % subset of the corpus data.

We measured the inter-annotator agreement using Cohens Kappa, which is a statistical measure of the degree of agreement for a data point (gender, dialect, and age in our case), that was labeled by two annotators, over what would be expected by chance. We obtained substantial results for the agreement with average Kappa values of **0.99**, **0.92**, and **0.88** respectively for the annotation of gender, dialect, and age. The exact Kappa values per region are shown in Table 3.

5 Challenges

In the following, we report on some challenges that we faced when developing this corpus:

First, we encountered some difficulties in finding user accounts for some age groups for certain regions. This was the case for Sudani, Iraqi, and Gulf females whose age is 35 or above. For some Arab regions, female Twitter users tend to hide their real name and also avoid putting their photos on social media because of the local cul-

Region	**Gender**	**Age**	**Dialect**
Lebanon-Syria	1	0.924	1
Tunisia	1	0.742	1
Iraq	1	0.696	0.9
Saudi Arabia	1	0.856	0.9
Palestine-Jordan	0.989	0.856	0.9
Qatar	0.969	0.901	0.9
Kuwait	0.979	0.886	0.9
Sudan	1	0.734	0.8
UAE	0.979	0.848	0.8
Yemen	1	0.984	1
Morocco	1	0.954	0.9
Algeria	0.959	0.931	0.9
Libya	0.989	0.924	1
Egypt	1	0.969	1
Oman	0.989	0.931	0.9
Overall	0.99	0.88	0.92

Table 3: Kappa values per region for gender, age, and dialect annotation.

ture and norms. This was an issue for us as we sometimes depend on the Twitter profile photos to guess the age especially if the username is not a real name. Second, we notice that older Arabs often write in Modern Standard Arabic (MSA) on social media, and this made the task of finding dialect users above 35 years much harder. Third, we faced an issue with Maghrebi users who often tweet in French more than Arabic.

To address these issues, we put more effort into the user selection and we replaced any users who did not fulfill the criteria explained above.

Fourth, we noticed that some users use more than one Arabic dialect to write their tweets on Twitter. For example, in Arab regions with high immigration rates such as Gulf countries, the dialect of the residents is sometimes affected by the language of their host country. Consequently, these users tweet in multiple dialects and they might mix dialects even in one same tweet. Other users had similar issues because their parents come from different Arab countries. To address this issues, all tweets were manually reviewed and users tweeting in two dialects or more were replaced by mono-dialectal users.

Fifth, we worked on regions sequentially, i.e., we selected a large set of accounts from Twitter for one region, annotated them, and then retrieved their tweets. Then, we worked on the next region. As a result, when we retrieved again the users or their tweets we noticed that some profiles were either made private or even closed. In such cases, we were no longer able to access the users profile and tweets. We addressed this problem by periodically checking the users' accounts and replacing any accounts that were deactivated or made private.

6 Conclusion

We presented in this paper a significantly improved version of a large-scale manually-annotated and fine-grained multi-dialectal corpus of Arabic tweets, which is compsed of 5 millions dialectal Arabic tweets of about 3000 Twitter users from 17 Arab countries. To the best of our knowledge, our corpus is the most comprehensive dialectal corpus in terms of coverage for so many Arab dialects and the data volume. Moreover, our corpus is balanced with respect to dialect, gender, and age. The corpus was annotated manually based on well-defined annotation guidelines and it was fully evaluated using the inter-annotation agreement measures.

The corpus was already and it can further be used to promote research on Arabic NLP including author profiling (Rosso et al., 2018), authorship attribution, dialect identification, sentiment analysis in dialectal Arabic, and bots detection in dialectal Arabic.

Acknowledgments

We thank our dedicated annotators: Hoda Ibrahim, Safaa Ellaham, Syrine Guediche for their hard work on the corpus annotation. This publication was made possible by NPRP 9-175-1-033 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

References

Faisal Al-Shargi, Aidan Kaplan, Ramy Eskander, Nizar Habash, and Owen Rambow. 2016. Morphologically annotated corpora and morphological analyzers for moroccan and sanaani yemeni arabic. In *10th Language Resources and Evaluation Conference (LREC 2016)*.

Nora Al-Twairesh, Rawan Al-Matham, Nora Madi, Nada Almugren, Al-Hanouf Al-Aljmi, Shahad Alshalan, Raghad Alshalan, Nafla Alrumayyan, Shams Al-Manea, Sumayah Bawazeer, et al. 2018. Suar: Towards building a corpus for the saudi dialect. *Procedia computer science* 142:72–82.

Israa Alsarsour, Esraa Mohamed, Reem Suwaileh, and Tamer Elsayed. 2018. Dart: A large dataset of dialectal arabic tweets. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Houda Bouamor, Nizar Habash, and Kemal Oflazer. 2014. A multidialectal parallel corpus of arabic. In *LREC*. pages 1240–1245.

Houda Bouamor, Nizar Habash, Mohammad Salameh, Wajdi Zaghouani, Owen Rambow, Dana Abdulrahim, Ossama Obeid, Salam Khalifa, Fadhl Eryani, Alexander Erdmann, et al. 2018. The madar arabic dialect corpus and lexicon. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Mona Diab and Nizar Habash. 2007. Arabic dialect processing tutorial. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Tutorial Abstracts*. Association for Computational Linguistics, pages 5–6.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 426–432.

Nizar Y Habash. 2010. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies* 3(1):1–187.

Mustafa Jarrar, Nizar Habash, Diyam Fuad Akra, and Nasser Zalmout. 2014. Building a corpus for palestinian arabic: a preliminary study .

Salam Khalifa, Nizar Habash, Dana Abdulrahim, and Sara Hassan. 2016. A large scale corpus of gulf arabic. *arXiv preprint arXiv:1609.02960* .

Salam Khalifa, Nizar Habash, Fadhl Eryani, Ossama Obeid, Dana Abdulrahim, and Meera Al Kaabi. 2018. A morphologically annotated corpus of emirati arabic. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, Mona T Diab, Nizar Habash, Owen Rambow, and Dalila Tabessi. 2006. Developing and using a pilot dialectal arabic treebank. In *LREC*. pages 443–448.

Mohamed Maamouri, Ann Bies, Seth Kulick, Michael Ciul, Nizar Habash, and Ramy Eskander. 2014. Developing an egyptian arabic treebank: Impact of dialectal morphology on annotation and tool development. In *LREC*. pages 2348–2354.

Mohamed Maamouri, Ann Bies, Seth Kulick, Wajdi Zaghouani, David Graff, and Michael Ciul. 2010. From speech to trees: Applying treebank annotation to arabic broadcast news. In *LREC*.

Hamdy Mubarak and Kareem Darwish. 2014. Using twitter to collect a multi-dialectal corpus of arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. pages 1–7.

Martha Palmer, Olga Babko-Malaya, Ann Bies, Mona T Diab, Mohamed Maamouri, Aous Mansouri, and Wajdi Zaghouani. 2008. A pilot arabic propbank. In *LREC*.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*. volume 14, pages 1094–1101.

Paolo Rosso, Francisco M. Rangel Pardo, Bilal Ghanem, and Anis Charfi. 2018. ARAP: arabic author profiling project for cyber-security. *Procesamiento del Lenguaje Natural* 61:135–138.

Clare R Voss, Stephen Tratz, Jamal Laoudi, and Douglas M Briesch. 2014. Finding romanized arabic dialect in code-mixed tweets. In *LREC*. pages 2249–2253.

Wajdi Zaghouani and Anis Charfi. 2018a. Arap-tweet: A large multi-dialect twitter corpus for gender, age and language variety identification. *arXiv preprint arXiv:1808.07674* .

Wajdi Zaghouani and Anis Charfi. 2018b. Guidelines and annotation framework for arabic author profiling. *arXiv preprint arXiv:1808.07678* .

Wajdi Zaghouani, Mona Diab, Aous Mansouri, Sameer Pradhan, and Martha Palmer. 2010. The revised arabic propbank. In *Proceedings of the fourth linguistic annotation workshop*. Association for Computational Linguistics, pages 222–226.

Wajdi Zaghouani, Abdelati Hawwari, and Mona Diab. 2012. A pilot propbank annotation for quranic arabic. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*. pages 78–83.

Rabih Zbib, Erika Malchiodi, Jacob Devlin, David Stallard, Spyros Matsoukas, Richard Schwartz, John Makhoul, Omar F Zaidan, and Chris Callison-Burch. 2012. Machine translation of arabic dialects. In *Proceedings of the NAACL 2012*. pages 49–59.

Personality-dependent Neural Text Summarization

Pablo Botton da Costa
University of São Paulo
São Paulo, Brazil
pablo.botton.costa@gmail.com

Ivandré Paraboni
University of São Paulo
São Paulo, Brazil
ivandre@usp.br

Abstract

In Natural Language Generation (NLG) systems, personalization strategies - i.e., the use of information about a target author to generate text that (more) closely resembles human-produced language - have long been applied to improve results. The present work addresses one such strategy - namely, the use of Big Five personality information about the target author - applied to the case of abstractive text summarization using neural sequence-to-sequence models. Initial results suggest that having access to personality information does lead to more accurate (or human-like) text summaries, and paves the way for more robust systems of this kind.

1 Introduction

Computational approaches to text summarization may be divided into two general categories: abstractive and extractive summarization. Extractive summarization consists of selecting relevant pieces of text to compose a subset of the original sentences, whereas the more complex abstractive summarization involves interpreting the input text and rewriting its main ideas in a new, shorter version. Both strategies may be modelled as a machine learning problem by making use of unsupervised (Ren et al., 2017), graph-based and neural methods (Wan and Yang, 2006; Cao et al., 2015), among others. The present work focuses on the issue of neural abstractive summarization, addressing the issue of personalized text generation in systems of this kind.

Text-generating systems may in principle produce always the same fixed output from a given input representation. In order to generate more natural (or 'human-like') output, however, systems of this kind will often implement a range of stylistic variation strategies. Among these, the use of computational models of *human personality* has emerged as a popular alternative, and it is commonly associated with the rise of the Big Five model of human personality (Goldberg, 1990) in many related fields.

The Big Five model is based on the assumption that differences in personality are reflected in natural language use, and comprises five fundamental dimensions of personality: *Extraversion*, *Agreeableness*, *Conscientiousness*, *Neuroticism*, and *Openness to experience*. Given its linguistic motivation, the Big Five personality traits have been addressed in a wide range of studies in both natural language understanding and generation alike. Thus, for instance, the work in Mairesse and Walker (2007) introduces PERSONAGE, a fully-functional NLG system that produces restaurant recommendations. PERSONAGE and many of its subsequent extensions support multiple stylistic variations that are controlled by personality information provided as an input.

The use of personality information for text summarization, by contrast, seems to be far less common, and we are not aware of any existing work that addresses the issue of personality-dependent neural text summarization. Based on these observations, this paper introduces a personality-dependent text summarization model that makes use of a corpus of source and summary text pairs labelled with personality information about their authors. In doing so, our goal is to use personality information to generate summaries that more closely resemble those produced by humans.

The rest of this paper is structured as follows. Section 2 discusses the issues of sequence-to-sequence learning and attention mechanism for text summarization. These are the basis of our current work described in Section 3. Section 4

Proceedings of Recent Advances in Natural Language Processing, pages 205–212,
Varna, Bulgaria, Sep 2–4 2019.

reports two experiments comparing the proposed models against a number of alternatives, and Section 5 presents final remarks and future work.

2 Background

Due to the capacity of neural language generation models to learn and automatically induce representations from text (Rush et al., 2015; Nallapati et al., 2016; Mikolov et al., 2013), neural abstractive summarization has attracted a great deal of attention in the field. Architectures of this kind may not only produce high-quality summaries, but may also embed external information easily (See et al., 2017). Accordingly, these models have achieved significant results, at least in terms of intrinsic evaluation measures such as BLEU (Papineni et al., 2002) or ROUGE (Lin and Hovy, 2003), when comparing to extractive approaches (Celikyilmaz et al., 2018).

2.1 Sequence-to-sequence Learning

Neural text summarization models are often grounded on a particular kind of neural network, the sequence-to sequence architecture (Sutskever et al., 2014a; Cho et al., 2014). In models of this kind, input text is modelled as a sequence of representations carrying any contextual information from end to end in the generation process. More formally, a sequence-to-sequence model is defined in Goodfellow et al. (2016) as a neural network that directly models the conditional probability $p(y|x)$ of a source sequence, $x_1, ..., x_n$, to a target sequence, $y_1, ..., y_m$[1].

A basic form of sequence-to-sequence model consists of two main components: (i) an encoder that computes a representation s for each source sequence; and (ii) a decoder that generates one target token at a time, decomposing the conditional probability as follows:

$$p(y|x) = \sum_{j=1}^{m}(y_j|y_{<j}, s)$$

A common strategy for learning sequence representations is by making use of Recurrent Neural Networks (RNN) (Rumelhart et al., 1986). According to Hochreiter and Schmidhuber (1997), a RNN generalizes the concept of feed-forward neural network to sequences. Given a temporal sequence of inputs $(x_1, ..., x_t)$, the standard RNN computes a sequence of outputs $(y_1, ..., y_t)$ mapped onto sequences using the following equation (Sundermeyer et al., 2012):

$$h_t = sigmoid(W^{hx}x_t + W^{hh}h_{t-1})$$
$$y_t = W^{yh}h_t$$

A simple strategy for general sequence learning is to map the input sequence to a fixed-sized vector using a RNN, and then map the vector to the target sequence by using a second RNN. This may in principle be successful, but long term dependencies may make the training of the two networks difficult (Bengio et al., 1994; Hochreiter, 1998). As an alternative, Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and their simplification known as Gated Recurrent Unit (GRU) (Cho et al., 2014) are known to learn problems with long range temporal dependencies, and may therefore succeed in this setting.

The goal of a LSTM/GRU network is to estimate the conditional probability $p(y|x)$, where $(x_1, ..., x_{t'})$ is an input sequence and $(y_1, ..., y_t)$ is its corresponding output sequence whose length t' may differ from t (Cho et al., 2014). The conditional probability is computed by first obtaining the fixed dimensional representation v of the input sequence $(x_1, ..., x_{t'})$ given by the last hidden state of the network, and by computing the probability of $(y_1, ..., y_t)$ with a standard LSTM/GRU formulation in which the initial hidden state is set to the representation v of $(x_1, ..., x_{t'})$. Finally, each $p(y_j|s, y_1, ..., y_{j-1})$ distribution is represented with a softmax over all the words in the vocabulary.

GRUs are distinct from LSTMs in that a GRU architecture contains only a single unit to control when the current states 'forgets' a piece of information (Goodfellow et al., 2016). Due to this simplification, GRUs can directly access all hidden states without bearing the price of a memory state (Cho et al., 2014).

GRU architectures model sequences as causal relationships through the input sequence by examining left-to-right relationships only (Goodfellow et al., 2016). However, many sequence classification problems may require predicting an output that depends (bidirectionally) on the entire input sequence, that is, from left to right and also from right to left. This is the case, for instance, of a large number of common NLP applications that

[1] Sentences are assumed to start with a special 'start-of-sentence' token $< bos >$ and end with an 'end-of-sequence' token $< eos >$.

need to pay regard to contextual dependency when modelling phrases and sentences.

Bidirectional GRUs (Bi-GRUs) are applied to a wide range of tasks to scan and learn both left-to-right and right-to-left dependencies, which can capture complementary types of information from its inputs. The left and right hidden representations produced by GRUs can be linearly combined (θ) to form a final representation (Goodfellow et al., 2016): $h_t = h_t^{\leftarrow} \ \theta \ h_t^{\rightarrow}$.

2.2 Attention Mechanism

Sequence-to-sequence architectures have been successfully applied to a wide range of tasks, including machine translation and natural text generation (Cho et al., 2014; Sutskever et al., 2014a) and, accordingly, have been subject to a great deal of extensions and improvements. Among these, the use of more context-aware sequence generation methods (Cho et al., 2014) and the use of attention mechanism to score and select words that best describe the intended output are discussed below.

In natural language generation, attention models as introduced in Cho et al. (2014) and Sutskever et al. (2014a) are intended to generalize the text generation task so as to handle sequence pairs with different sizes of inputs and outputs. This approach, subsequently called sequence-to-sequence with attention mechanism, applies a mapping strategy from a variable-length sentence to another variable-length sentence. This mapping strategy is a scoring system over the contextual information from the input sequence (Cho et al., 2014), making a set of attention weights.

Attention-based models (Sutskever et al., 2014b; Luong et al., 2015) are sequence-to-sequence networks that employ an encoder to represent the text utterance and an attention-based decoder that generates the response, one token at a time. More specifically, neural text summarization can be viewed as a sequence-to-sequence problem (Sutskever et al., 2014a), where a sequence of input language tokens $x = x_1, ..., x_m$ describing the input text are mapped onto a sequence of output language tokens $y_1, ..., y_n$ describing the target text output. The encoder is a GRU unit (Cho et al., 2014) that converts $x_, ..., x_m$ into a sequence of context-sensitive embeddings $b_1, ..., b_m$. A general-attention decoder generates output tokens one at a time. At each time step j, the decoder generates y_j based on the current hidden state s_j, and then updates the hidden state s_{j+1} based on s_j and y_j. Formally, the attention decoder is defined by original equation proposed in Cho et al. (2014):

$$s_1 = tanh(W^{(s)} b_m)$$
$$p(y_j = w|x, y_{1:j-1}) \ \alpha \ exp(U[s_j, c_j])$$
$$s_{j+1} = GRU([\phi^{(out)}(y_j), c_j], s_j)$$

where $i \ \epsilon \ \{1, ..., m\}$, $j \ \epsilon \ \{1, ..., m\}$ and the context vector c_j, is the result of general attention (Luong et al., 2015). The matrices $W^{(s)}$, $W^{(\alpha)}$, U and the embedding function $\phi^{(out)}$ are decoder parameters.

3 Current Work

Our basic model is generally inspired from the architecture in Cho et al. (2014), with an added personality embedding layer. As in many other sequence-to-sequence models with attention, our model takes as an input a sentence, and produces as an output a set of words that summarizes the given input. The actual rendering of this output as structured text is presently not addressed.

The proposed architecture is illustrated in Figure 1, which is adapted from Cho et al. (2014), and further discussed below.

In this example, $B \ B \ B \ B$ represent the input sequence from the target sequence $Z \ X$, and C is the personality embedding representation. The five main components of the architecture are as follows.

(A) a bidirectional GRU that maps words to personality types

(B) a word embedding layer

(C) a personality embedding layer

(D) an attention mechanism

(E) a bidirectional GRU that outputs word encodings

The input bidirectional GRU (A) produces a word-to-personality compositional representation of each word. This serves two main purposes: combining the composite sequences of words and personality information, and combining attention weights over sequences in our decoder model.

The word embeddings layer (B) produces a typical word-level representation of each input word. In the present work, we make use of both random

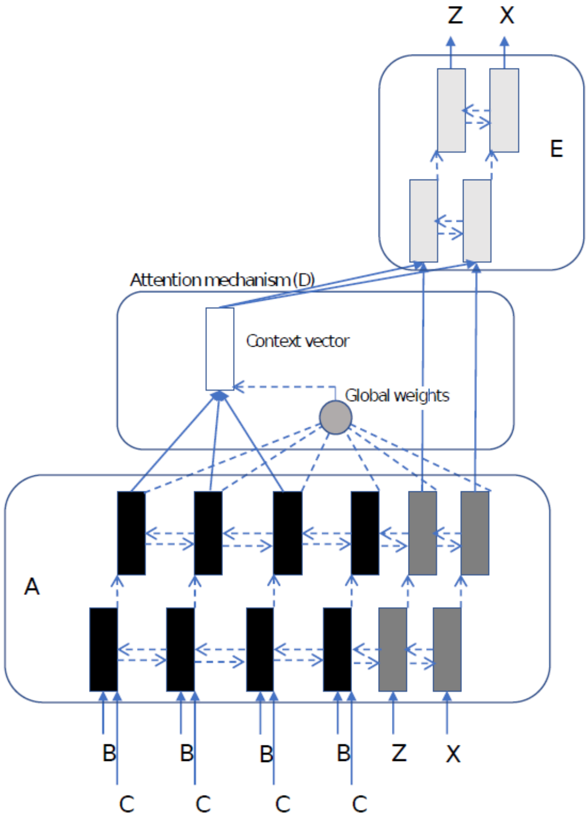


Figure 1: Model architecture

and pre-trained word embeddings. The latter are Skip-gram 300 word embeddings taken from Hartmann et al. (2017).

Word embeddings are complemented with induced personality embeddings (C) for each target author. The role of this layer is twofold. First, it is intended to learn the probability $P(Y|X, personality)$, that is, the $personality$ representation of each author for each word in the vocabulary. Second, this layer is also intended to decide which profile value should be selected (from the corpus gold standard annotation) in order to generate a summary.

The attention mechanism (D) attempts to learn a general representation from the most important parts of the input text at each time step. To this end, the experiments described in the next section will consider two score function alternatives: general attention and dot product.

Finally, the output bidirectional GRU (E) combines the attention weight representations, and produces a final encoding for each word. A loss function describe the overall generation probability, and it is intended to optimize the above parameters. This function is described as follows.

$$\ell_1(\theta, D(c), D(x,y))$$

$$= - \sum_{(X,Y)\ \epsilon\ D^c \cup D^{pr}} logP(Y|X, < k_i, v_i >)$$

$$= - \sum_{(X,Y)\ \epsilon\ D^c} logP^{fr}(Y|X)$$

The first term of the function is the negative log likelihood of observing $D^{(c)}$ and the second term for $D^{(pr)}$. $D^{(pr)}$ consists of pairs where a summary is related to a profile key and its response match to the summary, and $D^{(c)}$ has only general text-summary pairs. $< k_i, v_i >$ is the personality representation. The decoder P^{fr} does not have shared parameters. A simple epoch-based training strategy using gradient descendent is performed.

4 Evaluation

We envisaged two experiments on neural text summarization based on the model described in the previous section. The first experiment aims to assess whether a general or a dot product attention mechanism is more suitable to the task. The second experiment focuses on our main research question, that is, on whether the use of personality information does improve summarization results.

As in many (or most) sequence-to-sequence approaches to text generation, our work focuses on the selection of text segments to compose an abstract summary, but it does not address the actual rendering of the final output text, which would normally require additional post-processing. Each of the two experiments is discussed in turn in the following sections, but first we describe the dataset taken as their basis.

4.1 Data

We make use of the text and caption portions of the b5 corpus in Ramos et al. (2018), called *b5-text* and *b5-caption*. The corpus conveys 1510 multi- and single-sentence image description pairs, all of which labelled with Big Five personality information about their authors. Table 1 summarizes the corpus descriptive statistics.

The corpus was elicited from a crowd sourcing task in which participants were requested to provide both long and short descriptions for 10 stimulus images taken from GAPED, a database of images classified by valence and normative significance designed to elicit various reactions (Dan-

Table 1: Corpus descriptive statistics.

Data	Words	Average	Types	Average
text	84463	559.4	37210	246.4
caption	4896	32.4	4121	27.3

Figure 2: Stimulus image from *GAPED* (Dan-Glauser and Scherer, 2011).

Glauser and Scherer, 2011). From a set of 10 selected images with valence degrees in the 3 to 54 range, participants were first instructed to describe everything that they could see in the scene (e.g., as if helping a visually-impaired person to understand the picture) and, subsequently, were requested to summarize it in a single sentence (similar to a picture caption.)

An example of stimulus image is illustrated in Figure 2. We notice however that in the present work we only consider the text elicited from these images, and not the images themselves.

Based on scenes as in Figure 2, the following is a possible long description (translated from the Portuguese original text) of the kind found in the corpus.

> '*A black baby, about one year old. He's in a cradle. He is dressed in a dirty blue blouse, on a pink sheet, without a pillow. A blue blanket is next to the baby. It seems that he has not taken a shower for a while.*'

A single-sentence summary for the same scene (and which would have been written by the same participant in the data collection task) may be represented as the following example.

> '*A sad-looking baby.*'

In the experiments described in the next sections, texts were pre-processed by removing punctuation and numerical symbols. In addition to that, the first data split performed for the purpose of cross-validation is shown in Table 2.

Table 2: Data split

Split	Samples
Train	1358
Validation	152
Total	1510

4.2 Experiment 1: Basic Neural Summarization with Attention Mechanism

In Encoder-Decoder Recurrent Neural Networks, the global attention mechanism may be seen as a model-inferred context vector computed as a weighted average of all inputs by making use of a score function. The choice of score function may have a great impact on the overall performance of the model, and for that reason in what follows we examine two alternatives: using the dot product over the context vectors of the source, and using the learned representation over the context states.

To this end, our first experiment evaluates our basic summarization model (cf. the previous section) in two versions, namely, using general and dot product attention mechanisms. Both of these models, hereby called *sDot* and *sGen*, will make use of encoder/decoder randomized word embedding of size 300, and two encoder/decoder hidden units of size 600.

Both models were trained using Adam optimization with mini batches of size 128. The initial learning rate was set to 0.0001 with a gradient clipping based on the norm of the values. We also applied different learning rates for the decoder module, set to five times the learning rate of the encoder. In order to reduce over-fitting, a 0.5 drop-out regularization was applied to both embedding layers.

Model optimization was performed by using gradient descendent with masked loss, and by applying early stopping when the BLEU scores over the evaluation dataset did not increase for 20 epochs. Except for the embedding layer, all other

Table 3: 10-fold cross validation BLEU scores for text summarization using dot product (sDot) and general (sGen) attention. the best result is highlighted.

Model	BLEU
sGen	**13.88**
sDot	13.63

parameters were initialized by sampling from a uniform distribution $U(-sqrt(3/n), sqrt(3/n))$, where n is the parameter dimension.

We performed 10-fold cross-validation over our corpus data, and we compared the output summaries produced by both models using BLEU[2]. Results are presented in Table 3.

From these results, we notice that the attention mechanism based on the general function in *sGen* outperforms the use of dot function in *sDot*. Although the difference is small, the use of a generalized network to learn how to align the contextual information is superior to simply concatenating contextual information obtained from the global weights. Based on these results, the general attention strategy will be our choice for the next experiment.

4.3 Experiment 2: Personality-dependent Summarization

Our second and main experiment assesses the use of personality information in text summarization. To this end, two models are considered: the full personality-aware model presented in Section 3, hereby called *sPers*, and a simplified baseline version of the same architecture without access to personality information, hereby called *sBase*. In doing so, our goal is to show that summaries produced by *sPers* resemble the human-made texts (as seen in the corpus) more closely than those produced by *sBase*.

Both *sPers* and *sBase* make use of pre-trained skip-gram 300 word embeddings for the Brazilian Portuguese language taken from Hartmann et al. (2017). Both models also make use of encoder/decoder randomized word embedding of size 300, and two encoder/decoder hidden units of size 600 with general attention.

Table 4: 10-fold cross validation BLEU scores for text summarization with (sPers) and without (sBase) personality information. The best result is highlighted.

Model	BLEU
sBase	14.21
sPers	**14.58**

All optimization, training and other basic procedures are the same as in the previous experiment. Results are presented in Table 3.

We notice that personality-dependent summarization as provided by *sPers* outperforms standard summarization (i.e., with no access to personality information) as provided by *sBase*. Although the difference is once again small (which may be explained by the limited size of our dataset), this outcome offers support to our main research hypothesis by illustrating that the use of author personality information may improve summarization accuracy.

4.4 Selected Examples

As a means to illustrate the kinds of output that may be produced by our models, Table 5 presents a number of examples taken from the original corpus summaries, and the corresponding summaries obtained from the same input by making use of the *sBase* baseline and by the personality-dependent *sPers* models. For ease of illustration, the examples are informally grouped into three error categories (small, moderate and large) according to the distance between the corpus summaries and their *sPers* counterparts, and are presented in both original (Portuguese) and translated (English) forms.

5 Final Remarks

This paper addressed the use of Big Five personality information about the target author to generate personalized summaries in neural sequence-to-sequence text summarization. The model - consisting of two bidirectional GRUs, word embeddings and attention mechanism - was evaluated in two versions, namely, with and without an additional personality embedding layer. Initial results suggest that having access to personality information does lead to more accurate (or human-like) text summaries.

The use of personality information is of course only one among many possible personalization

[2] We are aware that, although popular in machine translation and text generation, BLEU may not be the ideal metrics for the present task (Liu et al., 2011; Song et al., 2013), and that it may not co-relate well with, e.g., human judgments (Reiter and Belz, 2009).

Table 5: Selected examples taken from the corpus, baseline (sBase) and personality-dependent (sPers) summarization models, grouped by distance (small, moderate or large) between sPers and the expected (corpus) summary in original Portuguese (Pt) and translated English (En).

Error	Model	Summary (Pt)	Summary (En)
small	corpus	*homem na cerca*	*man by fence*
	sBase	*homem idoso*	*elderly man*
	sPers	*homem na cerca*	*man by fence*
moderate	corpus	*pessoas pedindo ajuda*	*people asking for help*
	sBase	*pessoas esperando*	*people waiting*
	sPers	*pessoas aguardam atendimento*	*people waiting for help*
large	corpus	*menino com um balde de terra*	*boy with a bucket full of soil*
	sBase	*crianca com balde*	*child with bucket*
	sPers	*crianca com balde de terra*	*child with bucket full of soil*

strategies for text summarization. In particular, we notice that the increasing availability of text corpora labelled with author demographics in general (e.g., gender, age, education information etc.) may in principle support a broad range of speaker-dependent summarization models. Thus, as future work we intend to extend the current approach along these lines, and provide additional summarization strategies that may represent more significant gains over the standard, fixed-output summarization approach.

Acknowledgements

The authors acknowledge support by FAPESP grant 2016/14223-0 and from the University of São Paulo.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Ziqiang Cao, Furu Wei, Sujian Li, Wenjie Li, Ming Zhou, and Houfeng Wang. 2015. Learning summary prior representation for extractive summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. volume 2, pages 829–833.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 1662–1675.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, pages 103–111.

Elise S. Dan-Glauser and Klaus R. Scherer. 2011. The Geneva affective picture database (GAPED): a new 730-picture database focusing on valence and normative significance. *Behavior Research Methods* 43(2):468–477.

Lewis R. Goldberg. 1990. An alternative description of personality: The Big-Five factor structure. *Journal of Personality and Social Psychology* 59:1216–1229.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Nathan Hartmann, Erick Fonseca, Christopher Shulby, Marcos Treviso, Jéssica Rodrigues, and Sandra Aluísio. 2017. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. In *11th Brazilian Symposium in Information and Human Language Technology - STIL*. Uberlândia, Brazil, pages 122–131.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(02):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Chin-Ye Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL 2003*. Association for Computational Linguistics, Edmonton, Canada, pages 71–78.

Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. 2011. Better evaluation metrics lead to better machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 375–384.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

François Mairesse and Marilyn Walker. 2007. PERSONAGE: Personality generation for dialogue. In *45th Annual Meeting-Association For Computational Linguistics*. Association for Computational Linguistics (ACL), Sheffield, pages 496–503.

Tomas Mikolov, Scott Wen-tau, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proc. of NAACL-HLT-2013*. Association for Computational Linguistics, Atlanta, USA, pages 746–751.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Berlin, Germany, pages 280–290. https://doi.org/10.18653/v1/K16-1028.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceddings of ACL-2002*. Association for Computational Linguistics, Philadelphia, PA, USA, pages 311–318.

Ricelli Moreira Silva Ramos, Georges Basile Stavracas Neto, Barbara Barbosa Claudino Silva, Danielle Sampaio Monteiro, Ivandré Paraboni, and Rafael Felipe Sandroni Dias. 2018. Building a corpus for personality-dependent natural language understanding and generation. In *11th International Conference on Language Resources and Evaluation (LREC-2018)*. ELRA, Miyazaki, Japan, pages 1138–1145.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics* 35(4):529–558.

Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 95–104.

David E. Rumelhart, Geoffrey Hinton, and Ronald J. Williams. 1986. Learning representations by back propagating errors. *Nature* 323:533–536. https://doi.org/10.1038/323533a0.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 379–389. https://doi.org/10.18653/v1/D15-1044.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 1073–1083.

Xingyi Song, Trevor Cohn, and Lucia Specia. 2013. Bleu deconstructed: Designing a better mt evaluation metric. *International Journal of Computational Linguistics and Applications* 4(2):29–44.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014a. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014b. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the human language technology conference of the NAACL, Companion volume: Short papers*. Association for Computational Linguistics, pages 181–184.

Self-Adaptation for Unsupervised Domain Adaptation

Xia Cui and **Danushka Bollegala**
Department of Computer Science
University of Liverpool
Ashton Street, Liverpool L69 3BX, United Kingdom
{xia.cui, danushka.bollegala}@liverpool.ac.uk

Abstract

Lack of labelled data in the target domain for training is a common problem in domain adaptation. To overcome this problem, we propose a novel unsupervised domain adaptation method that combines projection and self-training based approaches. Using the labelled data from the source domain, we first learn a projection that maximises the distance among the nearest neighbours with opposite labels in the source domain. Next, we project the source domain labelled data using the learnt projection and train a classifier for the target class prediction. We then use the trained classifier to predict pseudo labels for the target domain unlabelled data. Finally, we learn a projection for the target domain as we did for the source domain using the pseudo-labelled target domain data, where we maximise the distance between nearest neighbours having opposite pseudo labels. Experiments on a standard benchmark dataset for domain adaptation show that the proposed method consistently outperforms numerous baselines and returns competitive results comparable to that of SOTA including self-training, tri-training, and neural adaptations.

1 Introduction

A machine learning model trained using data from one domain (source domain) might not necessarily perform well on a different (target) domain when their distributions are different. Domain adaptation (DA) considers the problem of adapting a machine learning model such as a classifier that is trained using a source domain to a target domain. In particular, in Unsupervised Domain Adaptation (UDA) (Blitzer et al., 2006, 2007; Pan et al., 2010) we do not assume the availability of any labelled instances from the target domain but a set of labelled instances from the source domain and unlabelled instances from both source and the target domains.

Two main approaches for UDA can be identified from prior work: **projection-based** and **self-training**.

Projection[1]-based methods for UDA learn an embedding space where the distribution of features in the source and the target domains become closer to each other than they were in the original feature spaces (Blitzer et al., 2006). For this purpose, the union of the source and target feature spaces is split into domain-independent (often referred to as *pivots*) and domain-specific features using heuristics such as mutual information or frequency of a feature in a domain. A projection is then learnt between those two feature spaces and used to adapt a classifier trained from the source domain labelled data. For example, methods based on different approaches such as graph-decomposition *spectral feature alignment* (Pan et al., 2010) or autoencoders (Louizos et al., 2015) have been proposed for this purpose.

Self-training (Yarowsky, 1995; Abney, 2007) is a technique to iteratively increase a set of labelled instances by training a classifier using current labelled instances and applying the trained classifier to predict pseudo-labels for unlabelled instances. High confident predictions are then appended to the current labelled dataset, thereby increasing the number of labelled instances. The process is iterated until no additional pseudo-labelled instances can be found. Self-training provides a direct solution to the lack of labelled data in the target do-

[1] we consider the terms *project* and *embed* as synonymous in this paper

Proceedings of Recent Advances in Natural Language Processing, pages 213–222,
Varna, Bulgaria, Sep 2–4 2019.

main in UDA (McClosky et al., 2006; Reichart and Rappoport, 2007; Drury et al., 2011). Specifically, the source domain's labelled instances are used to initialise the self-training process and during subsequent iterations labels are inferred for the target domain's unlabelled instances, which can be used to train a classifier for the task of interest.

So far in UDA projection-learning and self-trained approaches have been explored separately. An interesting research question we ask and answer positively in this paper is whether *can we improve the performance of projection-based methods in UDA using self-training?* In particular, recent work on UDA (Morerio et al., 2018) has shown that minimising the entropy of a classifier on its predictions in the source and target domains is equivalent to learning a projection space that maximises the correlation between source and target instances. Motivated by these developments, we propose **Self-Adapt**, a method that combines the complementary strengths of projection-based methods and self-training methods for UDA.

Our proposed method consists of three steps.

- First, using labelled instances from the source domain we learn a projection ($\mathcal{S}_{\text{prj}}$) that maximises the distance between each source domain labelled instance and its nearest neighbours with opposite labels. Intuitively, this process will learn a projected feature space in the source domain where the margin between the opposite labelled nearest neighbours is maximised, thereby minimising the risk of misclassifications. We project the source domain's labelled instances using $\mathcal{S}_{\text{prj}}$ for the purpose of training a classifier for predicting the target task labels such as positive/negative sentiment in cross-domain sentiment classification or part-of-speech tags in cross-domain part-of-speech tagging.
- Second, we use the classifier trained in the previous step to assign pseudo labels for the (unlabelled) target domain instances. Different strategies can be used for this label inference process such as selecting instances with the highest classifier confidence as in self-training or checking the agreement among multiple classifier as in tri-training.
- Third, we use the pseudo-labelled target domain instances to learn a projection for the target domain ($\mathcal{T}_{\text{prj}}$) following the same procedure used to learn $\mathcal{S}_{\text{prj}}$. Specifically, we learn a projected feature space in the target domain where the margin between the opposite pseudo-labelled nearest neighbours is maximised. We project labelled instances in the source domain and pseudo-labelled instances in the target domain respectively using $\mathcal{S}_{\text{prj}}$ and $\mathcal{T}_{\text{prj}}$, and use those projected instances to learn a classifier for the target task.

As an evaluation task, we perform cross-domain sentiment classification on the Amazon multi-domain sentiment dataset (Blitzer et al., 2007). Although most prior work on UDA have used this dataset as a standard evaluation benchmark, the evaluations have been limited to the four domains *books*, *dvds*, *electronic appliances* and *kitchen appliances*. We too report performances on those four domains for the ease of comparison against prior work. However, to reliably estimate the generalisability of the proposed method, we perform an additional extensive evaluation using 16 other domains included in the original version of the Amazon multi-domain sentiment dataset.

Results from the cross-domain sentiment classification reveal several interesting facts. A baseline that uses $\mathcal{S}_{\text{prj}}$ alone would still outperform a baseline that uses a classifier trained using only the source domain's labelled instances on the target domain test instances, without performing any adaptations. This result shows that it is useful to consider the label distribution available in the source domain to learn a projection even though it might be different to that in the target domain.

On the other hand, training a classifier using the pseudo-labelled target domain instances alone, without learning $\mathcal{T}_{\text{prj}}$ further improves performance. This result shows that pseudo labels inferred for the target domain unlabelled instances can be used to overcome the issue of lack of labelled instances in the target domain.

Moreover, if we further use the pseudo-labelled instances to learn $\mathcal{T}_{\text{prj}}$, then we see a significant improvement of performance across all domain pairs, suggesting that UDA can benefit from both projection learning and self-training.

These experimental results support our claim that it is beneficial to combine projection-based and self-training-based UDA approaches. Moreover, our proposed method outperforms all self-training based domain adaptation methods such as tri-training (Zhou and Li, 2005; Søgaard, 2010) and is competitive against neural domain adaptation methods (Louizos et al., 2015; Ganin et al., 2016; Saito et al., 2017; Ruder and Plank, 2018).

2 Related Work

Self-training (Yarowsky, 1995) has been adapted to various cross-domain NLP tasks such as document classification (Raina et al., 2007), POS tagging (McClosky et al., 2006; Reichart and Rappoport, 2007) and sentiment classification (Drury et al., 2011). Although different variants of self-training algorithms have been proposed (Abney, 2007; Yu and Kübler, 2011) a common recipe can be recognised involving the following three steps: (a) Initialise the training dataset, $\mathcal{L} = \mathcal{S}_L$, to the labelled data in the source domain, and train a classifier for the target task using $\mathcal{L}$, (b) apply the classifier trained in step (a) to the unlabelled data in the target domain $\mathcal{T}_U$, and append the most confident predictions as identified by the classifier (e.g. higher than a pre-defined confidence threshold τ) to the labelled dataset $\mathcal{L}$, (c) repeat the two steps (a) and (b) until we cannot append additional high confidence predictions to $\mathcal{L}$.

Another popular approach for inferring labels for the target domain is co-training (Blum and Mitchell, 1998), where the availability of multiple views of the feature space is assumed. In the simplest case where there are two views available for the instances, a separate classifier is trained using the source domain's labelled instances that involve features from a particular view only. Next, the two classifiers are used to predict pseudo labels for the target domain unlabelled instances. If the two classifiers agree on the label for a particular unlabelled instances, then that label is assigned to that instance. Co-training has been applied to UDA (Yu and Kübler, 2011; Chen et al., 2011), where the feature spaces in the source and target domains were considered as the multiple views. The performance of co-training will depend on the complementarity of the information captured by the different feature spaces. Therefore, it is an important to carefully design multiple feature spaces when performing UDA. In contrast, our proposed method does not require such multiple views and does not require training multiple classifiers for the purpose of assigning pseudo labels for the target domain unlabelled instances, which makes the proposed method easy to implement.

Tri-training (Zhou and Li, 2005) relaxes the requirement of co-training for the feature spaces to be sufficient and redundant views. Specifically, in tri-training, as the name implies three separate classifiers are trained using bootstrapped subsets of instances sampled from the labelled instances. If at least two out of the three classifiers agree upon a label for an unlabelled instance, that label is then assigned to the unlabelled instance. Søgaard (2010) proposed a variation of tri-training (i.e. tri-training with diversification) that diversifies the sampling process and reduces the number of additional instances, where they require exactly two out of the three classifiers to agree upon a label and the third classifier to disagree. It has been shown that the classic tri-training algorithm when applied to UDA acts as a strong baseline that outperforms even more complex SoTA neural adaptation methods (Ruder and Plank, 2018). As later shown in our experiments, the proposed **Self-Adapt** method consistently outperforms self-training, tri-training and tri-training with diversification across most of the domain pairs considered.

Projection-based approaches for UDA learn a (possibly lower-dimensional) projection where the difference between the source and target feature spaces is reduced. For example, Structural Correspondence Learning (SCL) (Blitzer et al., 2006, 2007) learns a projection using a set of domain invariant common features called *pivots*. Different strategies have been proposed in the literature for finding pivots for different tasks such as the frequency of a feature in a domain for cross-domain POS tagging (Blitzer et al., 2006; Cui et al., 2017a), mutual information (Blitzer et al., 2007) and pointwise mutual information (Bollegala et al., 2011, 2015) for cross-domain sentiment classification. Cui et al. (2017b) proposed a method for learning the appropriateness of a feature as a pivot (*pivothood*) from the data during training, without requiring any heuristics. Although we use projections in the proposed method, unlike prior work on projection-based UDA, we *do not* require splitting the feature space into domain independent and domain specific features. Moreover, we learn two separate projections for each of the source and target domain, which gives us more flexibility to address the domain-specific constrains in the learnt projections.

Neural adaptation methods have recently reported SoTA for UDA. Louizos et al. (2015) proposed a Variational Fair Autoencoder (VFAE) to learn an invariant representation for a domain.

They used Maximum Mean Discrepancy (MMD) (Gretton et al., 2006) for further promoting the invariant projected feature space. Ganin et al. (2016) proposed Domain Adversarial Neural Network (DANN) to learn features that combine the discriminative power of a classifier and the domain-invariance of the projection space to simultaneously learn adaptable and discriminative projections. Saito et al. (2017) proposed a deep tri-training method with three neural networks, two for pseudo labelling the target unlabelled data and another one for learning discriminator using the inferred pseudo labels for the target domain. Ruder and Plank (2018) proposed Multi-task Tri-training (MT-Tri) based on tri-training and Bi-LSTM. They show that tri-training is a competitive baseline and rivals more complex neural adaptation methods. Although MT-Tri does not outperform SoTA on cross-domain sentiment classification tasks, their proposal reduces the time and space complexity required by the classical tri-training.

As stated above, our proposed method **Self-Adapt**, differs from the prior work discussed above in that it (a) does not require pivots, (b) does not require multiple feature views, (c) learns two different projections for the source and target domains and (d) combines a projection and a self-training step in a non-iterative manner to improve the performance in UDA.

3 Self-Adaptation (Self-Adapt)

In UDA, we are given a set of positively ($\mathcal{S}_L^+$) and negatively ($\mathcal{S}_L^-$) labelled instances for a source domain $\mathcal{S}$ ($\mathcal{S}_L = \mathcal{S}_L^+ \cup \mathcal{S}_L^-$), and sets of unlabelled instances $\mathcal{S}_U$ and $\mathcal{T}_U$ respectively for the source and target domain $\mathcal{T}$. Given a dataset $\mathcal{D}$, we are required to learn a classifier $f(\boldsymbol{x}, y; \mathcal{D})$ that returns the probability of a test instance $\boldsymbol{x}$ taking a label y. For simplicity, we consider the pairwise adaptation from a single source to single target, and binary ($y \in \{-1, 1\}$) classification as the target task. However, self-adapt can be easily extended to multi-domain and multi-class UDA.

We represent an instance (document/review) x by a bag-of-n-gram (BonG) embedding (Arora et al., 2018), where we add the pre-trained d-dimensional word embeddings $\boldsymbol{w} \in \mathbb{R}^d$ for the words $w \in x$ to create a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$ representing x. Self-adapt consists of three steps: (a) learning a source projection using $\mathcal{S}_L$ (Section 3.1), (b) pseudo labelling $\mathcal{T}_U$ using a classifier trained on the projected $\mathcal{S}_L$ (Section 3.2); (c) learning a target projection using the pseudo-labelled target instances, and then learning a classifier f for the target task (Section 3.3).

3.1 Source Projection Learning ($\mathcal{S}_{\text{prj}}$)

In UDA, the adaptation task does not vary between the source and target domains. Therefore, we can use $\mathcal{S}_L$ to learn a projection for the source domain $\mathcal{S}_{\text{prj}}$ where the separation between an instance $x \in \mathcal{S}_L$ and its opposite-labelled nearest neighbours is maximised. Specifically, for an instance x we represent the set of k of its nearest neighbours $\text{NN}(\boldsymbol{x}, \mathcal{D}, k)$ selected from a set $\mathcal{D}$ by a vector $\phi(\boldsymbol{x}, \mathcal{D}, k)$ as the sum of the word embeddings of the neighbours given by (1).

$$\phi(\boldsymbol{x}, \mathcal{D}, k) = \sum_{u \in \text{NN}(\boldsymbol{x}, \mathcal{D}, k)} \theta(x, u)\boldsymbol{u}. \qquad (1)$$

Here, the weight $\theta(x, u)$ is computed using the cosine similarity between $\boldsymbol{u}$ and $\boldsymbol{x}$, and is normalised s.t. $\sum_{u \in \text{NN}(\boldsymbol{x}, \mathcal{D}, k)} \theta(x, u) = 1$. Other similarity measures can also be used instead of cosine, for example, Euclidean distance (Van Asch and Daelemans, 2016). Then, $\mathcal{S}_{\text{prj}}$ is defined by the projection matrices $\mathbf{A}_+ \in \mathbb{R}^{d \times d}$ and $\mathbf{A}_- \in \mathbb{R}^{d \times d}$ and is learnt by maximising the objective O_L given by (2).

$$\begin{aligned} O_L(\mathbf{A}_+, \mathbf{A}_-) = & \sum_{\boldsymbol{x} \in \mathcal{S}_L^+} \left|\left|\mathbf{A}_+\boldsymbol{x} - \mathbf{A}_-\phi(\boldsymbol{x}, \mathcal{S}_L^-, k)\right|\right|_2^2 \\ & + \sum_{\boldsymbol{x} \in \mathcal{S}_L^-} \left|\left|\mathbf{A}_-\boldsymbol{x} - \mathbf{A}_+\phi(\boldsymbol{x}, \mathcal{S}_L^+, k)\right|\right|_2^2 \qquad (2) \end{aligned}$$

We initialise $\mathbf{A}_+$ and $\mathbf{A}_-$ to the identify matrix $\mathbf{I} \in \mathbb{R}^{d \times d}$ and apply Adam (Kingma and Ba, 2014) to find their optimal values denoted respectively by $\mathbf{A}_+^*$ and $\mathbf{A}_-^*$. Finally, we project $\mathcal{S}_L$ using the learnt $\mathcal{S}_{\text{prj}}$ to obtain a projected set of source domain labelled instances $\mathcal{S}_L^* = \mathbf{A}_+^* \circ \mathcal{S}_L^+ \cup \mathbf{A}_-^* \circ \mathcal{S}_L^-$. Here, we use the notation $\mathbf{A} \circ \mathcal{D} = \{\mathbf{A}\boldsymbol{x} | \boldsymbol{x} \in \mathcal{D}\}$ to indicate the application of a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ on elements $\boldsymbol{x} \in \mathbb{R}^d$ in a dataset $\mathcal{D}$.

3.2 Pseudo Label Generation (PL)

In UDA, we do not have labelled data for the target domain. To overcome this issue, inspired by prior work on self-training approaches to UDA, we train a classifier $f(\boldsymbol{x}, y; \mathcal{S}_L^*)$ on $\mathcal{S}_L^*$ first and then use this classifier to assign pseudo labels for the target domain's unlabelled data $\mathcal{T}_U$, if the classifier

Algorithm 1 Pseudo Label Generation

```
Input: source domain positively labelled data $\mathcal{S}_L^+$,
       source domain negatively labelled data $\mathcal{S}_L^-$,
       source domain positive transformation matrix $\mathbf{A}_+$,
       source domain negative transformation matrix $\mathbf{A}_-$,
       target domain unlabelled data $\mathcal{T}_U$,
       a set of target classes $Y = \{+1, -1\}$,
       classification confidence threshold $\tau$.
Output: target domain pseudo-labelled data $\mathcal{T}'_L$
  $\mathcal{S}_L^* \leftarrow \mathbf{A}_+^* \mathcal{S}_L^+ \cup \mathbf{A}_-^* \mathcal{S}_L^-$
  $\mathcal{T}'_L \leftarrow \emptyset$
  for $\boldsymbol{x} \in \mathcal{T}_U$ do
    $y \in Y$, $p(t = y|\boldsymbol{x}) = f(\boldsymbol{x}, y; \mathcal{S}_L^*)$ {probability of $\boldsymbol{x}$ belongs to class $y$}
    if $p(t = y|\boldsymbol{x})) > \tau$ then
      $\mathcal{T}'_L \leftarrow \mathcal{T}'_L \cup \{(\boldsymbol{x}, y)\}$
    end if
  end for
  return $\mathcal{T}'_L$
```

is more confident than a pre-defined threshold τ. Algorithm 1 returns a pseudo-labelled dataset $\mathcal{T}'_L$ for the target domain. According to the classical self-training (Yarowsky, 1995; Abney, 2007), $\mathcal{T}'_L$ will be appended to $\mathcal{S}_L^*$ and the classifier is retrained on this extended labelled dataset. The process is repeated until no further unlabelled instances can be assigned labels with confidence higher than τ. However, in our preliminary experiments, we found that this process does not improve the performance in UDA beyond the first iteration. Therefore, we limit the number of iterations to one as shown in Algorithm 1. Doing so also speeds up the training process over classical self-training, which retrains the classifier and iterates.

3.3 Target Projection Learning ($\mathcal{T}_{\text{prj}}$)

Armed with the pseudo-labelled data generated via Algorithm 1, we can now learn a projection for the target domain, $\mathcal{T}_{\text{prj}}$, following the same procedure we proposed for learning $\mathcal{S}_{\text{prj}}$ in Section 3.1. Specifically, $\mathcal{T}_{\text{prj}}$ is defined by the two target-domain projection matrices $\mathbf{B}_+ \in \mathbb{R}^{d \times d}$ and $\mathbf{B}_- \in \mathbb{R}^{d \times d}$ that maximises the distance between each pseudo-labelled target instance x and its k opposite labelled nearest neighbours selected from positively ($\mathcal{T}_L'^+$) and negatively ($\mathcal{T}_L'^-$) pseudo-labelled instances. The objective O'_L for this optimisation problem is given by (3).

$$O'_L(\mathbf{B}_+, \mathbf{B}_-) = \sum_{\boldsymbol{x} \in \mathcal{T}_L'^+} \left|\left|\mathbf{B}_+\boldsymbol{x} - \mathbf{B}_-\phi(\boldsymbol{x}, \mathcal{T}_L'^-, k)\right|\right|_2^2 + \sum_{\boldsymbol{x} \in \mathcal{T}_L'^-} \left|\left|\mathbf{B}_-\boldsymbol{x} - \mathbf{B}_+\phi(\boldsymbol{x}, \mathcal{T}_L'^+, k)\right|\right|_2^2 \quad (3)$$

Likewise with $\mathcal{S}_{\text{prj}}$, $\mathbf{B}_+$ and $\mathbf{B}_-$ are initialised to the identify matrix $\mathbf{I} \in \mathbb{R}^{d \times d}$, and Adam is used to find their minimisers denoted respectively by $\mathbf{B}_+^*$ and $\mathbf{B}_-^*$. We project the target domain pseudo-labelled data using $\mathcal{T}_{\text{prj}}$ to obtain $\mathcal{T}_L'^* = \mathbf{B}_+^* \circ \mathcal{T}_L'^+ \cup \mathbf{B}_-^* \circ \mathcal{T}_L'^-$. Finally, we train a classifier $f(\boldsymbol{x}, y; \mathcal{S}_L^* \cup \mathcal{T}_L'^*)$ for the target task using both source and target projected labelled instances $\mathcal{S}_L^* \cup \mathcal{T}_L'^*$. Any binary classifier can be used for this purpose. In our experiments, we use ℓ_2 regularised logistic regression following prior work in UDA (Blitzer et al., 2006; Pan et al., 2010; Bollegala et al., 2013). Moreover, by using a simple linear classifier, we can decouple the projection learning step from the target classification task, thereby more directly evaluate the performance of the former.

4 Experiments

Our proposed method *does not* assume any information about the target task and can be in principle applied for any domain adaptation task. We use cross-domain sentiment classification as an evaluation task in this paper because it has been used extensively in prior work on UDA, thereby enabling us to directly compare the performance of our proposed method against previously proposed UDA methods. In particular, we use the Amazon multi-domain sentiment dataset, originally created by Blitzer et al. (2007), as a benchmark dataset in our experiments. This dataset includes Amazon Product Reviews from four categories: Books (**B**), DVDs (**D**), Electronic Appliances (**E**) and Kitchen Appliances (**K**). Considering each category as a domain[2], we can generate $\binom{4}{2} = 12$ pair-wise adaptation tasks involving a single source and a single target domain.

An Amazon product review is assigned 1-5 star rating and product reviews with 4 or 5 stars are labelled as positive, whereas 1 or 2 star reviews

[2] Multiple reviews might exist for the same product within the same domain. Products are not shared across domains.

are labelled as negative. 3 star reviews are ignored because of their ambiguity. In addition to the labelled reviews, the Amazon multi-domain dataset contains a large number of unlabelled reviews for each domain. We use the official balanced train and test dataset splits, which has 800 (pos), 800 (neg) training instances and 200 (pos), 200 (neg) test instances for each domain. We name this dataset as the Multi-domain Adaptation Dataset (**MAD**).

One issue that is often raised with **MAD** is that it contains only four domains. In order to robustly evaluate the performance of an UDA method we must evaluate on multiple domains. Therefore, in addition to **MAD**, we also evaluate on an extended dataset that contains 16 domains. We name this dataset as the Extended Multi-domain Adaptation Dataset (**EMAD**). The reviews for the 16 domains contained in **EMAD** were also collected by Blitzer et al. (2007), but were not used in the evaluations. The same star-based procedure used in **MAD** is used to label the reviews in **EMAD**. We randomly select 20% of the available labelled reviews as test data and construct a balanced training dataset from the rest of the labelled reviews (i.e. for each domain we have equal number of positive and negative labelled instances in the train datasets). Likewise in **MAD**, we generate $\binom{16}{2} = 240$ pair-wise domain adaptation tasks from **EMAD**.

We train an ℓ_2 regularised logistic regression as the target (sentiment) classifier, in which we tune the regularisation coefficient using validation data. We randomly select 10% from the target domain labelled data, which is separate from the train or test data. We tune regularisation coefficient in $[0.001, 0.01, 0.1, 1]$. We use 300 dimensional pre-trained GloVe word embeddings (Pennington et al., 2014) to create BonG embeddings for uni and bigrams. We found that a maximum of 100 epochs to be sufficient to reach convergence in all projection learning tasks for all domains. The source code implementation of **self-adapt** will be made publicly available upon paper acceptance.

4.1 Effect of Projection Learning and Pseudo-Labelling

Our proposed method consists of 3 main steps as described in Section 3: source projection learning, pseudo labelling and target projection learning. Using **MAD**, in Table 1, we compare the relative effectiveness of these three steps towards the overall performance in UDA using $k = 1$ for all the steps. Specifically, we consider the following baselines.

NA: No-adaptation. Learn a classifier from $\mathcal{S}_L$ and simply use it to classify sentiment on target domain test instances, without performing any domain adaptation.

$\mathcal{S}_{\text{prj}}$: Learn a source projection $\mathcal{S}_{\text{prj}}$ and apply it to project $\mathcal{S}_L$ to obtain $\mathcal{S}_L^* = \mathbf{A}_+^* \circ \mathcal{S}_L^+ \cup \mathbf{A}_-^* \circ \mathcal{S}_L^-$. Train a sentiment classifier using $\mathcal{S}_L^*$ and use it to classify target domain test instances.

$\mathcal{S}_{\text{prj}}$ +PL: Use the classifier trained using $\mathcal{S}_L^*$ on target domain unlabelled data to create a pseudo-labelled dataset $\mathcal{T}'_L$. Train a sentiment classifier on $\mathcal{S}_L^* \cup \mathcal{T}'_L$ and use it to classify target domain test instances.

$\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$: This is the proposed method including all three steps. A target projection $\mathcal{T}_{\text{prj}}$ is learnt using $\mathcal{T}'_L$ and is applied to obtain $\mathcal{T}'^*_L = \mathbf{B}_+^* \circ \mathcal{T}'^+_L \cup \mathbf{B}_-^* \circ \mathcal{T}'^-_L$. Finally, a sentiment classifier is trained using $\mathcal{S}_L^* \cup \mathcal{T}'^*_L$ and used to classify target domain test instances.

With all methods, we keep $k = 1$ in the nearest neighbour feature representation in (1) for the ease of comparisons. Confidence threshold τ is tuned in the range $[0.5, 0.9]$ using cross-validation. From Table 1 we see that $\mathcal{S}_{\text{prj}}$ consistently outperforms **NA**, showing that even without using any information from the target domain, it is still useful to learn source domain projections that discriminates instances with opposite labels. When we perform pseudo labelling on top of source projection learning ($\mathcal{S}_{\text{prj}}$ **+PL**) we see a slight but consistent improvement in all domain-pairs. However, when we use the pseudo labelled instances to learn a target projection ($\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$) we obtain the best performance in all domain-pairs. Moreover, the obtained results are significantly better under the stricter $p < 0.001$ level over the **NA** baseline in 7 out of 12 domain-pairs.

Table 3 shows the classification accuracy for the **EMAD**. Due to the limited availability of space, we show the average classification accuracy for adapting to the same target domain instead of showing all 240 domain-pairs for **EMAD**. Likewise in **MAD**, we see in **EMAD** we obtain the best results when we use both source and target projections. Interestingly, we see that the proposed method adapting well even to the domains

$\mathcal{S}-\mathcal{T}$	**NA**	$\mathcal{S}_{\textbf{prj}}$	$\mathcal{S}_{\textbf{prj}}$ **+PL**	$\mathcal{S}_{\textbf{prj}}$ **+**$\mathcal{T}_{\textbf{prj}}$
B-D	73.50	74.25	74.50	**76.25**
B-E	64.00	73.25**	73.50**	**77.50****
B-K	68.50	75.25*	76.00*	**78.75****
D-B	74.00	**75.50**	**75.50**	**75.50**
D-E	64.75	71.25*	71.50*	**74.25****
D-K	71.50	75.00	76.25	**79.75****
E-B	67.25	74.50*	75.25**	**76.25****
E-D	66.75	67.75	68.00	**69.50**
E-K	76.00	**81.00**	**81.00**	**81.00**
K-B	63.00	73.75**	73.75**	**75.00****
K-D	71.25	71.25	71.00	**72.50**
K-E	69.00	77.50**	78.00**	**78.25****

Table 1: Target domain test data classification accuracy for the different steps in the proposed method ($k = 1$). $\mathcal{S} - \mathcal{T}$ denotes adapting from a source $\mathcal{S}$ to a target $\mathcal{T}$ domain. The best result for each domain-pair is bolded. Statistically significant improvements over **NA** according to the binomial exact test are shown by "*" and "**" respectively at $p = 0.01$ and $p = 0.001$ levels.

with smaller numbers of unlabelled data such as **gourmet_food** (168 labelled and 267 unlabelled train instances). This is encouraging because it shows that the proposed method can overcome the lack of labelled instances via pseudo labelling and projection learning.

4.2 Comparisons against Self-Training

Ruder and Plank (2018) evaluated classical general-purpose semi-supervised learning methods proposed for inducing pseudo labels for unlabelled instances using a seed set of labelled instances in the context of UDA. They found that tri-training to outperform more complex neural SoTA UDA methods. Considering the fact that **Self-Adapt** is performing pseudo-labelling, similar to other self-training methods, it is interesting to see how well it compares against classical self-training methods for inducing labels (Yarowsky, 1995; Abney, 2007; Zhou and Li, 2005; Søgaard, 2010) when applied to UDA. Specifically, we consider the classical self-training (Yarowsky, 1995; Abney, 2007) (**Self**), Tri-training (Zhou and Li, 2005) (**Tri**) and Tri-training with diversification (Søgaard, 2010) (**Tri-D**). For each of those methods, we use the labelled data in the source domain as seeds and induce labels for the unlabelled data in the target domain. Table 2 reports the results on **MAD**.

We re-implement the classical self-training methods considered by Ruder and Plank (2018) and evaluated them against the proposed self-adapt on the same datasets, feature representations and settings to conduct a fair comparison. All classical self-training methods were trained using the source domain labelled instances $\mathcal{S}_L$ as seed data. As discussed in Section 3.2, similar to **Self-Adapt**, we observed that the performance did not significantly increase beyond the first iteration for any of the classical self-training methods in UDA. Consequently, we compare all classical self-training methods for their peak performance, obtained after the first iteration. We tune the confidence threshold τ for each method using validation data and found the optimal value of τ to fall in the range $[0.6, 0.9]$. k is a hyperparameter selected using validation dataset for **Self-Adapt** in comparison.

Experimental results on **MAD** and **EMAD** are shown respectively in Tables 2 and 4. From those Tables, we see that **Self-Adapt** for most of the domain pairs performs similarly or slightly worse than **NA**. Although **Tri** and **Tri-D** outperform **NA** on all cases, we found that those two methods are highly sensitive to the seed instances used to initialise the pseudo-labelling process. We find the proposed **Self-Adapt** to outperform all classical self-training based methods in 11 out of 12 domain pairs in **MAD** and in all 16 target domains in **EMAD**, showing a strong and robust improvement over classical self-training methods. This result shows that by combining source and target domain projections with self-training, we can obtain superior performance in UDA in comparison to using classical self-training methods alone.

S-T	**NA**	**Self**	**Tri**	**Tri-D**	**Self-Adapt**
B-D	73.50	74.25	74.75	**77.25**	**77.25**
B-E	64.00	65.00	73.25**	72.00**	**78.50****
B-K	68.50	70.75	73.25	73.75	**79.00****
D-B	74.00	73.00	77.00	76.00	**81.00***
D-E	64.75	65.25	73.75**	70.50*	**75.50****
D-K	71.50	71.75	75.75	74.00	**79.75****
E-B	67.25	68.25	68.50	74.25*	**76.25****
E-D	66.75	66.25	68.25	**73.50***	69.50
E-K	76.00	76.50	**82.50***	81.00	**82.50***
K-B	63.00	63.00	70.00*	73.00**	**75.00****
K-D	71.25	71.00	71.25	72.00	**72.75**
K-E	69.00	68.75	73.00	75.50*	**79.00****

Table 2: Target domain test data classification accuracy of classical self-training methods when applied to UDA.

4.3 Comparisons against Neural UDA

In Table 5, we compare **Self-Adapt** against the following neural UDA methods on **MAD**: Variational Fair Autoencoder (Louizos et al., 2015) (**VFAE**), Domain-adversarial Neural Networks (Ganin et al., 2016) (**DANN**), Asymmet-

Target Domain	NA	$\mathcal{S}_{\text{prj}}$	$\mathcal{S}_{\text{prj}}$ +PL	$\mathcal{S}_{\text{prj}}$ +$\mathcal{T}_{\text{prj}}$
apparel	71.39	75.31	75.66	**76.68***
baby	68.00	71.06	71.37	**72.63**
beauty	69.66	73.20	73.18	**73.70**
camera_and_photo	68.46	73.46	74.02	**74.54***
computer_and_video_games	61.78	67.38	67.85	**68.11***
gourmet_food	72.34	81.13**	82.89**	**83.15****
grocery	71.01	76.90*	77.57 *	**78.14***
health_and_personal_care	65.85	68.38	68.67	**69.24**
jewelry_and_watches	68.90	77.86**	79.11**	**79.79****
magazines	65.28	71.54*	71.45*	**72.16***
music	66.27	70.69	70.89	**71.41**
outdoor_living	71.97	76.77	78.01*	**78.93***
software	65.72	69.56	69.60	**70.31**
sports_and_outdoors	66.56	70.18	70.67	**71.02**
toys_and_games	69.57	72.82	73.22	**73.65**
video	64.19	67.59	68.37	**68.97**

Table 3: Average classification accuracy on each target domain in **EMAD** for the steps in the proposed method ($k = 1$).

ric Tri-training (Saito et al., 2017) (**Asy-Tri**), and Multi-task Tri-training (Ruder and Plank, 2018) (**MT-Tri**). We select these methods as they are the current SoTA for UDA on **MAD**, and report the results from the original publications in Table 5.

Although only in 3 out of 12 domain-pairs **Self-Adapt** is obtaining the best performance, the difference of performance between **DANN** and **Self-Adapt** is not statistically significant. Although **MT-Tri** is outperforming **Self-Adapt** in 8 domain-pairs, it is noteworthy that **MT-Tri** is using a larger feature space than that of **Self-Adapt**. Specifically, **MT-Tri** is using 5000 dimensional tf-idf weighted unigram and bigram vectors for representing reviews, whereas we **Self-Adapt** uses a 300 dimensional BonG representation computed using pre-trained GloVe vectors. Moreover, prior work on neural UDA have not used the entire unlabelled datasets and have sampled a smaller subset due to computational feasibility. For example, **MT-Tri** uses only 2000 unlabelled instances for each domain despite the fact that the original unlabelled datasets contain much larger numbers of reviews. This is not only a waste of available data but it is also non-obvious as how to subsample unlabelled data for training. Our preliminary experiments revealed that the performance of neural UDA methods to be sensitive to the unlabelled datasets used.[3] On the other hand, **Self-Adapt** does not require sub-sampling of unlabelled data and uses all the available unlabelled data for UDA. During the pseudo-labelling step, **Self-Adapt** automatically selects a subset of unlabelled target instances that are determined to be confident by the classifier more than a pre-defined threshold τ. The ability to operate on a lower-dimensional feature space and obviating the need to subsample unlabelled data are properties of the proposed method that are attractive when applying UDA methods on large datasets and across multiple domains.

Target Domain	NA	Self	Tri	Tri-D	Self-Adapt
apparel	71.39	71.38	73.96	73.89	**76.68**
baby	68.00	67.96	69.71	69.84	**72.63**
beauty	69.66	70.54	71.73	70.49	**73.70**
camera_and_photo	68.46	68.44	71.31	71.28	**74.54***
computer_and_video_games	61.78	62.28	64.65	64.93	**68.11***
gourmet_food	72.34	74.10	75.39	77.88	**83.15****
grocery	71.01	71.83	75.22	74.29	**78.14***
health_and_personal_care	65.85	66.08	67.10	67.07	**69.24**
jewelry_and_watches	68.90	71.04	76.67**	76.21**	**79.79****
magazines	65.28	65.34	67.58	67.47	**72.16***
music	66.27	66.22	68.82	68.27	**71.41**
outdoor_living	71.97	74.01	76.21	74.79	**78.93***
software	65.72	65.47	67.36	67.53	**70.31**
sports_and_outdoors	66.56	66.87	69.22	68.05	**71.02**
toys_and_games	69.57	70.03	71.76	72.32	**73.65**
video	64.19	64.88	66.34	67.49	**68.97**

Table 4: Average classification accuracy on each target domain in **EMAD** of classical self-training based methods when applied to UDA.

S-T	VFAE	DANN	Asy-Tri	MT-Tri	Self-Adapt
B-D	79.90	78.40	80.70	**81.47**	77.25
B-E	79.20	73.30	**79.80**	78.62	78.50
B-K	81.60	77.90	**82.50**	78.09	79.00
D-B	75.50	72.30	73.20	77.49	**81.00****
D-E	78.60	75.40	77.00	**79.66**	75.50
D-K	82.20	78.30	**82.50**	81.23	79.75
E-B	72.70	71.10	73.20	73.43	**76.25**
E-D	**76.50**	73.80	72.90	75.05	69.50
E-K	85.00	85.40	86.90	**87.07**	82.50
K-B	72.00	70.90	72.50	73.60	**75.00**
K-D	73.30	74.00	74.90	**77.41**	72.75
K-E	83.80	84.30	84.60	**86.06**	79.00

Table 5: Classification accuracy compared with neural adaptation methods. The best result is bolded. Statistically significant improvements over **DANN** according to the binomial exact test are shown by "*" and "**" respectively at $p = 0.01$ and $p = 0.001$ levels.

5 Conclusions

We proposed **Self-Adapt**, an UDA method that combines projection learning and self-training. Our experimental results on two datasets for cross-domain sentiment classification show that projection learning and self-training have complementary strengths and jointly contribute to improve UDA performance. In future, we plan to apply **Self-Adapt** to other UDA tasks in NLP such as cross-domain POS tagging and NER.

[3] Unfortunately, the source code for MT-Tri was not available for us to run this method with the same set of features and unlabelled dataset that we used. Therefore, we report the results from the original publication.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*, 1st edition. Chapman & Hall/CRC.

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and LSTMs. In *International Conference on Learning Representations*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*, pages 440–447.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proc. of EMNLP*, pages 120–128.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Danushka Bollegala, Tingting Mu, and Yannis Goulermas. 2015. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 28(2):398–410.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL/HLT'11*, pages 132 – 141.

Danushka Bollegala, David Weir, and John Carroll. 2013. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1719 – 1731.

Minmim Chen, Kilian Q. Weinberger, and John Blitzer. 2011. Co-training for domain adaptation. In *NIPS'11*.

Xia Cui, Frans Coenen, and Danushka Bollegala. 2017a. Effect of data imbalance on unsupervised domain adaptation of part-of-speech tagging and pivot selection strategies. In *Proc. of the Wokshop on Learning With Imbalanced Domains: Theory and Applications (LIDTA) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 103–115.

Xia Cui, Frans Coenen, and Danushka Bollegala. 2017b. TSP: Learning task-specific pivots for unsupervised domain adaptation. In *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 754–771.

Brett Drury, Luís Torgo, and Jose Joao Almeida. 2011. Guided self training for sentiment classification. In *Proceedings of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pages 9–16.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. 2006. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. 2015. The variational fair autoencoder. *CoRR*, abs/1511.00830.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344. Association for Computational Linguistics.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. 2018. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In *International Conference on Learning Representations*.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proc. of WWW*, pages 751–760.

Jeffery Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *ICML'07*.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL 2007*, pages 616 – 623.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. pages 1044–1054.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 205–208, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vincent Van Asch and Walter Daelemans. 2016. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ning Yu and Sandra Kübler. 2011. Filling the gap: Semi-supervised learning for opinion detection across domains. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 200–209. Association for Computational Linguistics.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

SPECULATION and NEGATION DETECTION in FRENCH BIOMEDICAL CORPORA

Clément Dalloux and **Vincent Claveau**
Univ Rennes, Inria, CNRS, IRISA, F-35000 Rennes, France
name.surname@irisa.fr

Natalia Grabar
UMR 8163 STL CNRS, Université de Lille, France
natalia.grabar@univ-lille.fr

Abstract

In this work, we propose to address the detection of negation and speculation, and of their scope, in French biomedical documents. It has been indeed observed that they play an important role and provide crucial clues for other NLP applications. Our methods are based on CRFs and BiLSTM. We reach up to 97.21 % and 91.30 % F-measure for the detection of negation and speculation cues, respectively, using CRFs. For the computing of scope, we reach up to 90.81 % and 86.73 % F-measure on negation and speculation, respectively, using BiLSTM-CRF fed with word embeddings.

1 INTRODUCTION

The detection of speculation and negation in texts has become one of the unavoidable pre-requisites in many information extraction tasks. Both are common in language and provide information on factuality and polarity of facts, which is particularly important for the biomedical field (Elkin et al., 2005b; Denny and Peterson, 2007; Gindl et al., 2008; Chapman et al., 2001). Indeed, negation and speculation provide there crucial information for detecting patient's present, speculated or absent pathologies and co-morbidities, detecting whether a particular medication has been, may have been, or has not been prescribed or taken, defining the certainty of diagnosis, etc. In order to efficiently identify speculation and negation instances, it is first necessary to identify their cues, i.e., words (or morphological units) that express speculation and negation, and then their scopes, i.e., tokens within the sentence which are affected by the negation or speculation. In this paper, we present two French datasets annotated with negation and speculation cues and their scope. We also propose machine learning and deep learning based systems to tackle the automatic detection of cues and scopes that achieve high performance.

2 EXPRESSION of NEGATION and SPECULATION

In French, the expression of negation and speculation have some specifics that are described below.

2.1 NEGATION

The negation cues may consist of one word or prefix, or of multiple words. Moreover, negation can be expressed via a large panel of cues which can be morphological (*an, in, im, ir, dis*), lexical (*absence de*(absence of), *à l'exception de*(excepting)), and grammatical (*non, ne...pas, ni...ni*). In the following examples, we present sentences with instances of negation (the cues are underlined).

1. *En alternative des traitements locaux (chirurgie, radiothérapie, radiofréquence, cryoablation) peuvent être indiqués mais ils ne sont pas* ***toujours*** *faisables.* *(Alternatively, local treatments (surgery, radiotherapy, radiofrequency, cryoablation) may be indicated but are not* ***always*** *feasible.)*

2. *Il n'existe* ***toujours*** *pas aujourd'hui de consensus quant à une définition précise de ce phénomène hétérogène ou des modalités de sa prise en charge.* *(There is* ***still*** *no consensus today on a precise definition of this heterogeneous phenomenon or the modalities of its management.)*

3. *il n'y a pas de traitement curateur de la maladie* ***en dehors de*** *l'allogreffe de moelle.* *(there is no curative treatment for this disease* ***apart from*** *bone-marrow homograft)*

4. *Lymphome non hodgkinien à cellules B matures récidivant/réfractaire.* *(Relapsed/refractory mature B-cell non-Hodgkin's lymphoma.)*

Proceedings of Recent Advances in Natural Language Processing, pages 223–232,
Varna, Bulgaria, Sep 2–4 2019.

5. *Elle n'est soulagée que par la marche et doit donc écouter la télévision en faisant les cent* ***pas*** *dans son salon. (She is only relieved by walking and must therefore listen to the TV pacing in her living room.)*

Examples (1-2) show the possible effect of frequency adverbs, here *toujours (always)*. In the first sentence *traitements locaux (chirurgie, radiothérapie, radiofréquence, cryoablation)*, the content would be negated without *toujours(always)*. In the second sentence, with or without *toujours*, the meaning of the sentence does not change, therefore, the scope of the negation remains the same.

Example (3) shows how the preposition *en dehors de(apart from)*, can stop the scope of negation. Many other prepositions such as *à part*, *à l'exception de* or *excepté*, with more or less the same meaning than *en dehors de (apart from)*, would have the same effect on the negation scope. However, these prepositions can also play the role of negation by themselves.

Examples (4) show that cues can also be included in medical concepts such as ***non*** *hodgkinien (non-Hodgkin's)*. In our work, we chose to label *hodgkinien* as part of the negation scope.

Finally, example (5) shows the context in which the ambiguous word *pas*, meaning both *no/not* and *footstep*, has the non-negation meaning. In this example, ***pas*** is part of the idiomatic expression ***faire les cent pas*** *(pacing, walking around)*. Another ambiguity is related to the adverb ***plus*** meaning either ***more*** or, in conjunction with ***ne***, ***no more***.

2.2 SPECULATION

The expression of speculation can be even more complex than negation. Indeed, speculation can be triggered by many specific sequences of words. We describe several of them below.

1. *En effet, l'arrêt du traitement antituberculeux en soi pourrait* ***permettre un rétablissement de la fonction normale des héptocytes*** *en éliminant la source de l'atteinte hépatique. (Indeed, stopping TB treatment per se could restore normal hepatocyte function by eliminating the source of liver damage.)*

2. ***Le bénéfice de l'association lénalomide + R-CHOP au rapport par rapport au R-CHOP*** *reste à démontrer. (The benefit of the combination of lenalomide + R-CHOP compared with R-CHOP remains to be demonstrated.)*

3. *Elle aurait* ***eu la pose à 2 reprises d'un dispositif intrautérin*** *(DIU). (She would have had the pose of an intrauterine device (IUD) twice.)*

Example (1) shows a typical occurrence of *pourrait (could)*. As in English, where *can, could, may, etc.* express speculation, in French, *pouvoir* can occur in many forms.

Example (2) shows the effect of *reste à (remains to)* combined with an infinitive verb, here *démontrer*. Other infinitive verbs, such as *expliquer (to explain)* or *vérifier (to verify)*, associated with *reste à* trigger speculation.

Example (3) shows how the conditional tense triggers speculation. In English, *would* triggers speculation in this case, which is simpler to detect. Indeed, in French, the conditional tense is expressed via suffixes (*-ais, -ais, -ait, -ions, -iez, -aient*), which makes the detection harder, especially for supervised learning techniques.

3 RELATED WORK

We present several corpora and methods that have been proposed in the existing work to tackle the tasks of speculation and negation detection.

3.1 DATA

In the recent years, several specialized corpora in English have been annotated with speculation and negation, which has resulted in models for their automatic detection. These corpora can be divided into two categories: (1) corpora annotated with cues and scopes, such as Bioscope (Vincze et al., 2008) or **SEM-2012*, and (2) corpora focusing on concepts and named entities, such as I2B2 and Mipacq. We briefly describe these corpora. The Bioscope corpus (Vincze et al., 2008) contains reports of radiological examinations, scientific articles, and abstracts from biomedical articles. Each sentence and each negation and speculation cue/scope pair receives unique identifier. The *SEM-2012 corpus (Morante and Blanco, 2012) consists of a Sherlock Holmes novel and three other short stories written by Sir Arthur Conan Doyle. It contains 5,520 sentences, among which 1,227 sentences are negated. Each occurrence of the negation, the cue and its scope are annotated, as well as the focus of the negation if relevant. In this corpus, cues and scopes can be discontinuous. The I2B2/VA-2010 challenge (Uzuner et al., 2011) featured several tasks using US clinical records. One task aimed the detection of statements and of their scope. Medical concepts had to be associated with the corresponding statement: *present*, *absent*, *possible*, *conditional*, *hypothetical* or *not associated with the patient*. Mipacq (Albright et al., 2013) is another corpus with clinical data in English annotated with syntactic and semantic labels. Each detected UMLS entity has two attribute locations: *negation* (*true* or *false*) and *status* (*none*, *possible*, *HistoryOf* or *FamilyHistoryOf*).

3.2 EXPERT SYSTEMS

Among the rule based systems dedicated to the negation detection, *NegEx* (Chapman et al., 2001) pioneered the area. It uses regular expressions to detect the cues and to identify medical terms in their scope. It was later adapted to various languages including French (Deléger and Grouin, 2012). *ConText* (Harkema et al., 2009), derived from *NegEx*, covers more objectives: negation, temporality, and the subject concerned by this information in the clinical texts. It has been adapted to French (Abdaoui et al., 2017). In another work, medical concepts may receive additional labels (positive, negative or uncertain) Elkin et al. (2005a). Özgür and Radev (2009); Øvrelid et al. (2010); Kilicoglu and Bergler (2010) exploit lexical, grammatical and syntactic information to detect speculation and its scope. ScopeFinder (Apostolova et al., 2011) detects the scope of negation and speculation with rules built automatically from BioScope (lexico-syntactic patterns extraction). *NegBio* (Peng et al., 2018) detects both negation and speculation in radiology reports with rules based on universal dependency graphs.

3.3 SUPERVISED LEARNING

To our knowledge, Light et al. (2004) is the first work to include supervised learning for speculation detection. It relies on SVM to select speculative sentences in MEDLINE abstracts. Tang et al. (2010) proposes a cascade method based on CRF and SVM classifiers to detect speculation cues and another CRF classifier to identify their scopes. Velldal et al. (2012) proposes a SVM-based cue detection system, trained on simple n-grams features computed on the local lexical context (words and lemmas). This system offers a hybrid detection of the scope, which combines expert rules, operating on syntactic dependency trees, with a ranking SVM that learns a discriminative ranking function over nodes in constituent trees. It was further improved by Read et al. (2012) and is used as a fall-back by Packard et al. (2014) when the main MRS (minimal recursion semantics) Crawler cannot parse the sentence. Qian et al. (2016) addresses the scope detection with an approach based on a convolutional neural network which extracts features from various syntactic paths between the cues and the candidate tokens in constituency and dependency parsed trees. Fancellu et al. (2016) uses neural networks to solve the problem of negation scope detection. One approach uses Feed-forward neural network, while the other, which appears to be more efficient for the task, uses a bidirectional Long Short-Term Memory (BiLSTM) neural network. Given the results from the latter approach, it inspired our work.

	French clin. trials	French clin. cases
Documents	–	200
Sentences	6,547	3,811
Tokens	150,084	87,487
Vocabulary (types)	7,880	10,500
Negative sentences	1,025	804
Speculative sentences	630	226

Table 1: Statistics on the two French corpora

4 FRENCH MEDICAL CORPORA

We manually annotated two corpora from the biomedical field. Table 1 presents some statistics on these corpora: the number of words, the variety of the vocabulary, the number of sentences, the number of negative sentences with one or more negations. The Inter Annotator Agreement (IAA) on negation annotation is high (Cohen's κ=0.8461).

4.1 ESSAI: FRENCH CORPUS with CLINICAL TRIALS

One corpus contains clinical trial protocols in French. They were mainly obtained from the National Cancer Institute registry[1]. The typical protocol consists of two parts: the summary of the trial, which indicates the purpose of the trial and the methods applied; and a detailed description of the trial with the inclusion and exclusion criteria.

[1] https://www.e-cancer.fr

Form	Lemma	POS	Cue	scope
Pas	pas	ADV	pas	_
de	de	PRP	_	de
dyspnée	dyspnée	NOM	_	dyspnée
.	.	SENT	_	_

Table 2: Excerpt from the CAS corpus. The columns contain linguistic (lemmas, POS-tag) and reference (cue, scope) information.

4.2 CAS: FRENCH CORPUS with CLINICAL CASES

This corpus contains clinical cases published in scientific literature and training material. They are published in different journals from French-speaking countries (France, Belgium, Switzerland, Canada, African countries, tropical countries) and are related to various medical specialties (cardiology, urology, oncology...). The purpose of clinical cases is to describe clinical situations of patients. Hence, their content is close to the content of clinical narratives (description of diagnoses, treatments or procedures, evolution, family history, expected audience, etc.). In clinical cases, the negation is frequently used for describing the patient signs, symptoms, and diagnosis. Speculation is present as well but less frequently.

4.3 ANNOTATION LAYERS

These two corpora are Part-of-Speech tagged and lemmatized with TreeTagger (Schmid, 1994). For the creation of the reference data necessary for machine learning, both corpora were annotated manually to mark up negation and speculation cues and their scope. However, the 200 annotated clinical cases did not include enough examples of speculation for a machine learning models to train properly. Therefore, speculation detection is either trained and tested with ESSAI alone or with ESSAI and CAS. Table 2 presents an annotated sentence from CAS: *No dyspnea.*. The corpora also includes two additional columns (sentence number and token position).

5 METHODOLOGY

As indicated on Figure 1, our methods rely on specifically trained word vectors and supervised learning techniques (BiLSTM and CRF). The objective is to classify each word as being part or not of the negation/speculation cue and/or scope.

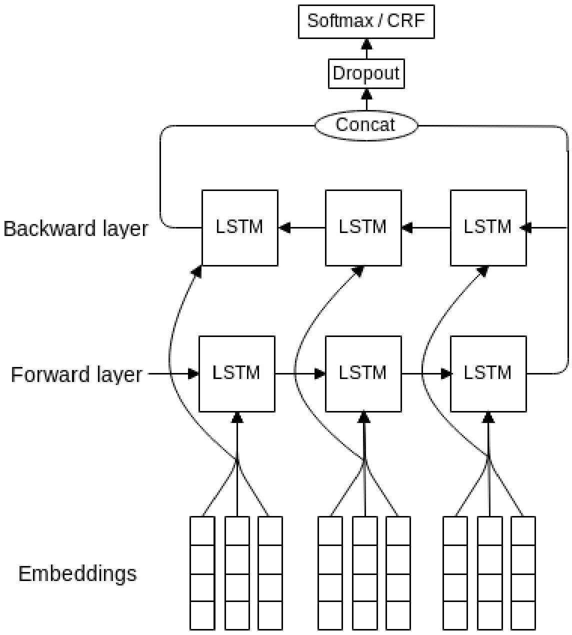


Figure 1: Our bidirectional RNN uses LSTM cells with either a softmax or CRF output layer. Features are either words/lemmas/PoS for cue detection or words/PoS/Cue information for scope detection.

5.1 WORD VECTOR REPRESENTATIONS

In the recent years, several models have been introduced to generate vector representations of words helping machine learning approaches to better capture their semantics. The models used in the negation/speculation detection task are the following ones.

word2vec (Mikolov et al., 2013) is a predictive model to learn word embeddings from plain text. The embeddings can be calculated using two model architectures: the continuous bag-of-words (CBOW) and *Skip-Gram* (SG) models. In this work, we use use the SG model; it treats each context-target pair as new observation, which is suitable for large datasets.

fastText (Bojanowski et al., 2017) addresses the Word2vec's main issue: the words, which do not occur in the vocabulary, cannot be represented. Hence, this algorithm uses subword information: each word is represented as a bag of all possible character n-grams it contains. The word is padded using a set of unique symbols which helps singling out prefixes and suffixes. The full sequence is added to the bag of n-grams as well. The vector now denotes every char n-gram and the word vector is the sum of its char n-gram vectors. Since the char n-gram representations across words are often shared, rare words can also get reliable representations.

These two word embedding models are trained using the *Skip-Gram* algorithm, 100 dimensions, a window of 5 words before and after each word, a minimum count of five occurrences for each word and negative sampling. The training data are composed of the French Wikipedia articles and biomedical data. The latter includes the ESSAI and CAS corpora, the French Medical Corpus from CRTT[2] and the Corpus QUAERO Médical du français[3] (Névéol et al., 2014). These models are trained using the Gensim[4] (Rehurek and Sojka, 2010) python library.

5.2 RECURRENT NEURAL NETWORK

Recurrent neural network takes into account the previously seen data in addition to the currently seen data. This is implemented with loops in the architecture of the network, which allows the information to persist in memory. Among the RNNs, long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) are the most efficient for the learning of long-term dependencies and are therefore more suitable to solve the problem of discontinuous scope, which is typical for the negation. LSTM cells are also more efficient at retaining useful information during backpropagation.

We use a bidirectional LSTM, which operates forward and backward on the sentence, to detect cues and scopes. The backward pass is relevant for the scope detection because the scope may be before or after the cue. Prediction is computed by either a *softmax* or a CRF (suitable for the sequence labeling) output layer. We use embeddings of dimension $k = 100$ and a dimensionality of the output space of 400 units per layer (backward/forward) with 0.5 dropout. 50 epochs achieve the highest F_1 score on the validation sets.

5.3 CONDITIONAL RANDOM FIELDS

Conditional random fields (CRFs) (Lafferty et al., 2001) are statistical methods used to label word sequences. By training a model on appropriate features and labels to be predicted, the CRFs generally obtain good results with much lower training time than neural networks.

We performed the gradient descent using the L-BFGS (Limited-memory BFGS) method with 0.1 L1 penalty and 0.01 L2 penalty. We only experiment with CRFs for the cue detection task, in comparison with BiLSTM-CRF.

5.4 EVALUATING LABELING SYSTEMS

We use standard evaluation measures: precision P, recall R, and F_1 score. The scope detection is evaluated in two ways: (1) on individual scope tokens which is the standard evaluation, and (2) on exact scopes to assess more strictly how efficient our models are. For the latter, we use the available evaluation script[5]. Each corpus is randomly segmented into the training set (80%, 20% for validation), and the test set (20%).

6 CUE DETECTION

The speculation and negation cue detection is the first step of the task. To tackle this problem, we experiment with two supervised learning approaches. First, we train a CRF using several features (words, lemmas and POS-tags) with empirically defined window over features. Our second approach uses a BiLSTM with a CRF output layer, which is trained on the same features. We did not use any pre-trained embeddings for this task.

Table 3 presents the results obtained with our approaches on the ESSAI and CAS corpora. We can see that cue detection shows high evaluation values: 93.92 to 97.21 F-measure for negation cues, and 86.88 to 91.30 F-measure for speculation cues. Although there is little room for improvement on negation cue detection, indeed 10k-fold cross-validation with our CRF reaches more than 95 F-measure on both corpora, speculation cue detection would benefit from more training examples. Indeed, the potential number of cues and the numerous contexts in which they appear and do or do not express speculation makes them harder to detect and will require more annotated examples. For both negation and speculation cue detection, our CRF is slightly more efficient than the BiLSTM-CRF when the CAS corpus is involved, which indicates that the CAS corpus contains less complex examples than ESSAI.

7 SCOPE DETECTION

In all the scope detection experiments proposed, we only train the neural networks on nega-

[2] https://bit.ly/2LOJfEW

[3] https://quaerofrenchmed.limsi.fr/

[4] https://radimrehurek.com/gensim/

[5] https://github.com/ffancellu/NegNN

	System	Corpus	window size	P	R	F_1
Negation	CRF	ESSAI	(4)	96.05	91.89	93.92
	BiLSTM-CRF		None	95.10	94.58	**94.84**
	CRF	CAS	(4)	97.05	97.37	**97.21**
	BiLSTM-CRF		None	97.02	97.02	97.02
Speculation	CRF	ESSAI	(4)	91.43	82.76	86.88
	BiLSTM-CRF		None	91	83.84	**87.27**
	CRF	ESSAI+CAS	(4)	93.93	88.82	**91.30**
	BiLSTM-CRF		None	91.22	87.92	89.54

Table 3: Precision, Recall and F$_1$-score for the cue detection task on the two corpora (bold: best scores).

				Scope tokens			Exact scope match		
	Corpus	System	WE	P	R	F_1	P	R	F_1
Negation	ESSAI	BiLSTM-S	WI	86.21	82.85	**84.50**	100	55.61	71.47
			W2V	83.54	83.68	83.61	100	56.59	72.27
			FT	80.79	86.41	83.51	100	56.59	72.27
		BiLSTM-CRF	WI	84.65	84.09	84.37	100	59.51	74.62
			W2V	83.86	83.10	83.48	100	61.95	**76.51**
			FT	82.38	84.84	83.59	100	59.51	74.61
	CAS	BiLSTM-S	WI	93.72	87.30	90.40	100	73.21	84.54
			W2V	93.03	88.69	**90.81**	100	75.59	86.10
			FT	91.50	88.69	90.08	100	72.02	83.74
		BiLSTM-CRF	WI	91.87	88.59	90.20	100	68.45	81.27
			W2V	91.47	88.29	89.85	100	76.19	86.49
			FT	94.82	87.10	90.80	100	78.57	**88.00**
Speculation	ESSAI	BiLSTM-S	WI	89.27	82.14	85.56	100	52.76	69.07
			W2V	88.61	84.42	**86.47**	100	56.69	72.36
			FT	85.84	84.58	85.21	100	58.27	**73.63**
		BiLSTM-CRF	WI	89.77	83.03	86.27	100	51.97	68.39
			W2V	87.85	83.77	85.76	100	55.91	71.72
			FT	91.04	79.61	84.94	100	57.48	73.00
	ESSAI+CAS	BiLSTM-S	WI	88.90	83.94	86.35	100	57.56	73.06
			W2V	86.20	85.19	85.69	100	58.14	73.53
			FT	85.15	87.46	86.29	100	59.30	74.45
		BiLSTM-CRF	WI	89.49	81.16	85.12	100	56.98	72.59
			W2V	88.48	85.04	**86.73**	100	65.12	**78.87**
			FT	89.15	83.14	86.04	100	62.79	77.14

Table 4: Precision, Recall and F$_1$-score for the scope detection task (bold: best scores).

				Scope tokens			Exact scope match		
System		Train	Test	P	R	F_1	P	R	F_1
BiLSTM-CRF	Negation	ESSAI	CAS	76.73	76.36	**76.54**	100	36.08	**53.03**
		CAS	ESSAI	82.36	55.23	66.12	100	28.20	43.99
	Speculation	ESSAI	CAS	76.46	68.03	**72.00**	100	45.37	**62.42**
		CAS	ESSAI	72.43	65.05	68.54	100	30.79	47.09

Table 5: Cross-corpora Precision, Recall and F$_1$-score for the scope detection task (bold: best scores).

tive/speculative sentences. The base system takes as input an instance $I(w, c, t)$, where each word is represented by: w vector (*word-embedding*), c vector (*cue-embedding*), indicating if the word is a cue or not, t vector (*POS-tag-embedding*). Preliminary tests showed that adding lemmas as features only decreases the F-measure. For each system, we use the same empirically defined hyperparameters given before. During training, embeddings weights are updated.

Table 4 indicate the results obtained for the scope detection task. One can see that it is easier to predict the scope of negation cues (up to 90.81 F-measure) than of speculation cues (up to 86.73 F-measure). Results show that using pre-trained embeddings improves F-measures for exact scope detection by up to 6 points for both negation and speculation. Moreover, the CRF output layer either outperforms the *softmax* layer or reaches an equivalent F-measure for exact scope detection. In another experiment, we trained the models on one corpus and tested them on the other (Table 5): the models trained on ESSAI are more efficient, and the negation and speculation structures are more stable in the CAS corpus. However, even though CAS (speculation) was only trained on 226 examples, the model still shows decent results in scope tokens detection.

7.1 ERROR ANALYSIS

An analysis of the results makes it possible to isolate frequent types of errors. In the following examples, the speculation and negation cues are underlined, the scope is between brackets, while the segments in bold correspond to predictions errors.

7.1.1 NEGATION

In the first example, the prediction fails at labeling ***rénale*** *(renal)*. In the majority of cases in the reference data, the scopes associated to the cue ***sans*** often only include one token, which may be causing this error that impacts recall:

- GOLD: *Le patient sortira du service de réanimation guéri et sans [insuffisance rénale] après huit jours de prise en charge et cinq séances d'hémodialyse.*
- PRED: *Le patient sortira du service de réanimation guéri et sans [insuffisance]* ***rénale*** *après huit jours de prise en charge et cinq séances d'hémodialyse.*

  (The patient will be discharged from the intensive care unit without renal failure after eight days of management and five hemodialysis sessions.)

The second example illustrates the error that impacts precision. Here, the model wrongly predicts that all tokens in the sentence are within the scope. In the reference data, the cue ***aucun*** *(any, no)* often occurs at the beginning of sentences, and in sentences with many instances of negation. The model, mostly trained on this kind of examples, may try to reproduce these structures which causes bad prediction in some cases.

- GOLD: *Les colorations spéciales (PAS, coloration de Ziehl-Neelsen, coloration de Grocott) ne [mettaient en évidence] aucun [agent pathogène].*
- PRED: ***[Les colorations spéciales (PAS, coloration de Ziehl-Neelsen, coloration de Grocott]*** *) ne [mettaient en évidence] aucun [agent pathogène].*

  (Special stains (PAS, Ziehl-Neelsen stain, Grocott stain) showed no pathogens.)

In the third example, the error impacts both precision and recall. In this example, we have two instances of negation with the same cues: ***n...pas***. Usually, its scope follows, however, in the first instance it precedes. As we do not have many examples of this kind to train on, the model fails to correctly label the sequence. In the second negation instance, the scope may be shorter than usual, which impacts precision.

- GOLD: *[Le retrait du matériel d'ostéosynthèse incriminé] n'[est] pas [systématique], ce qui explique qu'il n'[ait] pas [été proposé] à notre patient asymptomatique.*
- PRED: ***Le retrait du matériel d'ostéosynthèse incriminé*** *n'[est] pas [systématique], ce qui explique qu'il n'[ait] pas [été proposé* ***à notre patient asymptomatique*** *].*

  (The removal of the implicated osteosynthesis material is not systematic, which explains why it has not been proposed to our asymptomatic patient.)

7.1.2 SPECULATION

In the first example, the scope of the speculation has been predicted up to the end of the preposition while in the reference data, the scope covers the verbal group. This typically impacts precision.

- GOLD: *Ce médicament n'a pas la toxicité de la chimiothérapie, mais entraine une disparition des lymphocytes B normaux pendant plusieurs mois, ce qui pourrait [favoriser la survenue d'infections graves] car ces lymphocytes participent à la défense immunitaire.*
- PRED: *Ce médicament n'a pas la toxicité de la chimiothérapie, mais entraine une disparition des lymphocytes B normaux pendant plusieurs mois, ce qui pourrait [favoriser la survenue d'infections graves* ***car ces lymphocytes participent à la défense immunitaire****]*

  (This drug does not have the toxicity of chemotherapy, but causes the disappearance of normal B lymphocytes for several months, which could increase the occurrence of serious infections because these lymphocytes participate in the immune defense.)

However, most of our errors impact recall, like in the following example.

- GOLD: *Elles possèdent les caractéristiques de la tumeur et pourraient [permettre à l'avenir de faire le diagnostic de tumeur sans biopsie ainsi que de suivre l'évolution de la tumeur traitée], les CTC disparaissant quand le traitement fonctionne.*
- PRED: *Elles possèdent les caractéristiques de la tumeur et pourraient [permettre à l'avenir de faire le diagnostic de tumeur]* ***sans biopsie ainsi que de suivre l'évolution de la tumeur traitée****, les CTC disparaissant quand le traitement fonctionne.*

  (It has the characteristics of the tumor and could in the future be used to diagnose the tumor without biopsy and to follow-up the evolution of the treated tumor, the CTC disappearing when the treatment is efficient.)

Several examples show errors where both precision and recall are impacted. Usually, the reason is that they are rare cases in our corpora where the scope is reversed. For instance, in the example below, most of the scope of *devrait* precedes it, while in the reference data, its scope follows this cue.

- GOLD: *L'objectif de cet essai est d'évaluer [la diminution des complications postopératoires de l'oesophagectomie qui] devrait [être obtenue] en réalisant une partie de l'intervention sous coelioscopie, chez des patients ayant un cancer de l'oesophage résécable.*
- PRED: *L'objectif de cet essai est d'évaluer la diminution des complications postopératoires de l'oesophagectomie qui devrait [être obtenue* ***en réalisant une partie de l'intervention sous coelioscopie****], chez des patients ayant un cancer de l'oesophage résécable.*

  (The objective of this trial is to evaluate the decrease in postoperative complications of esophagectomy that should be achieved by performing part of the procedure under laparoscopy, for patients with resectable esophageal cancer.)

8 CONCLUSION and FUTURE WORK

The interest for the automatic detection of speculation and negation in English with supervised machine learning has increased in the recent years. Yet, the lack of data for other languages and for specialized domains hampers the further development of such approaches. In our work, we presented new bodies of biomedical data in French annotated with negation and speculation (cues and their scope). Prior to the dissemination to the research community, the French clinical trial protocols corpus will be finalized through the integration of new data and the computation of the inter-annotator agreement. The French CAS corpus will be distributed as more clinical cases are manually annotated, as we need more speculative sentences to train supervised learning models on this dataset. Another contribution of our work is the study of different types of word vector representations and recurrent neural networks for the detection of negation and speculation. There has not been much work of this type on French corpora, especially for the biomedical domain which contains specific negation and speculation phenomena. We showed that a *CRF* layer yields better performance than *softmax* on exact scope match. Finally, the models have been applied in a cross-corpus context. Besides, we plan to improve our neural network performance by providing richer feature. In particular, recent embedding techniques, such as BERT or ELMO (Devlin et al., 2018; Peters et al., 2018) may provide more accurate representation of the sentences. Moreover, in order to provide more accurate features, we plan to move from TreeTagger, which makes a substantial number of mistakes on our datasets, to a POS tagger/lemmatizer dedicated to French biomedical texts. Syntactic parsing of sentences may also provide useful features for the detection of scope.

ACKNOWLEDGMENTS

This work was partly funded by the French government support granted to the CominLabs LabEx managed by the ANR in Investing for the Future program under reference ANR-10-LABX-07-01.

References

Amine Abdaoui, Andon Tchechmedjiev, William Digan, Sandra Bringay, and Clement Jonquet. 2017. French context: Détecter la négation, la temporalité et le sujet dans les textes cliniques français. In *4ème Symposium sur l'Ingénierie de l'Information Médicale, SIIM'17*.

Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, William F Styler IV, Colin Warner, Jena D Hwang, Jinho D Choi, Dmitriy Dligach, Rodney D Nielsen, James Martin, et al. 2013. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association* 20(5):922–930.

Emilia Apostolova, Noriko Tomuro, and Dina Demner-Fushman. 2011. Automatic extraction of lexico-syntactic patterns for detection of negation and speculation scopes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 283–287.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

W Chapman, W Bridewell, P Hanbury, GF Cooper, and BG Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform* 34(5):301–10.

Louise Deléger and Cyril Grouin. 2012. Detecting negation of medical problems in french clinical notes. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*.

JC Denny and JF Peterson. 2007. Identifying qt prolongation from ecg impressions using natural language processing and negation detection. In *Medinfo*. pages 1283–8.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Peter L. Elkin, Steven H. Brown, Brent A. Bauer, Casey S. Husser, William Carruth, Larry R. Bergstrom, and Dietlind L. Wahner-Roedler. 2005a. A controlled trial of automated classification of negation from clinical notes. *BMC medical informatics and decision making* 5. https://doi.org/10.1186/1472-6947-5-13.

PL Elkin, SH Brown, BA Bauer, CS Husser, W Carruth, LR Bergstrom, and DL Wahner-Roedler. 2005b. A controlled trial of automated classification of negation from clinical notes. *BMC Med Inform Decis Mak.* 5(13).

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural networks for negation scope detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 1.

S Gindl, K Kaiser, and S Miksch. 2008. Syntactical negation detection in clinical practice guidelines. In *Stud Health Technol Inform*. pages 187–92.

Henk Harkema, John N Dowling, Tyler Thornblade, and Wendy W Chapman. 2009. Context: an algorithm for determining negation, experiencer, and temporal status from clinical reports. *Journal of biomedical informatics* 42(5):839–851.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8). https://doi.org/10.1162/neco.1997.9.8.1735.

Halil Kilicoglu and Sabine Bergler. 2010. A high-precision approach to detecting hedges and their scopes. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*. Association for Computational Linguistics, pages 70–77.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1.

Marc Light, Xin Ying Qiu, and Padmini Srinivasan. 2004. The language of bioscience: Facts, speculations, and statements in between. In *HLT-NAACL 2004 Workshop: Linking Biological Literature, Ontologies and Databases*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Roser Morante and Eduardo Blanco. 2012. * sem 2012 shared task: Resolving the scope and focus of negation. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*. volume 1, pages 265–274.

Aurélie Névéol, Cyril Grouin, Jeremy Leixa, Sophie Rosset, and Pierre Zweigenbaum. 2014. The quaero french medical corpus: A ressource for medical entity recognition and normalization. In *In Proc BioTextM, Reykjavik*. Citeseer.

Lilja Øvrelid, Erik Velldal, and Stephan Oepen. 2010. Syntactic scope resolution in uncertainty analysis. In *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, pages 1379–1387.

Arzucan Özgür and Dragomir R Radev. 2009. Detecting speculations and their scopes in scientific text. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. Association for Computational Linguistics, pages 1398–1407.

Woodley Packard, Emily M Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 69–78.

Yifan Peng, Xiaosong Wang, Le Lu, Mohammadhadi Bagheri, Ronald Summers, and Zhiyong Lu. 2018. Negbio: a high-performance tool for negation and uncertainty detection in radiology reports. *AMIA 2018 Informatics Summit* .

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo, and Wei Luo. 2016. Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 815–825.

Jonathon Read, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. 2012. Uio 1: Constituent-based discriminative ranking for negation resolution. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 310–318.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

Helmut Schmid. 1994. Probabilistic part-ofispeech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing, Manchester, UK.*.

Buzhou Tang, Xiaolong Wang, Xuan Wang, Bo Yuan, and Shixi Fan. 2010. A cascade method for detecting hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning—Shared Task*. Association for Computational Linguistics, pages 13–17.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association* 18(5):552–556.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational Linguistics* 38(2).

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics* 9.

Porting Multilingual Morphological Resources to OntoLex-Lemon

Thierry Declerck
German Research Center
for Artificial Intelligence
Saarbrücken, Germany
thierry.declerck@dfki.de

Stefania Racioppa
German Research Center
for Artificial Intelligence
Saarbrücken, Germany
stefania.racioppa@dfki.de

Abstract

We describe work consisting in porting various morphological resources to the OntoLex-Lemon model. A main objective of this work is to offer a uniform representation of different morphological data sets in order to be able to compare and interlink multilingual resources and to cross-check and interlink or merge the content of morphological resources of one and the same language. The results of our work will be published on the Linguistic Linked Open Data cloud.

1 Introduction

A significant number of linguistic resources have been published on the Linguistic Linked Open Data cloud (LLOD),[1] and projects like ELEXIS[2] and Prêt-à-LLOD[3] are contributing to its further extension. But we notice that only very few, if any, specific morphological resources are included in the LLOD cloud. Available morphology information is mostly contained in lexical or dictionary entries. Our aim is to make also specialized morphological resources available in this cloud. With this step we want to support the interlinking of such resources with other types of linguistic data, in a multilingual fashion, extending work described in (Gromann and Declerck, 2019), which is linking synsets of the Princeton WordNet (PWN) (Fellbaum, 1998) that are associated with plural forms to full lexical descriptions.

A first step of our current work consisted in mapping the MMorph[4] set of multilingual morphological resources to the OntoLex-Lemon model.[5]

In the next sections, we first describe briefly the Linguistic Linked Open Data cloud, and one of its salient component, the OntoLex-Lemon model. Following this summary, we describe the (multilingual) morphological resources we selected for mapping to OntoLex-Lemon. We present the result of such mappings and conclude with a description of the next steps of our work, aiming at supporting the cross-lingual comparison of morphological resources, and the cross-checking, correcting and merging of different morphological resources for one and the same language.

2 The Linguistic Linked Open Data Cloud

The LLOD initiative had its inception in 2012 at a workshop co-located with the 34th Annual Conference of the German Linguistic Society (DGfS). The workshop was organized by members of the Open Knowledge Foundation,[6] and the contributions to this workshop are available in (Chiarcos et al., 2012). The workshop has been a point of focal activity for several research and infrastructure projects, as well as for the "Ontology Lexica" W3C Community Group.[7] Those developments are described in (McCrae et al., 2016). A major result of those activities is the development of the OntoLex-Lemon model, which is described in more details in Section 3.

We adopted OntoLex-Lemon for the representation of morphological resources, as this model was shown to be able to represent both classical lexicographic description (McCrae et al., 2017) and lex-

[1] See http://www.linguistic-lod.org/

[2] https://elex.is/. See also (Krek et al., 2018) for a general overview of ELEXIS and (Declerck et al., 2018) for a focused description of the role of LLOD in the context of the project.

[3] https://www.pret-a-llod.eu.

[4] See (Petitpierre and Russell, 1995).

[5] See (Cimiano et al., 2016) and https://www.w3.org/2016/05/ontolex/.

[6] See https://okfn.org/.

[7] See for more details https://www.w3.org/community/ontolex/.

ical semantics networks, like WordNet (McCrae et al., 2014), to which we want to link full morphological descriptions.

3 OntoLex-Lemon

The OntoLex-Lemon model was originally developed with the aim to provide a rich linguistic grounding for ontologies, meaning that the natural language expressions used in the description of ontology elements are equipped with an extensive linguistic description.[8] This rich linguistic grounding includes the representation of morphological and syntactic properties of lexical entries as well as the syntax-semantics interface, i.e. the meaning of these lexical entries with respect to an ontology or to specialized vocabularies.

The main organizing unit for those linguistic descriptions is the lexical entry, which enables the representation of morphological patterns for each entry (a MWE, a word or an affix). The connection of a lexical entry to an ontological entity is marked mainly by the `denotes` property or is mediated by the `LexicalSense` or the `LexicalConcept` properties, as this is represented in Figure 1, which displays the core module of the model.

OntoLex-Lemon builds on and extends the *lemon* model (McCrae et al., 2012b). A major difference is that OntoLex-Lemon includes an explicit way to encode conceptual hierarchies, using the SKOS standard.[9] As can be seen in Figure 1, lexical entries can be linked, via the `ontolex:evokes` property, to such SKOS concepts, which can represent WordNet synsets. This structure is paralleling the relation between lexical entries and ontological resources, which is implemented either directly by the `ontolex:reference` property or mediated by the instances of the `ontolex:LexicalSense` class.

More recent developments of the model have been described in (McCrae et al., 2017). Currently two extension modules are being discussed: a lexicographic and a morphology module.[10] Our work can also be seen as preparing the field for a detailed representation of morphological components of lexical data by first porting morphological resources to the core module of OntoLex-Lemon, displayed in 1, before applying the representation guidelines of the morphology extension module, which is not yet in a stable and final state.

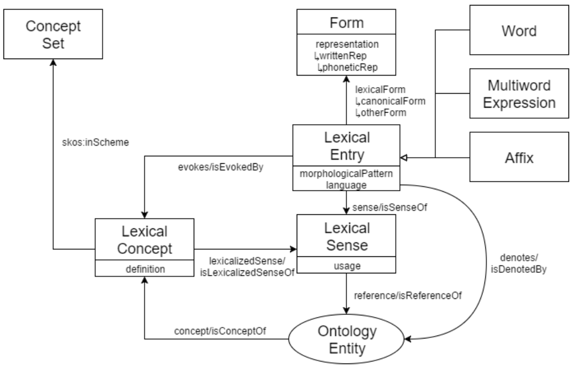


Figure 1: The core module of OntoLex-Lemon: Ontology Lexicon Interface. Graphic taken from `https://www.w3.org/2016/05/ontolex/`.

4 The Morphological Resources

We considered two types of morphological data sets. One is an updated version of the multilingual MMorph resource (Petitpierre and Russell, 1995), covering 5 languages. And we also mapped two monolingual data sets, one for German and one for Italian. We will use those additional data sets for the comparison, cross-checking and merging of monolingual morphological resources, using the uniform representation of the data in OntoLex-Lemon.

4.1 MMorph

MMorph was originally developed by ISSCO at University of Geneva in the past MULTEXT project.[11] For our purposes, we used the extended MMorph version developed at DFKI LT Lab (*MMorph3*). This version includes huge morphological resources for English, French, German, Italian and Spanish.

We choose this resource as it provides already in its original format a largely unified representation of the morphological data in the different lan-

[8] See (McCrae et al., 2012a), (Cimiano et al., 2016) and also `https://www.w3.org/community/ontolex/wiki/Final_Model_Specification`.

[9] SKOS stands for "Simple Knowledge Organization System". SKOS provides "a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary" (https://www.w3.org/TR/skos-primer/)

[10] See respectively `https://www.w3.org/community/ontolex/wiki/Lexicography` and `https://www.w3.org/community/ontolex/wiki/Morphology`.

[11] See `https://www.issco.unige.ch/en/research/projects/MULTEXT.html` for more details on the resulting `MMorph 2.3.4` version.

guages, with only few differences across the distinct sources.

Very generally, the MMorph tool relates a word to a morphosyntactic description (MSD) containing free-definable attribute and values. The MMorph lexicon used to realize such MSD consists of a set of lexical entries and structural rules. For example, the following rule creates in English a noun plural concatenating the singular form and the noun suffix “s” (Petitpierre and Russell, 1995):

```
"s" noun_suffix[number=plur]

noun[number=plur gender=$gen]
    <- noun[number=sing gender=$gen]
       noun_suffix[number=plur]
```

Note how the rule ensures that the gender does not change in the plural form. Further *adjustment rules* are defined to catch the orthographic features of a specific language (e.g. *box+s* = *boxes* in English).

The MMorph lexica can be dumped to full form lists for the usage in further programs:

Listing 1: The MMorph entry for the German noun “Aachener” (*inhabitant of Aachen*)

```
"aachener" = "aachener" Noun[gender=masc
    number=singular case=nom|dat|acc]
"aachener" = "aachener" Noun[gender=masc
    number=plural case=nom|gen|acc]
"aachenern" = "aachener" Noun[gender=masc
    number=plural case=dat]
"aacheners" = "aachener" Noun[gender=masc
    number=singular case=gen]
```

As the reader can observe in Listing 1, the nominal entries are completed by appropriate features describing *case*, *gender*, and *number*. Multiple values of a feature are expressed by “|”. The user can freely define language- and word class-specific features (e.g. *clitics* for verbal entries or *rection* of prepositions). As the example above demonstrates, the dumped lexica are ideally suited for the mapping into the OntoLex-Lemon format, as they present their data in a well structured fashion.

Our German version of MMorph contains over 2.630.000 full-forms. Compared to the original version, it has specifically improved the coverage of compounds.

To transform the MMorph data into OntoLex-Lemon we used a Python script including the `rdflib` module[12], which supports the generation of RDF-graphs in `rdf:xml`, `turtle` syntax and other relevant formats.

In Listing 2 we show the resulting OntoLex-Lemon representation of the German noun “Aachener”.

Listing 2: The OntoLex-Lemon entry for *Aachener*

```
:lex_aachener a ontolex:LexicalEntry ;
   lexinfo:gender lexinfo:masculine ;
   lexinfo:partOfSpeech lexinfo:noun ;
   ontolex:canonicalForm :form_aachener ;
   ontolex:otherForm
      :form_aachener_dat_plural ,
      :form_aachener_gen_singular ,
      :form_aachener_nom-gen-acc_plural .

:form_aachener a ontolex:Form ;
   lexinfo:case lexinfo:accusative ,
      lexinfo:dative ,
      lexinfo:nominative ;
   lexinfo:number lexinfo:singular ;
   ontolex:writtenRep "Aachener"@de .

:form_aachener_dat_plural a
      ontolex:Form ;
   lexinfo:case lexinfo:dative ;
   lexinfo:number lexinfo:plural ;
   ontolex:writtenRep "Aachenern"@de .

:form_aachener_gen_singular a
      ontolex:Form ;
   lexinfo:case lexinfo:genitive ;
   lexinfo:number lexinfo:singular ;
   ontolex:writtenRep "Aacheners"@de .

:form_aachener_nom-gen-acc_plural a
      ontolex:Form ;
   lexinfo:case lexinfo:accusative ,
      lexinfo:genitive ,
      lexinfo:nominative ;
   lexinfo:number lexinfo:plural ;
   ontolex:writtenRep "Aachener"@de .
```

The reader can observe how the relations between a lemma (an instance of the class `LexicalEntry`) and its different morphological forms (instances of the class `Form`) is made explicit by the use of named properties. Another feature of our work is the re-use of established vocabularies, for example the LexInfo vocabulary[13] to represent the morpho-syntactic features.

In Listing 3, we show examples of the resulting data for the lemma “cura” in Spanish.

Listing 3: The OntoLex-Lemon entry for *cura*

```
:lex_cura_1 a ontolex:LexicalEntry ;
    lexinfo:gender lexinfo:feminine ;
    lexinfo:partOfSpeech lexinfo:noun ;
    ontolex:canonicalForm :form_cura ;
    ontolex:otherForm :form_cura_plural .
```

[12] See `https://github.com/RDFLib/rdflib` for more details.

[13] See `https://lexinfo.net/` and (Cimiano et al., 2011) for more details

```
:lex_cura_2 a ontolex:LexicalEntry ;
    lexinfo:gender lexinfo:masculine ;
    lexinfo:partOfSpeech lexinfo:noun ;
    ontolex:canonicalForm :form_cura ;
    ontolex:otherForm :form_cura_plural .

:form_cura a ontolex:Form ;
    lexinfo:number lexinfo:singular ;
    ontolex:writtenRep "cura"@es .

:form_cura_plural a ontolex:Form ;
    lexinfo:number lexinfo:plural ;
    ontolex:writtenRep "curas"@es .
```

As the reader can observe, we have two lexical entries for the entry "cura", as this is suggested by the lexicographic module of Ontolex-Lemon.[14] "Cura" in feminine means *cure* or *healing*, while in masculine it refers to a *cure*. But one can also propose a unique entry for "cura" and add in each of the associated senses a usage restriction indicating the gender of the corresponding `ontolex:Form`.

The reader can also see the harmonized representation of morphological resources across languages (here German and Spanish). This is an important feature that will allow to link various lemmas (or senses) from different languages to a unique reference point in external information sources, like WordNet(s)[15] or knowledge Graphs, like DBpedia[16] or Wikidata[17].

The transformation of nominal entries from MMorph to the OntoLex-Lemon format resulted in 67778 instances of the class `LexicalEntry` for German, 17313 for Spanish, 21085 for Italian, 29959 for English and 13525 for French. The English nominal data in OntoLex-Lemon include 59108 instances of the class `Form`, while the German data consists of 224449 such forms. This largely depends on the maintenance state of the original resources, but gives nevertheless a good idea on the difference of morphological variations in the distinct languages.

4.2 Two Monolingual Resources

Other data sets we are considering are "Morph-it!"[18] for Italian and the "DE_morph_dict data"[19].

An entry in Morph-it! has the form displayed in Listing 4:

Listing 4: The Morph-it! entry for "abbassamento" (*lowering* or *reduction*)

```
abbassamento    abbassamento    NOUN–M:s
abbassamenti    abbassamento    NOUN–M:p
```

The corresponding OntoLex-Lemon encoding is displayed in Listing 5. While this representation looks much more complex than the original Morph-it! one, it represents the relations in an explicit and declarative way and at the same time it gives a full "autonomy" to the form variants, which are now represented as instances of the class `ontolex:Form` and equipped with an URI, so that they can be accessed independently of their corresponding headword.

Listing 5: The OntoLex-Lemon representation for "abbassamento" (*lowering* or *reduction*)

```
:lex_abbassamento a
        ontolex:LexicalEntry;
   lexinfo:gender lexinfo:masculine ;
   lexinfo:partOfSpeech lexinfo:noun ;
   ontolex:canonicalForm
:form_abbassamento ;
   ontolex:otherForm
     :form_abbassamento_m_p .

:form_abbassamento a ontolex:Form ;
   lexinfo:number lexinfo:singular ;
   ontolex:writtenRep "abbassamento"@it.

:form_abbassamento_m_p a ontolex:Form ;
   lexinfo:number lexinfo:plural ;
   ontolex:writtenRep "abbassamenti"@it.
```

In Listing 6 below, we display an original entry of the DE_morph_dict resource, in this example the word "Abgang" (*departure*, *leaving*, *dispatch*, etc.). The reader can immediately see the difference to the Italian entry in Listing 4, as in German there are four cases and three genders, a fact which leads to a high number of morphological form variants. Also this entry is including obsolete forms of the word, which is adding an additional line in the original encoding.

[14] See the discussion on this case at `https://www.w3.org/community/ontolex/wiki/Lexicography`.

[15] Concerning the linking to WordNets, we started linking the French, Italian and Spanish morphological data to their counterparts in the Open Multilingual WordNet initiative. See `http://compling.hss.ntu.edu.sg/omw/` and (Bond and Paik, 2012) for more details

[16] `https://wiki.dbpedia.org/`

[17] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

[18] `https://docs.sslmit.unibo.it/doku.php?id=resources:morph-it`. See also (Zanchetta and Baroni, 2005)).

[19] `https://github.com/DuyguA/german-morph-dictionaries`. See also for this resource the companion morphological analyser in (Altinok, 2018)).

Listing 6: The DE_morph_dict entry for “Abgang” (*departure* or *leaving* etc)

```
Abgang
Abgang NN,masc,acc,sing
Abgang NN,masc,nom,sing
Abgang NN,masc,dat,sing
Abgange
Abgang NN,masc,dat, ing,old
Abganges
Abgang NN,masc,gen,sing
Abgangs
Abgang NN,masc,gen,sing
Abgänge
Abgang NN,masc,nom,plu
Abgang NN,masc,acc,plu
Abgang NN,masc,gen,plu
Abgängen
Abgang NN,masc,dat,plu
```

The mapping of this entry to OntoLex-Lemon results in a representation that is by now familiar, and which is given in Listing 7.

Listing 7: The OntoLex-Lemon representation for “Abgang” (*departure* or *leaving* etc)

```
:lex_abgang a ontolex:LexicalEntry ;
    lexinfo:gender lexinfo:masculine ;
    lexinfo:partOfSpeech lexinfo:noun ;
    ontolex:canonicalForm :form_abgang ;
    ontolex:otherForm
        :form_abgang_dat_plu ,
        :form_abgang_dat_sing ,
        :form_abgang_gen_sing ,
        :form_abgang_nom-gen-acc_plu .

:form_abgang a ontolex:Form ;
    lexinfo:case lexinfo:accusative ,
        lexinfo:dative ,
        lexinfo:nominative ;
    lexinfo:number lexinfo:singular ;
    ontolex:writtenRep "Abgang"@de .

:form_abgang_dat_plu a ontolex:Form ;
    lexinfo:case lexinfo:dative ;
    lexinfo:number lexinfo:plural ;
    ontolex:writtenRep "Abgängen"@de .

(etc.)
```

With those resources represented in OntoLex-Lemon, which are duplicating the German and Italian resources we have already from MMorph, we aim at discovering possible inconsistencies or similarities within resources for one language, which could lead to both a improvement and a merging of the original resources.

We are in a sense extending a former experiment on automatically merging Italian morphological resources in the context of a finite automata environment, and which is described in (Declerck et al., 2012). The new work is not only a multilingual extension, but is aiming at a broad interoperability of morphological resources by using a defacto standard developed by a W3C Community Group and publishing the results in an accessible subset of the Linked Data cloud.

5 Conclusion

We described our current work consisting in porting a number of (multilingual) morphological resources to OntoLex-Lemon, in order to harmonize those and to support their interlinking, cross-checking, but also their linking with other data source in the Linguistic Linked Open Data, as for examples WordNets, or with data sets included in knowledge graphs, like DBpedia or Wikidata.

As a final goal of our work, we see the possibility to interlink or merge those morphological resources in the Linguistic Linked Open Data cloud.

Acknowledgments

Work presented in this paper has been supported in part by the H2020 project “Prêt-à-LLOD” with Grant Agreement number 825182. Contributions by Thierry Declerck are also supported in part by the H2020 project “ELEXIS” with Grant Agreement number 731015.

References

Duygu Altinok. 2018. Demorphy, german language morphological analyzer. *CoRR*, abs/1803.00902.

Francis Bond and Kyonghee Paik. 2012. A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)*, pages 64–71, Matsue.

Christian Chiarcos, Sebastian Nordhoff, and Sebastian Hellmann, editors. 2012. *Linked Data in Linguistics - Representing and Connecting Language Data and Language Metadata*. Springer.

Philipp Cimiano, Paul Buitelaar, John McCrae, and Michael Sintek. 2011. Lexinfo: A declarative model for the lexicon-ontology interface. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):29–51.

Philipp Cimiano, John P. McCrae, and Paul Buitelaar. 2016. Lexicon Model for Ontologies: Community Report.

Thierry Declerck, John McCrae, Roberto Navigli, Ksenia Zaytseva, and Tanja Wissik. 2018. Elexis - european lexicographic infrastructure: Contributions to and from the linguistic linked open data. In *Proceedings of the 2nd GLOBALEX Workshop. GLOBALEX (GLOBALEX-2018), Lexicography & WordNet, located at 11th Language Resources and Evaluation Conference (LREC 2018), May 8, Miyazaki, Japan*, pages 17–22. ELRA.

Thierry Declerck, Stefania Racioppa, and Karlheinz Mörth. 2012. Automatized merging of italian lexical resources. In *Proceeding of the LREC 2012 Workshop on Language Resource Merging*, Paris. ELRA, ELRA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London.

Dagmar Gromann and Thierry Declerck. 2019. Towards the detection and formal representation of semantic shifts in inflectional morphology. In *2nd Conference on Language, Data and Knowledge (LDK)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 21:1–21:15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Simon Krek, Iztok Kosem, John P. McCrae, Roberto Navigli, Bolette S. Pedersen, Carole Tiberius, and Tanja Wissik. 2018. European lexicographic infrastructure (elexis). In *Proceedings of the XVIII EURALEX International Congress: Lexicography in Global Contexts*, pages 881–891, Ljubljana, Slovenia. Ljubljana University Press, Faculty of Arts.

John McCrae, Guadalupe Aguado-de Cea, Paul Buitelaar, Philipp Cimiano, Thierry Declerck, Asuncion Gomez-Perez, Jorge Garcia, Laura Hollink, Elena Montiel-Ponsoda, Dennis Spohr, and Tobias Wunner. 2012a. Interchanging Lexical Resources on the Semantic Web. *Language Resources and Evaluation*, 46(4):701–719.

John McCrae, Guadalupe Aguado de Cea, Paul Buitelaar, Philipp Cimiano, Thierry Declerck, Asunción Gómez-Pérez, Jorge Gracia, Laura Hollink, Elena Montiel-Ponsoda, D ennis Spohr, and Tobias Wunner. 2012b. Interchanging lexical resources on the Semantic Web. *Language Resources and Evaluation*, 46(6):701–709.

John P. McCrae, Paul Buitelaar, and Philipp Cimiano. 2017. The OntoLex-Lemon Model: Development and Applications. In *Proceedings of eLex 2017*, pages 587–597. INT, Trojína and Lexical Computing, Lexical Computing CZ s.r.o.

John P. McCrae, Christian Chiarcos, Francis Bond, Philipp Cimiano, Thierry Declerck, Gerard de Melo, Jorge Gracia, Sebastian Hellmann, Bettina Klimek, Steven Moran, Petya Osenova, Antonio Pareja-Lora, and Jonathan Pool. 2016. The open linguistics working group: Developing the linguistic linked open data cloud. In *The 10th edition of the Language Resources and Evaluation Conference, 23-28 May 2016, Slovenia, Portorož*.

John P. McCrae, Christiane Fellbaum, and Philipp Cimiano. 2014. Publishing and linking wordnet using lemon and rdf. In *Proceedings of the 3 rd Workshop on Linked Data in Linguistics*.

Dominique Petitpierre and Graham. Russell. 1995. MMORPH: The Multext morphology program. Multext deliverable 2.3.1, ISSCO, University of Geneva.

Eros Zanchetta and Marco Baroni. 2005. Morph-it! a free corpus-based morphological resource for the italian language. In *Proceedings of Corpus Linguistics 2005*. University of Birmingham.

Dependency-Based Self-Attention for Transformer NMT

Hiroyuki Deguchi, Akihiro Tamura, Takashi Ninomiya
Ehime University
{deguchi@ai., tamura@, ninomiya@}cs.ehime-u.ac.jp

Abstract

In this paper, we propose a new Transformer neural machine translation (NMT) model that incorporates dependency relations into self-attention on both source and target sides, dependency-based self-attention. The dependency-based self-attention is trained to attend to the modifiee for each token under constraints based on the dependency relations, inspired by linguistically-informed self-attention (LISA). While LISA was originally designed for Transformer encoder for semantic role labeling, this paper extends LISA to Transformer NMT by masking future information on words in the decoder-side dependency-based self-attention. Additionally, our dependency-based self-attention operates at subword units created by byte pair encoding. Experiments demonstrate that our model achieved a 1.0 point gain in BLEU over the baseline model on the WAT'18 Asian Scientific Paper Excerpt Corpus Japanese-to-English translation task.

1 Introduction

In the field of machine translation (MT), the Transformer model (Vaswani et al., 2017) has outperformed recurrent neural network (RNN)-based models (Sutskever et al., 2014) and convolutional neural network (CNN)-based models (Gehring et al., 2017) on many translation tasks, and thus has garnered attention from MT researchers. The Transformer model computes the strength of a relationship between two words in a sentence by means of a self-attention mechanism, which has contributed to the performance improvement in not only MT but also various NLP tasks such as language modeling and semantic role labeling (SRL).

The performance of MT, including statistical machine translation and RNN-based neural machine translation (NMT), has been improved by incorporating sentence structures (Lin, 2004; Chen et al., 2017; Eriguchi et al., 2017; Wu et al., 2018). In addition, Strubell et al. (2018) have improved a Transformer-based SRL model by incorporating dependency structures of sentences into self-attention, which is called linguistically-informed self-attention (LISA). In LISA, one attention head of a multi-head self-attention is trained with constraints based on dependency relations to attend to syntactic parents for each token.

In the present work, we aim to improve translation performance by utilizing dependency relations in Transformer NMT. To this end, we propose a Transformer NMT model that incorporates dependency relations into self-attention on both source and target sides. Specifically, in training, a part of self-attention is learned with constraints based on dependency relations of source or target sentences to attend to a modifiee for each token, and, in decoding, the proposed model translates a sentence in consideration of dependency relations in both the source and target sides, which are captured by our self-attention mechanisms. Hereafter, the proposed self-attention is called dependency-based self-attention. Note that the dependency-based self-attention is inspired by LISA, but the straightforward adaptation of LISA, which is proposed for Transformer encoder, does not work well for NMT because a target sentence is not fully revealed in inference. Therefore, the proposed model masks future information on words in the decoder-side dependency-based self-attention to prevent from attending to unpredicted subsequent tokens.

Recent NMT models treat a sentence as a sub-

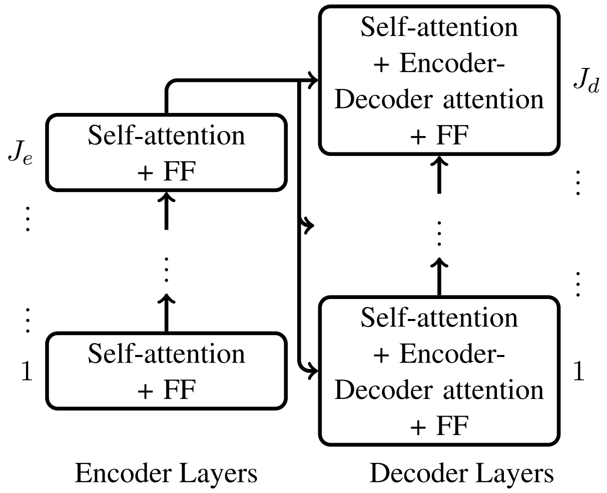

Self-attention + Encoder-Decoder attention + FF
J_d
J_e
Self-attention + FF
⋮
⋮
⋮
⋮
⋮
Self-attention + Encoder-Decoder attention + FF
1
Self-attention + FF
1
Encoder Layers
Decoder Layers

Figure 1: Transformer model.

word sequence rather than a word sequence to address the translation of out-of-vocabulary words (Sennrich et al., 2016). Therefore, we extend dependency-based self-attention to operate at subword units created by byte pair encoding (BPE) rather than word-units.

Our experiments demonstrate that the proposed Transformer NMT model performs 1.0 BLEU points higher than the baseline Transformer NMT model, which does not incorporate dependency structures, on the WAT'18 Asian Scientific Paper Excerpt Corpus (ASPEC) Japanese-to-English translation task. The experiments also demonstrate the effectiveness of each of our proposals, namely, encoder-side dependency-based self-attention, decoder-side dependency-based self-attention, and extension for BPE.

2 Transformer NMT

We provide here an overview of the Transformer NMT model (Vaswani et al., 2017), which is the basis of our proposed model. The outline of the Transformer NMT model is shown in Fig. 1.

The Transformer NMT model is an encoder-decoder model that has a self-attention mechanism. The encoder maps an input sequence of symbol representations (i.e., a source sentence) $X = (x_1, x_2, \ldots, x_{n_{enc}})^T$ to an intermediate vector. Then, the decoder generates an output sequence (i.e., a target sentence) $Y = (y_1, y_2, \ldots, y_{n_{dec}})^T$, given the intermediate vector. The encoder and the decoder are composed of a stack of J_e encoder layers and of J_d decoder layers, respectively.

Because the Transformer model does not include recurrent or convolutional structures, it encodes word positional information as sinusoidal positional encodings:

$$P_{(pos,2i)} = sin(pos/10000^{2i/d}), \tag{1}$$
$$P_{(pos,2i+1)} = cos(pos/10000^{2i/d}), \tag{2}$$

where pos is the position, i is the dimension, and d is the dimension of the intermediate representation. At the first layers of the encoder and decoder, the positional encodings calculated by Equations (1) and (2) are added to the input embeddings.

The j-th encoder layer's output $S_{enc}^{(j)}$ is generated by a self-attention layer $SelfAttn()$ and a position-wise fully connected feed forward network layer $FFN()$ as follows:

$$H_{enc}^{(j)} = LN(S_{enc}^{(j-1)} + SelfAttn(S_{enc}^{(j-1)})), \tag{3}$$
$$S_{enc}^{(j)} = LN(H_{enc}^{(j)} + FFN(H_{enc}^{(j)})), \tag{4}$$

where $S_{enc}^{(0)}$ is the input of the encoder, $H_{enc}^{(j)}$ is the output of the j-th encoder's self-attention, and $LN()$ is layer normalization (Lei Ba et al., 2016). The j-th decoder layer's output $S_{dec}^{(j)}$ is generated by an encoder-decoder attention layer $EncDecAttn()$ in addition to the two sublayers of the encoder (i.e., $SelfAttn()$ and $FFN()$) as follows:

$$H_{dec}^{(j)} = LN(S_{dec}^{(j-1)} + SelfAttn(S_{dec}^{(j-1)})), \tag{5}$$
$$G_{dec}^{(j)} = LN(H_{dec}^{(j)} + EncDecAttn(H_{dec}^{(j)})), \tag{6}$$
$$S_{dec}^{(j)} = LN(G_{dec}^{(j)} + FFN(H_{dec}^{(j)})), \tag{7}$$

where $S_{dec}^{(0)}$ is the input of the decoder, $H_{dec}^{(j)}$ is the output of the j-th decoder's self-attention, and $G_{dec}^{(j)}$ is the output of the j-th decoder's encoder-decoder attention.

The last decoder layer's output $S_{dec}^{(J_d)}$ is linearly mapped to a V-dimensional matrix, where V is the output vocabulary size. Then, the output sequence Y is generated based on $P(Y \mid X)$, which is calculated by applying the softmax function to the V-dimensional matrix.

Self-attention computes the strength of the relationship between two words in the same sentence (i.e., between two source words or between two target words), and encoder-decoder attention computes the strength of the relationship between a source word and a target word.

Both the self-attention and encoder-decoder attention are implemented with multi-head attention, which projects the embedding space into n_{head} subspaces of the $d_{head} = d/n_{head}$ dimension and calculates attention in each subspace. In the j-th layer's self-attention, the previous layer's output $S^{(j-1)} \in \mathbb{R}^{n \times d}$ is linearly mapped to three d_{head}-dimensional subspaces, $Q_h^{(j)}$, $K_h^{(j)}$, and $V_h^{(j)}$, using parameter matrices $W_h^{Q^{(j)}} \in \mathbb{R}^{d \times d_{head}}$, $W_h^{K^{(j)}} \in \mathbb{R}^{d \times d_{head}}$, and $W_h^{V^{(j)}} \in \mathbb{R}^{d \times d_{head}}$, where n is the length of the input sequence and $1 \leq h \leq n_{head}$[1]. In the j-th decoder layer's encoder-decoder attention, the previous layer's output $S_{dec}^{(j-1)}$ is mapped to $Q_h^{(j)}$, and the last encoder layer's output $S_{enc}^{(J_e)}$ is mapped to $K_h^{(j)}$ and $V_h^{(j)}$.

Then, an attention weight matrix, where each value represents the strength of the relationship between two words, is calculated on each subspace as follows:

$$A_h^{(j)} = softmax(d_{head}^{-0.5} Q_h^{(j)} K_h^{(j)^T}). \tag{8}$$

By multiplying $A_h^{(j)}$ and $V_h^{(j)}$, a weighted representation matrix $M_h^{(j)}$ is obtained:

$$M_h^{(j)} = A_h^{(j)} V_h^{(j)}. \tag{9}$$

$M_h^{(j)}$ in self-attention includes the strengths of the relationships with all words in the same sentence for each source or target word, and $M_h^{(j)}$ in encoder-decoder attention includes the strengths of the relationships with all source words for each target word.

Finally, the concatenation of all $M_h^{(j)}$ (i.e., $M_{1,2,\ldots,n_{head}}^{(j)}$) is mapped to a d-dimensional matrix $M^{(j)}$ as follows:

$$M^{(j)} = W^{M^{(j)}}[M_1^{(j)}; \ldots; M_{n_{head}}^{(j)}], \tag{10}$$

where $W^{M^{(j)}} \in \mathbb{R}^{d \times d}$ is a parameter matrix.

Note that, in training, the decoder's self-attention masks future words so as to ensure that the attentions of a target word do not rely on unpredicted words in inference.

3 Proposed Method

Figure 2 shows the outline of the proposed model. The proposed model incorporates dependency relations into self-attention on both source and target sides, *dependency-based self-attention*. In particular, it parses the dependency structures of source sentences and target sentences by one attention head of the p_e-th encoder layer's multi-head self-attention and one of the p_d-th decoder layer's multi-head self-attention, respectively, and translates a sentence based on the source-side and target-side dependency structures. We use the deep bi-affine parser (Dozat and Manning, 2016) as a model for dependency parsing in the dependency-based self-attention according to LISA. There are two inherent differences between LISA and our dependency-based self-attention: (i) our decoder-side dependency-based self-attention masks future information on words, and (ii) our dependency-based self-attention operates at subword units created by byte pair encoding rather than word-units.

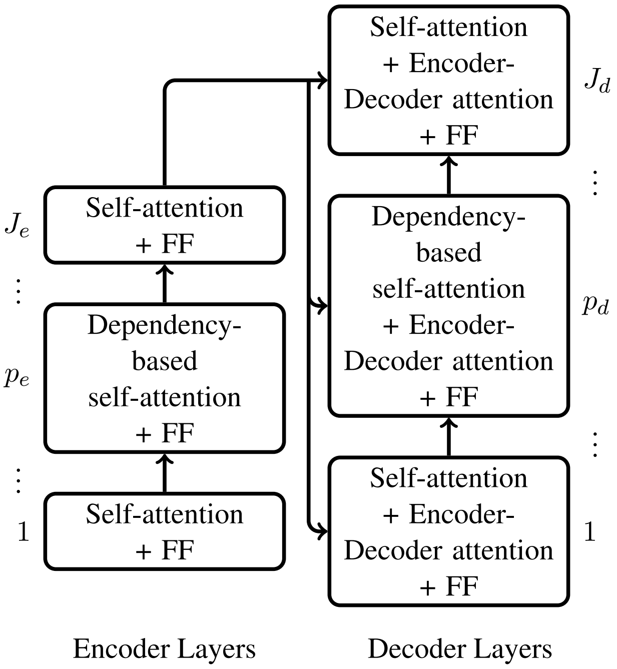

Self-attention + Encoder-Decoder attention + FF
J_d
⋮
Self-attention + FF
J_e
⋮
Dependency-based self-attention + Encoder-Decoder attention + FF
p_d
Dependency-based self-attention + FF
p_e
⋮
⋮
Self-attention + FF
1
Self-attention + Encoder-Decoder attention + FF
1
Encoder Layers
Decoder Layers

Figure 2: Proposed model.

3.1 Dependency-Based Self-Attention

The dependency-based self-attention parses dependency structures by extending the multi-head self-attention of the p-th layer of the encoder or decoder[2]. First, the p-th self-attention layer maps the previous layer's output $S^{(p-1)}$ of d-dimension to d_{head}-dimensional subspaces of multi-head at-

[1] $S^{(j)}$ indicates $S_{enc}^{(j)}$ for the encoder and $S_{dec}^{(j)}$ for the decoder.

[2] p indicates p_e for the encoder and p_j for the decoder.

tention as follows:

$$Q_{parse} = S^{(p-1)} W^{Q_{parse}}, \tag{11}$$

$$K_{parse} = S^{(p-1)} W^{K_{parse}}, \tag{12}$$

$$V_{parse} = S^{(p-1)} W^{V_{parse}}, \tag{13}$$

where $W^{Q_{parse}}$, $W^{K_{parse}}$, and $W^{V_{parse}}$ are $d \times d_{head}$ weight matrices. Next, an attention weight matrix A_{parse}, where each value indicates the dependency relationship between two words, is calculated by using the bi-affine operation as follows:

$$A_{parse} = softmax(Q_{parse} U^{(1)} K_{parse}^{T} + Q_{parse} U^{(2)}), \tag{14}$$

where $U^{(1)} \in \mathbb{R}^{d_{head} \times d_{head}}$, $U^{(2)} = \overbrace{(\mathbf{u} \ldots \mathbf{u})}^{n}$, and $\mathbf{u} \in \mathbb{R}^{d_{head}}$ are the parameters. In A_{parse}, the probability of token q being the head of token t (i.e., t modifying q) is modeled as $A_{parse}[t, q]$:

$$P(q = head(t) \mid X) = A_{parse}[t, q], \tag{15}$$

where X is a source sentence or a target sentence, and the root token is defined as having a self-loop (i.e., $q = head(t) = ROOT$). Then, a weighted representation matrix M_{parse}, which includes dependency relationships in the source sentence or target sentence, is obtained by multiplying A_{parse} and V_{parse}:

$$M_{parse} = A_{parse} V_{parse}. \tag{16}$$

Finally, after one attention head (e.g., $M^{(p)}_{n_{head}}$) is replaced with M_{parse}, the concatenation of all $M^{(p)}_h$ (i.e., M_{parse} and $M^{(p)}_{1,2,\ldots,n_{head-1}}$) is mapped to a d-dimensional matrix $M^{(p)}$ like the conventional multi-head attention:

$$M^{(p)} = W^{M^{(p)}} [M_{parse}; M^{(p)}_1; \ldots; M^{(p)}_{n_{head}-1}], \tag{17}$$

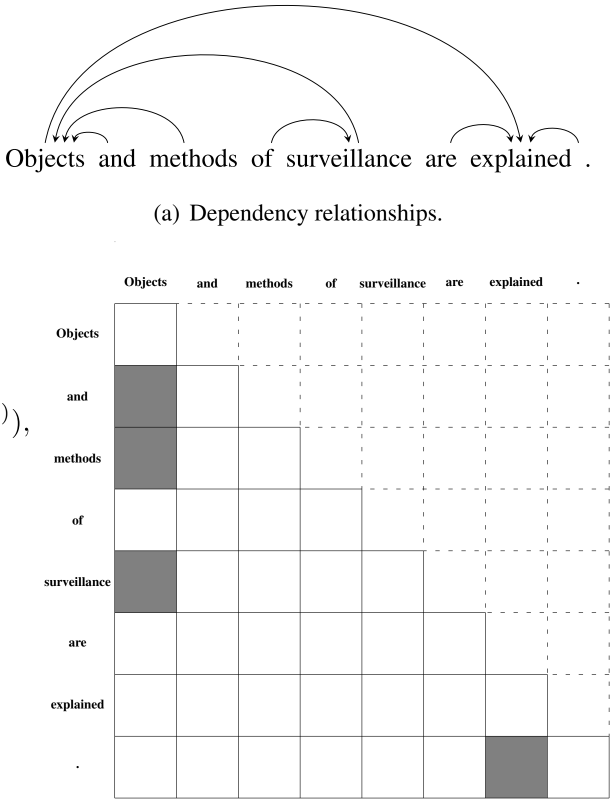
where $W^{M^{(p)}} \in \mathbb{R}^{d \times d}$ is a parameter matrix.

As can be seen in Equation (17), in the dependency-based self-attention, dependency relations are identified by one attention head M_{parse} of the p-th layer's multi-head attention.

3.2 Objective Function

Our model learns translation and dependency parsing at the same time by minimizing the following objective function:

$$e^{tokens} + \lambda_{enc} e^{parse}_{enc} + \lambda_{dec} e^{parse}_{dec}, \tag{18}$$



(a) Dependency relationships.

(b) Attention matrix representing supervisions.

Figure 3: Decoder side masked dependency-based self-attention.

where e^{tokens} is the error of translation, and e^{parse}_{enc} and e^{parse}_{dec} are the errors of dependency parsing in the encoder and the decoder, respectively. $\lambda_{enc} > 0$ and $\lambda_{dec} > 0$ are hyper-parameters to control the influence of dependency parsing errors in the encoder and the decoder, respectively. e^{tokens} is calculated by label smoothed cross entropy (Szegedy et al., 2016), and e^{parse}_{enc} and e^{parse}_{dec} are calculated by cross entropy.

Note that, in the training of the decoder-side dependency-based self-attention, future information is masked to prevent attending to unpredicted tokens in inference. An example of training data for the decoder-side dependency-based self-attention is provided in Figure 3, where (a) is an example of dependency structures[3] and (b) shows the attention matrix representing the supervisions from (a). In (b), a dark cell indicates a dependency relation and a dotted cell means a masked

[3] In this paper, an arrow is drawn from a modifier to its modifiee. For example, the arrow drawn from "Objects" to "explained" indicates that "Objects" modifies "explained" (i.e., "explained" = $head$("Objects")).

	sentence pairs
train	1,198,149
dev	1,790
test	1,812

Table 1: Statistics of the ASPEC data.

Model	BLEU
Trans.	27.29
Trans. + DBSA(Enc)	28.05
Trans. + DBSA(Dec)	27.86
Trans. + DBSA(Enc) + DBSA(Dec)	**28.29**

Table 2: Translation performance.

element. As shown, future information on each word is masked. For example, the dependency relation from "are" to "explained" is masked.

3.3 Subword Dependency-Based Self-Attention

Recent NMT models have improved the translation performance by treating a sentence as a subword sequence rather than a word sequence. Therefore, we extend dependency-based self-attention to work for subword sequences. In our subword dependency-based self-attention, a sentence is divided into a subword sequence by BPE (Sennrich et al., 2016). When a word is divided into multiple subwords, the modifiee (i.e., the head) of the rightmost subword is set to the modifiee of the original word and the modifiee of each subword other than the rightmost one is set to the right adjacent subword.

Figure 4 shows an example of subword-level dependency relations, where "@@" is a subword segmentation symbol. "Fingerprint" is divided into the three subwords: "Fing@@", "er@@", and "print". When the head of the word "Fingerprint" is "input" in the original word-level sentence, the heads of the three subwords are determined as follows: "er@@" = head("Fing@@"), "print" = head("er@@"), and "input" = head("print").

4 Experiments

4.1 Experiment Settings

In our experiments, we compared the proposed model with a conventional Transformer NMT model, which does not incorporate dependency structures, to confirm the effectiveness of the proposed model. We stacked six layers for each encoder and decoder and set $n_{head} = 8$ and $d = 512$. For the proposed model, we incorporated dependency-based self-attention into the fourth layer in both the encoder and the decoder (i.e., $p_e = p_d = 4$).

We evaluated translation performance on the WAT'18 ASPEC (Nakazawa et al., 2016) Japanese-to-English translation task. We tokenized each Japanese sentence with $KyTea$ (Neubig et al., 2011) and preprocessed according to the recommendations from WAT'18[4]. We used the vocabulary of 100K subword tokens based on BPE for both languages and used the first 1.5 million translation pairs as the training data. In the training, long sentences with over 250 subword-tokens were filtered out. Table 1 shows the statistics of our experiment data.

We used Japanese dependency structures generated by EDA[5] and English dependency structures generated by Stanford Dependencies[6] in the training of the source-side dependency-based self-attention and the target-side dependency-based self-attention, respectively. Note that Stanford Dependencies and EDA are not used in the testing.

4.2 Training Details

We trained each model using Adam (Kingma and Ba, 2014), where the learning rate and hyperparameter settings are set following Vaswani et al. (2017). For the objective function, we set ϵ_{ls} (Szegedy et al., 2016) in label smoothing to 0.1 and both the hyperparameters λ_{enc} and λ_{dec} to 1.0. We set the mini-batch size to 224 and the number of epochs to 20. We chose the model that achieved the best BLEU score on the development set and evaluated the sentences generated from the test set using beam search with a beam size of 4 and length penalty $\alpha = 0.6$ (Wu et al., 2016).

4.3 Experiment Results

Table 2 lists the experiment results. Translation performance is measured by BLEU (Papineni et al., 2002). In Table 2, DBSA denotes our dependency-based self-attention. As shown, our proposed model

[4] http://lotus.kuee.kyoto-u.ac.jp/WAT/WAT2018/baseline/dataPreparationJE.html

[5] http://www.ar.media.kyoto-u.ac.jp/tool/EDA

[6] https://nlp.stanford.edu/software/stanford-dependencies.html

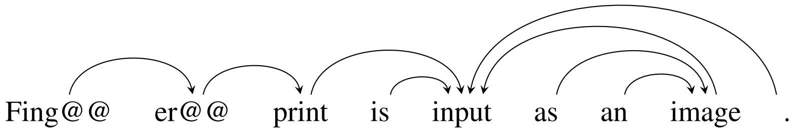


Figure 4: Subword-level dependency relationships.

Model	BPE	BLEU
Trans.	w/o	26.39
Trans.	w/	27.29
Trans. + DBSA(Enc) + DBSA(Dec)	w/o	26.62
Trans. + DBSA(Enc) + DBSA(Dec)	w/	**28.29**

Table 3: Effectiveness of subword.

"Trans.+DBSA(Enc)+DBSA(Dec)" performed significantly better than the baseline model "Trans.", which demonstrates the effectiveness of our dependency-based self-attention. Table 2 also shows that using either the encoder-side dependency-based self-attention or the decoder-side dependency-based self-attention improves translation performance, and using them in combination achieves further improvements.

5 Discussion

To determine the effectiveness of our extension to utilize subwords, we evaluated the models without BPE, where each sentence is treated as a word sequence. In the models without BPE, words that appeared fewer than five times in the training data were replaced with the special token "<UNK>". Table 3 lists the results. As shown, BPE improves the performance of both the baseline and the proposed model, which demonstrates the effectiveness of the subword dependency-based self-attention. Table 3 also shows that the proposed model outperforms the baseline model when BPE is not used. This strengthens the usefulness of our dependency-based self-attention.

6 Related Work

NMT models have been improved by incorporating source-side dependency relations (Chen et al., 2017), or target-side dependency relations (Eriguchi et al., 2017), or both (Wu et al., 2018).

Chen et al. (2017) have proposed SDRNMT, which computes dependency-based context vectors from source-side dependency trees by CNN and then uses the representations in the encoder of an RNN-based NMT model.

Eriguchi et al. (2017) have proposed NMT+RNNG, which combines the RNN-based dependency parser, RNNG (Dyer et al., 2016), and the decoder of an RNN-based NMT model.

Wu et al. (2018) have proposed a syntax-aware encoder, which encodes two extra sequences linearized from source-side dependency trees in addition to word sequences, and have incorporated Action RNN, which implements a shift-reduce transition-based dependency parsing by predicting action sequences, into the decoder. Their method has been applied to an RNN-based NMT model and a Transformer NMT model.

As far as we know, except for Wu et al. (2018), existing dependency-based NMT models have been based on RNN-based NMT. Although Wu et al. (2018) used dependency relations in Transformer NMT, they did not modify the Transformer model itself. In contrast, we have improved a Transformer NMT model to explicitly incorporate dependency relations (i.e., dependency-based self-attention). In addition, while Wu et al. (2018) need a parser for constructing source-side dependency structures in inference, our proposed method does not require an external parser in inference because the learned dependency-based self-attention of the encoder finds dependency relations.

7 Conclusion

In this paper, we have proposed a method to incorporate dependency relations on both source and target sides into Transformer NMT through dependency-based self-attention. Our decoder-side dependency-based self-attention masks future information to avoid conflicts between training and inference. In addition, our dependency-based self-attention is extended to work well for subword sequences. Experimental results showed that the proposed model achieved a 1.0 point gain in BLEU over the baseline Transformer model on the WAT'18 ASPEC Japanese-English translation task. In future work, we will explore the effectiveness of our proposed method for language pairs other than Japanese-to-English.

Acknowledgement

The research results have been achieved by "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation", the Commissioned Research of National Institute of Information and Communications Technology (NICT) , JAPAN. This work was partially supported by JSPS KAKENHI Grant Number JP18K18110.

References

Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita, and Tiejun Zhao. 2017. Neural machine translation with source dependency representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2846–2852.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734* .

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 199–209.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pages 72–78.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pages 1243–1252.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*. pages 625–630.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proc. of LREC 2016*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 529–533.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP 2018*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on CVPR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008.

S. Wu, D. Zhang, Z. Zhang, N. Yang, M. Li, and M. Zhou. 2018. Dependency-to-dependency neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26(11):2132–2141.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Detecting Toxicity in News Articles: Application to Bulgarian

Yoan Dinkov, Ivan Koychev
Faculty of Mathematics and Informatics
Sofia University
Sofia, Bulgaria
{jdinkov,koychev}@uni-sofia.bg

Preslav Nakov
Qatar Computing Research Institute
HBKU
Doha, Qatar
pnakov@hbku.edu.qa

Abstract

Online media aim for reaching ever bigger audience and for attracting ever longer attention span. This competition creates an environment that rewards sensational, fake, and toxic news. To help limit their spread and impact, we propose and develop a news toxicity detector that can recognize various types of toxic content. While previous research primarily focused on English, here we target Bulgarian. We created a new dataset by crawling a website that for five years has been collecting Bulgarian news articles that were manually categorized into eight toxicity groups. Then we trained a multi-class classifier with nine categories: eight toxic and one non-toxic. We experimented with different representations based on ElMo, BERT, and XLM, as well as with a variety of domain-specific features. Due to the small size of our dataset, we created a separate model for each feature type, and we ultimately combined these models into a meta-classifier. The evaluation results show an accuracy of 59.0% and a macro-F1 score of 39.7%, which represent sizable improvements over the majority-class baseline (Acc=30.3%, macro-F1=5.2%).

1 Introduction

The number of online news sources has grown dramatically in recent years, and so has the amount of news that has been bombarding users on a daily basis, especially in social media. As people have limited time to spend reading news, capturing people's attention is getting ever harder. Media have to use various techniques to get their readers back such as bigger advertisement and better services. Alternatively, it turns out that an easy way to attract people's attention is to use some toxicity in the articles, as people are intrigued by the unusual. Thus, our aim here is to try to detect articles that can harm, give false impressions, or deceive the readers. Such articles can use some of the following techniques:

- *Sensationalism*: overexposing insignificant or ordinary events by manipulating the main point of an article;
- *Fake news*: news that sound right, but totally misinterpret facts such as statistical data, locations, or dates, with the conscious aim of proving something wrong.
- *Conspiracy theories*: information that usually gives a lot of detail, but does not offer officially confirmed evidence or scientific research to back the claims that are being made. This is typically centered around political, or strange scientific phenomena.
- *Hate speech*: specifically targeting a person or a social group to brutalize them or to bully over the rate of normal conversation, to directly hurt or to manipulate them.

Given the proliferation of toxic news online, there have been many efforts to create tools and mechanisms to counteract their effect and spread. Such tools should help preserve and improve the reading integrity. Solving this problem is not a trivial task, and it requires a lot of effort by trained domain experts. Yet, there are limitations in how much it is possible to handle manually in a short period of time (and time is very critical as toxic content spreads fast). Thus, an attractive alternative is to use machine learning and natural language processing to automate the process of toxic news detection.

While most previous research has focused almost exclusively on English, here we target Bulgarian. In particular, we built a dataset for our experiments based on the knowledge base of *Media Scan*, which has catalogued and characterized many of the Bulgarian online media in the past five years. If a medium published a toxic news, this was recorded and the article, as well as the medium, got labelled accordingly. The analyzed media vary from digital newspapers, to media groups and blogs. For some articles there is detailed explanation with examples about why they were labelled like that. In some cases, the *Media Scan* website describes attempts to contact the authors of an article asking for clarification about some questionable facts that are being reported.

Here we use this information by performing multi-class classification over the *toxicity* labels: *fake news*, *sensations*, *hate speech*, *conspiracies*, *anti-democratic*, *pro-authoritarian*, *defamation*, *delusion*. Note that we allow multiple of these labels simultaneously. We further add a *non-toxic* label for articles that represent good news.

2 Related Work

The proliferation of false information has attracted a lot of research interest recently. This includes challenging the truthiness of news (Brill, 2001; Hardalov et al., 2016; Potthast et al., 2018), of news sources (Baly et al., 2018a, 2019; Dinkov et al., 2019), and of social media posts (Canini et al., 2011; Castillo et al., 2011), as well as studying credibility, influence, bias, and propaganda (Ba et al., 2016; Chen et al., 2013; Mihaylov et al., 2015a; Kulkarni et al., 2018; Baly et al., 2018a; Mihaylov et al., 2018; Barrón-Cedeño et al., 2019; Da San Martino et al., 2019; Zhang et al., 2019). Research was facilitated by shared tasks such as the SemEval 2017 and 2019 tasks on Rumor Detection (Derczynski et al., 2017; Gorrell et al., 2019), the CLEF 2018 and 2019 Check-That! labs (Nakov et al., 2018; Elsayed et al., 2019b,a), which featured tasks on automatic identification (Atanasova et al., 2018, 2019) and verification (Barrón-Cedeño et al., 2018; Hasanain et al., 2019) of claims in political debates, the FEVER 2018 and 2019 task on Fact Extraction and VERification (Thorne et al., 2018), and the SemEval 2019 task on Fact-Checking in Community Question Answering Forums (Mihaylova et al., 2019), among others.

The interested reader can learn more about "fake news" from the overview by Shu et al. (2017), which adopted a data mining perspective and focused on social media. Another recent survey (Thorne and Vlachos, 2018) took a fact-checking perspective on "fake news" and related problems. Yet another survey was performed by Li et al. (2016), and it covered truth discovery in general. Moreover, there were two recent articles in *Science*: Lazer et al. (2018) offered a general overview and discussion on the science of "fake news", while Vosoughi et al. (2018) focused on the proliferation of true and false news online.

The veracity of information has been studied at different levels: (*i*) claim (e.g., *fact-checking*), (*ii*) article (e.g., *"fake news" detection*), (*iii*) user (e.g., *hunting for trolls*), and (*iv*) medium (e.g., *source reliability estimation*). Our primary interest here is at the article-level.

2.1 Fact-Checking

At the claim-level, fact-checking and rumor detection have been primarily addressed using information extracted from social media, i.e., based on how users comment on the target claim (Canini et al., 2011; Castillo et al., 2011; Ma et al., 2016, 2017; Dungs et al., 2018; Kochkina et al., 2018). The Web has also been used as a source of information (Mukherjee and Weikum, 2015; Popat et al., 2016, 2017; Karadzhov et al., 2017b; Mihaylova et al., 2018; Baly et al., 2018b; Zlatkova et al., 2019).

In both cases, the most important information sources are *stance* (does a tweet or a news article agree or disagree with the claim?), and *source reliability* (do we trust the user who posted the tweet or the medium that published the news article?). Other important sources are linguistic expression, meta information, and temporal dynamics.

2.2 Stance Detection

Stance detection has been addressed as a task in its own right, where models have been developed based on data from the Fake News Challenge (Riedel et al., 2017; Thorne et al., 2017; Mohtarami et al., 2018; Hanselowski et al., 2018), or from SemEval-2017 Task 8 (Derczynski et al., 2017; Dungs et al., 2018). It has also been studied for other languages such as Arabic (Darwish et al., 2017b; Baly et al., 2018b; Mohtarami et al., 2019).

2.3 Source Reliability Estimation

Unlike stance detection, the problem of source reliability remains largely under-explored. In the case of social media, it concerns modeling the user[1] who posted a particular message/tweet, while in the case of the Web, it is about the trustworthiness of the source (the URL domain, the medium).

The source reliability of news media has often been estimated automatically based on the general stance of the target medium with respect to known manually fact-checked claims, without access to gold labels about the overall medium-level factuality of reporting (Mukherjee and Weikum, 2015; Popat et al., 2016, 2017, 2018). The assumption is that reliable media agree with true claims and disagree with false ones, while for unreliable media it is mostly the other way around. The trustworthiness of Web sources has also been studied from a Data Analytics perspective. For instance, Dong et al. (2015) proposed that a trustworthy source is one that contains very few false facts.

Note that estimating the reliability of a source is important not only when fact-checking a claim (Popat et al., 2017; Nguyen et al., 2018), but such reliability scores can be used as an important prior when addressing article-level factuality tasks such as "fake news" and click-bait detection (Brill, 2001; Hardalov et al., 2016; Karadzhov et al., 2017a; De Sarkar et al., 2018; Pérez-Rosas et al., 2018).

2.4 "Fake News" Detection

Most work on "fake news" detection has relied on medium-level labels, which were then assumed to hold for all articles from that source.

Horne and Adali (2017) analyzed three small datasets ranging from a couple of hundred to a few thousand articles from a couple of dozen sources, comparing (*i*) real news vs. (*ii*) "fake news" vs. (*iii*) satire, and found that the latter two have a lot in common across a number of dimensions. They designed a rich set of features that analyze the text of a news article, modeling its complexity, style, and psychological characteristics. They found that "fake news" pack a lot of information in the title (as the focus is on users who do not read beyond the title), and use shorter, simpler, and repetitive content in the body (as writing fake information takes a lot of effort). Thus, they argued that the title and the body should be analyzed separately.

In follow-up work, Horne et al. (2018b) created a large-scale dataset covering 136K articles from 92 sources from `opensources.co`, which they characterize based on 130 features from seven categories: structural, sentiment, engagement, topic-dependent, complexity, bias, and morality. We use this set of features when analyzing news articles.

In yet another follow-up work, Horne et al. (2018a) trained a classifier to predict whether a given news article is coming from a reliable or from an unreliable ("*fake news*" or *conspiracy*)[2] source. Note that they assumed that all news from a given website would share the same reliability class. Such an assumption is fine for training (distant supervision), but it is problematic for testing, where manual documents-level labels are needed.

Potthast et al. (2018) used 1,627 articles from nine sources, whose factuality has been manually verified by professional journalists from BuzzFeed. They applied stylometric analysis, which was originally designed for authorship verification, to predict factuality (fake vs. real).

Rashkin et al. (2017) focused on the language used by "fake news" and compared the prevalence of several features in articles coming from trusted sources vs. hoaxes vs. satire vs. propaganda. However, their linguistic analysis and their automatic classification were at the article level and they only covered eight news media sources.

2.5 Work for Bulgarian

We are aware of only one piece of previous work for Bulgarian that targets toxicity. In particular, (Karadzhov et al., 2017a) built a fake news and click-bait detector for Bulgarian based on data from a hackaton.

While most of the above research has focused on isolated and specific task (such as trustworthiness, fake news, fact-checking), here we try to create a holistic approach by exploring several toxic and non-toxic labels simultaneously.

[1] User modeling in social media and news community forums has focused on finding malicious users such as opinion manipulation *trolls*, paid (Mihaylov et al., 2015b) or just perceived (Mihaylov et al., 2015a; Mihaylov and Nakov, 2016; Mihaylov et al., 2018; Mihaylova et al., 2018), *sockpuppets* (Maity et al., 2017), *Internet water army* (Chen et al., 2013), and *seminar users* (Darwish et al., 2017a).

[2] We show in parentheses, the labels from `opensources.co` that are used to define a category.

Characteristic	Value
Toxic articles	221
Non-toxic articles	96
Media	164
Average title length (chars)	70.47
Average title length (words)	11.08
Average text length (chars)	3,613.70
Average text length (words)	556.83
Average text length (sentences)	31.64

Table 1: Statistics about the dataset.

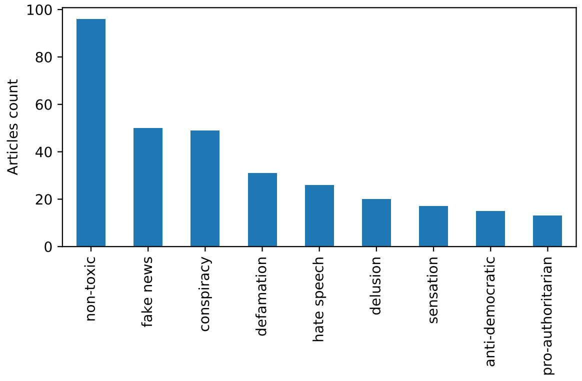


Figure 1: Label distribution in the dataset.

3 Data

We used *Media Scan*[3] as a source of toxicity labels for Bulgarian media Web sites. The site contains information about 700 media, and 150 of them are associated with at least one toxic article. Many of these toxic articles are removed after the respective media have been contacted and informed about the problems with these articles. Naturally, the author of *Media Scan* wanted to preserve the original form of the evaluated article, and thus had a link to a PDF copy in case the original HTML page was not accessible. We only crawled the HTML for articles that have not been changed or removed at the time we created the dataset.

For each Web page with a toxic label, we ran a mechanical crawler to obtain its contents. This was not very reliable as each individual medium site has its own structure, while the crawler expected more or less semantic and valid HTML to be able to process it. Thus, we manually verified the data, fixed any issues we could find and added any missing information. We ended up with a little over 200 articles with some kind of toxicity.

In addition to this dataset of only toxic articles, we added some “non-toxic” articles, fetched from media without toxicity examples in Media Scan: we added a total of 96 articles from 25 media.

Table 1 shows statistics about the dataset, and Figure 1 shows the distribution of the labels.

4 Method

We used a feature-rich classifier based on logistic regression and a neural network.

For each article, we extracted its title and its body. We further extracted some meta information about the corresponding news medium. As some NLP resources are only available or are better for English, we translated the articles to English, by using Google Translate API, so that we can extract features from them as explained in subsections 4.2, 4.6, 4.7, and 4.8.

4.1 LSA

We trained a Latent Semantic Analysis (LSA) model on our data. We first built TF.IDF vectors for the title and the body. Then, we applied singular value decomposition (SVD) to generate vectors of 15 dimensions for the titles and of 200 dimensions for the article bodies.

4.2 BERT

We used BERT (Devlin et al., 2019) for sentence representation, which has achieved very strong results on eleven natural language processing tasks including GLUE, MultiNLI, and SQuAD. Since then, it was used to improve over the state of the art for a number of NLP tasks. The original model was trained on English Wikipedia articles (2500M words). Due to the model complexity and to its size, it is hard to find enough data that represents a specific domain for a specific language.

We used BERT-as-a-service, which generates a vector of 768 numerical values for a given text. In its original form, this is a sentence representation tool, but we used it to generate text over the first 512 tokens of our article’s title or text. We used the multilingual cased pretrained model[4]. We experimented with all possible pooling strategies for representing the article title and its body, and we eventually chose the following pooling strategies: *REDUCE_MAX* for the title and *CLS_TOKEN* for the text of the article.

[3] http://mediascan.gadjokov.com

[4] http://github.com/google-research/bert#pre-trained-models

4.3 Stylometric Features

For the title and the body of each article, we calculate the following features:

- *avg_word_length_title*: average length of the word in the article title;
- *avg_word_length_text*: average length of the words in the article body
- *word_count_title*: number of words in the article title;
- *word_count_text*: number of words in the article body;
- *char_count_title*: number of characters in the article title;
- *char_count_text*: number of characters in the article body;
- *spec_char_count_title*: number of specific (non-alpha-numeric) characters in the article title;
- *spec_char_count_text*: number of specific (non-alpha-numeric) characters in the article body;
- *upper_char_count_title*: number of uppercase characters in the article title;
- *upper_char_count_text*: number of uppercase characters in the article body;
- *upper_word_count_title*: number of words starting with an uppercase character in the article title;
- *upper_word_count_text*: number of words starting with an uppercase character in the article body;
- *sentence_count_text*: number of sentences in the article;
- *avg_sentence_length_char_text*: average length of the sentences in the article body, in terms of characters;
- *avg_sentence_length_word_text*: average length of the sentences in the article body, in terms of words;

4.4 Media Features

We further extracted binary and numerical features characterizing the medium the article came from:

- *editor*: its value is 1 if the target medium has a designated chief editor, and it is 0 otherwise;
- *responsible_person*: its value is 1 if the target medium has a responsible person, and it is 0 otherwise;
- *bg_server*: its value is 1 if the target medium's location is in Bulgaria, and it is 0 otherwise;
- *popularity*: reciprocal value of the target medium's rank the Web traffic analysis platform Alexa[5];
- *domain_person*: its value is 1 if the target medium has a designated owner, and it is 0 otherwise;
- *days_existing*: number of days between when the medium was created and 01.01.2019. As this value is quite large, we take the logarithm thereof. For example, a medium created on Januarty 1, 2005 would have 5,113 days of existence, which would correspond to 3.70 as this feature's value.

4.5 XLM

We further used cross-lingual representations from the Facebook's XLM model (Lample and Conneau, 2019), which creates cross-lingual representations based on the Transformer, similarly to BERT. This model is pretrained for 15 languages including Bulgarian and English. We use their pretrained models, which were fine-tuned for Cross-lingual Natural Language Interence (XNLI) tasks. This yielded a 1024-demnsional representation for the title, and another one for the article body.

4.6 Universal Sentence Encoder

We also extracted representation using Google's Universal Sentence Encoder, or USE, (Cer et al., 2018). We used the pretrained model from TF Hub[6]. As the model is only available for English, we used the translations of the news articles. We passed the model the first 300 tokens for each title or body to generate 512-dimensional vectors.

[5] http://www.alexa.com/

[6] http://tfhub.dev/google/universal-sentence-encoder/2

4.7 ElMo

Next, we use deep contextualized word representations from ElMo, which uses generative bidirectional language model pre-training (Peters et al., 2018). The model yields 1024-dimensional representation, which we generate separately for the article title and for its body.

4.8 NELA Features

Finally, we use features from the NELA toolkit (Horne et al., 2018a), which were previously shown useful for detecting fake news, political bias, etc. The toolkit implements 129 features, which we extract separately for the article title and for its body:

- **Structure**: POS tags, linguistic features (function words, pronouns, etc.), and features for clickbait title classification from (Chakraborty et al., 2016);
- **Sentiment**: sentiment scores using lexicons (Recasens et al., 2013; Mitchell et al., 2013) and full systems (Hutto and Gilbert, 2014);
- **Topic**: lexicon features to differentiate between science topics and personal concerns;
- **Complexity**: type-token ratio, readability, number of cognitive process words (identifying discrepancy, insight, certainty, etc.);
- **Bias**: features modeling bias (Recasens et al., 2013; Mukherjee and Weikum, 2015) and subjectivity (Horne et al., 2017);
- **Morality**: features based on the Moral Foundation Theory (Graham et al., 2009) and lexicons (Lin et al., 2018)

A summary of all features is shown in Table 2.

Feature Group	Title	Body
BERT	768	768
ElMO	1,024	1,024
LSA	15	200
NELA	129	129
Stylometry	19	6
USE	512	512
XLM	1,024	1,024
Media	6	

Table 2: Summary of our features.

5 Experiments and Evaluation

5.1 Experimental Setup

We used logistic regression as our main classification method. As we have a small dataset, we performed 5-fold cross-validation. For evaluation, we used accuracy and macro-average F_1 score. The results are presented in Table 3.

Our baseline approach is based on selecting the most frequent class, i.e., *non-toxic*, which covers 30.30% of the dataset (see Table 1).

5.2 Individual Models

We evaluated a total of 14 setups for feature combination. Four of them represent features generated from the original article's title and body as well as a combination thereof. The next four setups present feature sets generated from the English translation as well as a combination thereof. The final section of Table 3 shows three setups that are somewhat language-independent: meta media, all features combined together as well as a meta classifier. We tuned the logistic regression for each individual experimental setup, using an additional internal cross-validation for the training part of each experiment in the 5-fold cross-validation. In total, 15,000 additional experiments have been conducted to complete the fine-tuning.

We can see in Table 3 that the BERT features (setups 2, 9) perform well both for English and for Bulgarian. The feature combinations (setups 6, 11, 13) do not yield good results as this increases the number of features, while the number of training examples remains limited. Using only the right 6 meta features about the target medium yields 12% improvement over the baseline. Interestingly, LSA turns out to be the best text representation model.

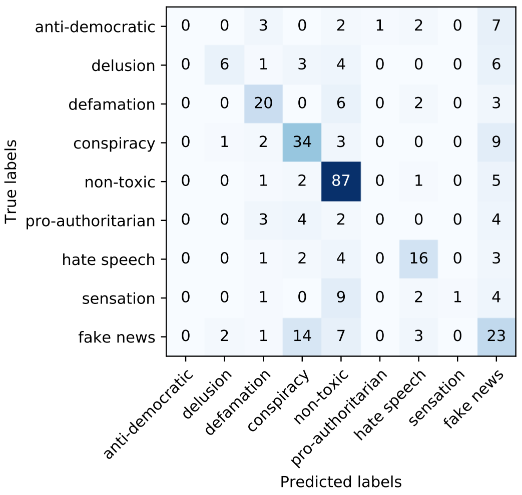
5.3 Meta Classifier

Next we tried a meta classifier. For this purpose, we extracted the posterior probabilities of the individual classifiers in Table 3 (2-5, 7-10, 12). Then, we trained a logistic classifier on these posteriors (we made sure that we do not leak information about the labels when training the meta classifier).

We can see in Table 3 that the meta classifier yielded the best results, outperforming all of the individual models, and achieving 3.5% absolute gain in terms of accuracy.

Language	N	Feature Set	Dimension	Accuracy	F1-macro
-	1	Baseline	-	30.30	05.17
BG	2	BERT(title), BERT(text)	1,536	47.69	32.58
	3	XLM(title), XLM(text)	2,048	38.50	24.58
	4	Styl(title), Styl(text)	15	31.89	08.51
	5	LSA(title), LSA(text)	215	55.59	42.11
	6	Bulgarian combined	3,824	39.43	24.38
EN	7	USE(title), USE(text)	1,024	53.70	40.68
	8	NELA(title), NELA(text)	258	36.36	23.04
	9	BERT(title), BERT(text)	1,536	52.05	39.78
	10	ElMO(title), ElMO(text)	2,048	54.60	40.95
	11	English combined	4,878	45.45	31.42
-	12	Media meta	6	42.04	15.64
	13	All combined	8,694	38.16	26.04
	14	**Meta classifier**	**153**	**59.06**	**39.70**

Table 3: Evaluation results.



True labels \ Predicted labels	anti-democratic	delusion	defamation	conspiracy	non-toxic	pro-authoritarian	hate speech	sensation	fake news
anti-democratic	0	0	3	0	2	1	2	0	7
delusion	0	6	1	3	4	0	0	0	6
defamation	0	0	20	0	6	0	2	0	3
conspiracy	0	1	2	34	3	0	0	0	9
non-toxic	0	0	1	2	87	0	1	0	5
pro-authoritarian	0	0	3	4	2	0	0	0	4
hate speech	0	0	1	2	4	0	16	0	3
sensation	0	0	1	0	9	0	2	1	4
fake news	0	2	1	14	7	0	3	0	23

Figure 2: Confusion matrix.

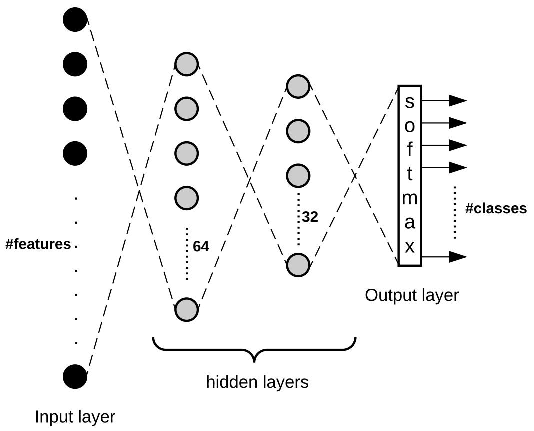


Figure 3: Our neural network.

5.4 Analysis

Figure 2 shows a confusion matrix over the entire dataset for the best model in Table 3, namely experiment 14. We can see that the model works best for the biggest *non-toxic* class. A decent chunk of *fake news* samples are misclassified as *conspiracy* as those two classes are the second and the third largest ones. For three of the labels, there are hardly any predictions; these are the smallest classes, and three of them combined cover less than 18% of the dataset.

5.5 What Did Not Work

Oversampling We evaluated all models from Table 3 but with oversampling the small classes. We tried simple random oversampling as well as the more complex SMOTE (Chawla et al., 2002) model. None of these techniques yielded improvements.

Neural Network We also tried a feed forward neural network with two hidden densely connected layers, with 64 nodes (ReLU activation) and 32 nodes (Tanh activation), and a dropout rate of 0.35 for each of them; see Figure 3. We used Adam for optimization, and we tried various parameter values, but we could not get improvements, possibly due to the small size of our dataset.

6 Discussion

Below we compare the performance of the models when working with Bulgarian vs. English resources, and we further discuss issues related to the small size of our dataset.

6.1 Language-Related Issues

The first part of the feature comparison in Table 3 is between English and Bulgarian, and it is interesting to compare them. The first comparison is between the BERT features. Even though we used a pre-trained BERT model, with same pooling techniques there is close to 4.5% absolute improvement when using the English translation. This is probably due to the English BERT being trained on more data as the English Wikipedia is much bigger for English: 5.7M English articles vs. just 250K Bulgarian articles.

Another notable comparison is between the types of models. Two of the Bulgarian feature sets are created via local models (experiments 4 and 5), while all of the English experiments are from transfer-learning. We can see that LSA (experiment 5) is the best feature set, and one can argue that this is to be expected. On such a small dataset in a non-English language, it is hard to represent the text with pre-trained models. Nevertheless, we can see that a combination between only pre-trained models (experiment 11) performs better compared to fusion between local and transfer-learning models (experiment 6).

6.2 Data Size Issues

There are several aspects of the above experiments where we can observe the negative effect of having insufficient data. First, in experiments 6, 11, 12 in Table 3, we can see that the combination of features performs worse in each language group compared to single feature types.

Another place where we felt we had insufficient data was in the neural network experiments, where we had many more parameters to train than in the simple logistic regression.

A related problem is that of class imbalance: we have seen in Section 5.4 above that the smallest classes were effectively ignored even by our best classifier. We can see in Figure 1 that those three classes have less than 60 articles combined, while the "non-toxic" only had 96 articles.

7 Conclusion and Future Work

We have presented experiments in detecting the toxicity of news articles. While previous research was mostly limited to English, here we focused on Bulgarian. We created a new dataset by crawling a website that has been collecting Bulgarian news articles and manually categorized them in eight toxicity groups. We then trained a multi-class classifier with nine categories: eight toxic and one non-toxic. We experimented with a variety of representations based on ElMo, BERT, xand XLM. We further used a variety of domain-specific features, which we eventually combined in a meta classifier. The evaluation results show an accuracy of 59.0% and a macro-F1 score of 39.7%, which represent sizable improvements over the majority-class baseline (Acc=30.3%, macro-F1=5.2%).

In future work, we plan to extend and also to balance the dataset. This can be achieved by either exploring another source for articles using the methodology of *Media Scan*, or by processing the unstructured PDF article, which we ignored in the present study. We also plan to explore new information sources. From a technical point of view, we would like to improve the neural network architecture as well as the oversampling techniques (with possible combination with undersampling).

Data and Code We are releasing all of the code for our experiments in a public repository that can be found in GitHub[7] with explanations about how to reproduce our environment. In that repository, we further release the full dataset together with the generated features, all the textual data, all the translations and all the meta data about the articles.

Acknowledgements

This research is part of the Tanbih project,[8] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute (QCRI), HBKU and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

This research is also partially supported by Project UNITe BG05M2OP001-1.001-0004 funded by the OP "Science and Education for Smart Growth" and the EU via the ESI Funds.

[7] github.com/yoandinkov/ranlp-2019

[8] http://tanbih.qcri.org/

References

Pepa Atanasova, Lluís Màrquez, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, Task 1: Check-worthiness. In *CLEF 2018 Working Notes*. CEUR-WS.org, Avignon, France.

Pepa Atanasova, Preslav Nakov, Georgi Karadzhov, Mitra Mohtarami, and Giovanni Da San Martino. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 1: Check-Worthiness. In *CLEF 2019 Working Notes*. CEUR-WS.org, Lugano, Switzerland.

Mouhamadou Lamine Ba, Laure Berti-Equille, Kushal Shah, and Hossam M. Hammady. 2016. VERA: A platform for veracity estimation over web data. In *Proceedings of the 25th International Conference Companion on World Wide Web*. Montréal, Québec, Canada, WWW '16, pages 159–162.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018a. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 3528–3539.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, NAACL-HLT '19, pages 2109–2116.

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018b. Integrating stance detection and fact checking in a unified corpus. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 21–27.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: Organizing the news based on their propagandistic content. *Information Processing & Management* 56(5):1849 – 1864.

Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, Task 2: Factuality. In *CLEF 2018 Working Notes*. CEUR-WS.org, Avignon, France.

Ann M Brill. 2001. Online journalists embrace new marketing function. *Newspaper Research Journal* 22(2):28–40.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and the IEEE International Conference on Social Computing*. Boston, MA, USA, SocialCom/PASSAT '11, pages 1–8.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*. Hyderabad, India, WWW '11, pages 675–684.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* .

Abhijnan Chakraborty, Bhargavi Paranjape, Kakarla Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. San Francisco, CA, USA, ASONAM '16, pages 9–16.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16:321–357.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the Internet Water Army: detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. Niagara, Canada, ASONAM '13, pages 116–120.

Giovanni Da San Martino, Seunghak Yu, Alberto Barron-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

Kareem Darwish, Dimitar Alexandrov, Preslav Nakov, and Yelena Mejova. 2017a. Seminar users in the Arabic Twitter sphere. In *Proceedings of the 9th International Conference on Social Informatics*. Oxford, UK, SocInfo '17, pages 91–108.

Kareem Darwish, Walid Magdy, and Tahar Zanouda. 2017b. Improved stance prediction in a user similarity feature space. In *Proceedings of the Conference on Advances in Social Networks Analysis and Mining*. Sydney, Australia, ASONAM '17, pages 145–148.

Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. 2018. Attending sentences to detect satirical fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3371–3380.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Vancouver, Canada, SemEval '17, pages 60–67.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, NAACL-HLT '2019, pages 4171–4186.

Yoan Dinkov, Ahmed Ali, Ivan Koychev, and Preslav Nakov. 2019. Predicting the leading political ideology of Youtube channels using acoustic, textual and metadata information. In *Proceedings of the 20th Annual Conference of the International Speech Communication Association*. Graz, Austria, INTERSPEECH '19.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.* 8(9):938–949.

Sebastian Dungs, Ahmet Aker, Norbert Fuhr, and Kalina Bontcheva. 2018. Can rumour stance alone predict veracity? In *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3360–3370.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Pepa Atanasova, and Giovanni Da San Martino. 2019a. CheckThat! at CLEF 2019: Automatic identification and verification of claims. In *Proceedings of the 41st European Conference on Information Retrieval*. Cologne, Germany, ECIR '19, pages 309–315.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019b. Overview of the CLEF-2019 CheckThat!: Automatic identification and verification of claims. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Springer, Lugano, Switzerland.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, MN, USA, SemEval '19, pages 845–854.

Jesse Graham, Jonathan Haidt, and Brian A Nosek. 2009. Liberals and conservatives rely on different sets of moral foundations. *Journal of personality and social psychology* 96(5):1029.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 1859–1874.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Varna, Bulgaria, AIMSA '16, pages 172–180.

Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 2: Evidence and Factuality. In *CLEF 2019 Working Notes*. CEUR-WS.org, Lugano, Switzerland.

Benjamin Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *CoRR* abs/1703.09398.

Benjamin Horne, Sibel Adali, and Sujoy Sikdar. 2017. Identifying the social signals that drive online discussions: A case study of Reddit communities. In *Proceedings of the 26th IEEE International Conference on Computer Communication and Networks*. Vancouver, Canada, ICCCN '17, pages 1–9.

Benjamin D. Horne, William Dron, Sara Khedr, and Sibel Adali. 2018a. Assessing the news landscape: A multi-module toolkit for evaluating the credibility of news. In *Proceedings of the The Web Conference*. Lyon, France, WWW '18, pages 235–238.

Benjamin D. Horne, Sara Khedr, and Sibel Adali. 2018b. Sampling the news producers: A large news and feature data set for the study of the complex media landscape. In *Proceedings of the Twelfth International Conference on Web and Social Media*. Stanford, CA, USA, ICWSM '18, pages 518–527.

Clayton Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media*. Ann Arbor, MI, USA, ICWSM '14.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017a. We built a fake news & click-bait filter: What happened next will blow your mind! In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Varna, Bulgaria, RANLP '17, pages 334–343.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017b. Fully automated fact checking using external sources. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Varna, Bulgaria, RANLP '17, pages 344–353.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. In *Proceedings of the International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3402–3413.

Vivek Kulkarni, Junting Ye, Steve Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 3518–3527.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science* 359(6380):1094–1096.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.* 17(2):1–16.

Y. Lin, J. Hoover, G. Portillo-Wightman, C. Park, M. Dehghani, and H. Ji. 2018. Acquiring background knowledge to improve moral value prediction. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. Los Alamitos, CA, USA, ASONAM '18, pages 552–559.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. New York, NY, USA, IJCAI '16, pages 3818–3824.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, ACL '17, pages 708–717.

Suman Kalyan Maity, Aishik Chakraborty, Pawan Goyal, and Animesh Mukherjee. 2017. Detection of sockpuppets in social media. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing*. Portland, OR, USA, CSCW '17, pages 243–246.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015a. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China, CoNLL '15, pages 310–314.

Todor Mihaylov, Ivan Koychev, Georgi Georgiev, and Preslav Nakov. 2015b. Exposing paid opinion manipulation trolls. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. Hissar, Bulgaria, RANLP '15, pages 443–450.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research* 28(5):1292–1312.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, ACL '16, pages 399–405.

Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, MN, USA, SemEval '19, pages 860–869.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, AAAI '18, pages 879–886.

Lewis Mitchell, Morgan R Frank, Kameron Decker Harris, Peter Sheridan Dodds, and Christopher M Danforth. 2013. The geography of happiness: Connecting Twitter sentiment and expression, demographics, and objective characteristics of place. *PloS one* 8(5):e64417.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 767–776.

Mitra Mohtarami, James Glass, and Preslav Nakov. 2019. Contrastive language adaptation for cross-lingual stance detection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. Melbourne, Australia, CIKM '15, pages 353–362.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims. In *Proceedings of CLEF*. Avignon, France, pages 372–387.

An T. Nguyen, Aditya Kharosekar, Matthew Lease, and Byron C. Wallace. 2018. An interpretable joint graphical model for fact-checking from crowds. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, AAAI '18, pages 1511–1518.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3391–3401.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 2227–2237.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. Indianapolis, IN, USA, CIKM '16, pages 2173–2178.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the Web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*. Perth, Australia, WWW '17, pages 1003–1012.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2018. CredEye: A credibility lens for analyzing and explaining misinformation. In *Proceedings of The Web Conference 2018*. Lyon, France, WWW '18, pages 155–158.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Melbourne, Australia, ACL '18, pages 231–240.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, EMNLP '17, pages 2931–2937.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, ACL '13, pages 1650–1659.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *ArXiv:1707.03264* .

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.* 19(1):22–36.

James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. 2017. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the EMNLP Workshop on Natural Language Processing meets Journalism*. Copenhagen, Denmark, pages 80–83.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3346–3359.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 809–819.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359(6380):1146–1151.

Yifan Zhang, Giovanni Da San Martino, Alberto Barrón-Cedeño, Salvatore Romeo, Jisun An, Haewoon Kwak, Todor Staykovski, Israa Jaradat, Georgi Karadzhov, Ramy Baly, Kareem Darwish, and Preslav Nakov James Glass. 2019. Tanbih: Get to know what you are reading. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

Dimitrina Zlatkova, Preslav Nakov, and Ivan Koychev. 2019. Fact-checking meets fauxtography: Verifying claims about images. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

De-Identification of Emails:
Pseudonymizing Privacy-Sensitive Data in a German Email Corpus

Elisabeth Eder **Ulrike Krieg-Holz**
Institut für Germanistik
Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
{elisabeth.eder | ulrike.krieg-holz}@aau.at

Udo Hahn
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Jena, Germany
udo.hahn@uni-jena.de

Abstract

We deal with the pseudonymization of those stretches of text in emails that might allow to identify real individual persons. This task is decomposed into two steps. First, named entities carrying privacy-sensitive information (e.g., names of persons, locations, phone numbers or dates) are identified, and, second, these privacy-bearing entities are replaced by synthetically generated surrogates (e.g., a person originally named *'John Doe'* is renamed as *'Bill Powers'*). We describe a system architecture for surrogate generation and evaluate our approach on CODEALLTAG, a German email corpus.

1 Introduction

With the advent and rapidly increasing adoption of electronic interaction platforms, the communication patterns of modern societies have changed fundamentally. We observe an unprecedented upsurge of digitally transmitted private communication and exploding volumes of so-called user-generated contents (UGC). As a major characteristic of these new communication habits, a sender's individual email, post, comment, tweet is distributed to an often (very) large number of addressees—the recipients of an email, other bloggers, friends or followers in social media platforms, etc.. Hence, hitherto private communication becomes intentionally public.

Responding to these changes, digital (social) media communication has become a focus of research in NLP. Yet there seems to be a lack of awareness among NLP researchers that the exploitation of natural language data from such electronic communication channels, whether for commercial, administrative or academic purposes, has to comply with binding legal regulations (Wilson et al., 2016). Dependent on each country's law system, different rules for privacy protection in raw text data are enforced (cf., e.g., two recent analyses for the US (Mulligan et al., 2019) and the EU (Hoofnagle et al., 2019)). Even privacy-breach incidents in a legal grey zone can be harmful for the actors involved (including NLP researchers).

This is evidenced dramatically in the so-called AOL search data leak.[1] In August of 2006, American Online (AOL) made a large query log collection freely accessible on the Internet for a limited time. The data were extracted over three months from their search engine to support academic research. The collection represented 650k users issuing 20 million queries *without* any significant anonymization. The result of this release, among others, was the disclosure of private information for a number of AOL users. The most troubling aspect of the data leak was the ease by which single unique individuals could be pinpointed in the logs. Even ignoring the existence of social security, drive license, and credit card numbers, the *New York Times* demonstrated the ability to determine the identity of a real user.[2] The outline of this incident and counter-measures against this privacy crash are reported by Adar (2007) from which we adopted the case description as well.

Despite this specific case, query logs from search engines might still be at the lower end of the vulnerability chain for data privacy, while UGC bundled in freely distributed corpora is clearly at its higher end, since clear names of persons, locations, etc. are dispersed all over such documents.

[1] Briefly described in https://en.wikipedia.org/wiki/AOL_search_data_leak, last accessed on July 24, 2019.

[2] https://www.nytimes.com/2006/08/09/technology/09aol.html, last accessed July 24, 2019.

Surprisingly, despite its high relevance for NLP operating on UGC, the topic of data privacy has long been neglected by the mainstream of NLP research. While it has always been of utmost importance for medical, i.e., clinical, NLP (Meystre, 2015), it has received almost no attention in NLP's non-medical camp for a long time (for two early exceptions, cf. Rock (2001); Medlock (2006)).

This naïve perspective is beginning to change these days with the ever-growing importance of social media documents for text analytics. However, there are currently no systematic actions taken to hide personally sensitive information from down-stream applications when dealing with chat, blog, SMS or email raw data. Since this attitude also faces legal implications, a quest for the protection of individual data privacy has been raised and, in the meantime, finds active response in the most recent work of the NLP community (Li et al., 2018; Coavoux et al., 2018).

We distinguish two basic approaches to eliminate privacy-bearing data from raw text data. The first one, *anonymization*, identifies instances of relevant privacy categories (e.g., person names or dates) and replaces sensitive strings by some artificial code (e.g., *'xxx'*). This blinding approach might be appropriate to eliminate privacy-bearing data in the medical world, but it is inappropriate for most NLP applications since crucial discriminative information and contextual clues will be erased by such a scrubbing procedure.

The second approach, *pseudonymization*, preserves such valuable information by replacing privacy-bearing text strings with randomly chosen alternative synthetic instances from the same privacy type (e.g., the person name *'Suzanne Walker'* is mapped to *'Caroline Snyder'*). As a common denominator, the term *de-identification* subsumes both, anonymization and pseudonymization.

The focus of this paper will be on pseudonymization and more precisely on the methods needed to produce realistic synthetic replacements, a process often also referred to as *surrogate generation*. We start with a discussion of related work in Section 2 and then introduce the semantic types we consider as relevant carriers of personal information in emails in Section 3. Next, we provide an overview of the email corpus our experiments are based on in Section 4, including manual annotation efforts. In Section 5, we turn to the process of surrogate generation, with focus on German language data. Since surrogate generation constitutes a highly constrained case of language generation, in Section 6 we describe the results of an evaluation study to assess the naturalness of these replacements with native speakers of German, as well as the performance of a recognizer for privacy-relevant text stretches on original and already pseudonymized data.

2 Related Work

The main thrust of work on de-identification has been performed for clinical NLP.[3] Most influential for progress in this field have been two challenge competitions within the context of the I2B2 (Informatics for Integrating Biology & the Bedside) initiative[4] which focused on 18 different types of Protected Health Information (PHI) categories as required by US legislation (HIPAA).[5] The first of these challenge tasks was launched in 2006 for 889 hospital discharge summaries, with a total of 19,498 PHI instances of person-identifying verbal expressions (Uzuner et al., 2007). The second was run in 2014 and addressed an even broader set of PHI categories (Stubbs et al., 2015a). In summary, the best system performances peaked in the high 90s (F_1 score) using classical machine learning methods, Conditional Random Fields (CRFs) in particular, and hand-written rules, or a mixture of both. As a successor to I2B2, the CEGS-NGRID Shared Tasks and Workshop on Challenges in NLP for Clinical Data created a corpus of 1,000 manually de-identified psychiatric evaluation records (Stubbs et al., 2017). Interestingly, for the automatic de-identification task performance values dropped significantly down to 79.85 F_1 for the best-performing system indicating an only modest potential for domain and text genre portability (moving from discharge summaries to psychiatric evaluation records).

Recently, the deep learning wave has also hit the (clinical) de-identification community. For this task, bidirectional Long-Short Term Memory Networks (Bi-LSTMs) became quite popular as evidenced by the work of Dernoncourt et al. (2017)

[3] Note that we have to distinguish between data protection in structured data contained in (clinical) information systems, (for which k-anonymity (Sweeney, 2002) is a well-known model to minimize a person's re-identification risk) and pseudonym-based textual variant generation for unstructured verbal data we here focus on.

[4] https://www.i2b2.org/

[5] https://www.hhs.gov/hipaa/for-professionals/privacy/index.html

who achieve an F_1 score of 97.85 on the I2B2 2014 dataset, or Liu et al. (2017) who report performance figures ranging from 95.11% over 96.98% up to 98.28% micro F_1 score under increasingly sloppier matching criteria on the same dataset.

Note that these challenges were focusing on the *recognition* of privacy-relevant text stretches textual but did not incorporate pseudonymization, a more complex task (Stubbs et al., 2015b). Carrell et al. (2013) deal with the latter challenge within the context of the 'Hiding In Plain Sight' approach where the detected privacy-bearing identifiers are replaced with realistic synthetic surrogates in order to collectively render the few 'leaked' identifiers difficult to distinguish from the synthetic surrogates—a major advantage for pseudonymization over anonymization. Targeting English medical texts SCRUB (Sweeney, 1996) is one of the first surrogate generation systems followed by work from Uzuner et al. (2007), Yeniterzi et al. (2010), Deléger et al. (2014), Stubbs et al. (2015b) and Stubbs and Uzuner (2015). Similar procedures have been proposed for Swedish (Alfalahi et al., 2012) and Danish (Pantazos et al., 2011) clinical corpora, yet not for German ones.

Work outside the clinical domain is rare. While we found no work dealing with the anonymization or even pseudonymization of emails and Twitter-style social media data,[6] anonymizing SMSes is a topic of active research. Patel et al. (2013) introduce a system capable of anonymizing SMS (Short Message Service) communication. Their study builds on 90,000 authentic French text messages and uses dictionaries as well as decision trees as machine learning technique. Their evaluation task is, however, very coarse-grained—select those SMSes from a test corpus of 23,055 messages that either have or have not to be anonymized. There is no breakdown to PHI-like categories known from the medical domain.

Treurniet et al. (2012) were taking care of privacy-relevant data for a Dutch SMS corpus (52,913 messages, in total) in much more detail. They automatically anonymized all occurrrences of dates, times, decimal amounts, and numbers with more than one digit (telephone numbers, bank accounts, etc.), e-mail addresses, URLs, and IP addresses. All sensitive information was replaced with corresponding semantic placeholder codes of the encountered semantic *type* (e.g., each specific email address was replaced by the type symbol `EMAIL`), not by an alternative semantic *token*, i.e., a pseudonym. The same strategy was also chosen by Chen and Kan (2013) for their SMS corpus that contains more than 71,000 messages, focusing on English and Mandarin. However, neither are the methods of automatic anonymization described in detail, nor are performance figures of this process reported in both papers (Chen and Kan (2013) only mention the use of regular expressions for the anonymization process).

In conclusion, pseudonymization has to the best of our knowledge only been seriously applied to medical documents, up until now. Hence, our investigation opens this study field for the first time ever to non-medical applications of pseudonymization. Such de-identified corpora can then easily be distributed via public sites and so might stimulate further NLP research.

3 Named Entities for De-Identification

Perhaps the most relevant source and starting point for determining types of personally identifying information pieces in written documents is a catalogue of *Personal Health Information* (PHI) items that has been derived from the *Health Information Privacy Act* (HIPAA) which is binding law in the US. PHI enumerates altogether 18 privacy-sensitive items organized into eight main categories (Stubbs and Uzuner, 2015):

- *Name* includes the names of patients, doctors and user names,
- *Profession* of persons mentioned,
- *Location* includes rooms, clinical departments, hospital names, names of organizations, street names, city names, state names, names of countries, ZIPs, etc.,
- *Age* of persons,
- *Date* expressions,
- *Communication* data, e.g., phone or fax numbers, email addresses, URLs, IP addresses,
- all sorts of *IDs* such as Social Security number, medical record number, health plan number, account number, license number, vehicle ID, device ID, biometric ID, etc.,
- any *Other* form of personally sensitive data.

While some types of categories from above are generally useful also for non-medical anonymization procedures, others are quite domain-specific,

[6] Lüngen et al. (2017) report on *manual* anonymization efforts for German chat data.

because they are intrinsically attached to the clinical domain (such as the names of patients, doctors or nurses, the names of hospitals and their departments, or various forms of IDs, e.g., health insurance numbers). Hence, we adapted this list for email documents while, at the same time, we tried to avoid over-fitting to this text genre.

We, finally, came up with the category set depicted in Figure 1 which we stipulate to universally account for all types of emails, irrespective of any particular natural language and email (header) encoding. The categories are organized in a concise hierarchy whose top level categories are *SocialActor (ACTOR)*, *Date (DATE)*, *FormalIdentifier (FID)*, *Location (LOC)*, and *Address (ADD)*. We anticipate that this hierarchy can be further refined and accomodated to other text genres as well.

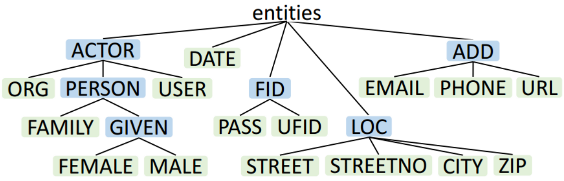


Figure 1: Hierarchy of privacy-bearing entity types (pis) relevant for emails (leaves in green)

The category of *SocialActor* can be further divided into *Organization (ORG)*, which includes all types of legal actors such as companies, brands, institutions and agencies, etc., human *Persons (PERSON)*, with subtypes *FamilyName (FAMILY)*, also including initials and credentials, and *GivenName (GIVEN)*, with another split into two subcategories, namely *FemaleName (FEMALE)* and *MaleName (MALE)*, both including nicknames and initials. Finally, *UserName (USER)* covers all kinds of invented user names for IT systems and platforms.

Date (DATE) includes all sorts of date descriptions, such as date of birth, marriage, or death, starting and ending dates of contracts, etc.

The category of *FormalIdentifier (FID)* includes *Password (PASS)* as user-provided supplementary artificial name for all kinds of technical appliances, and *UniqueFormalIdentifier (UFID)* to capture persons (students, customers, employees, members of social security systems (SSN), authors (ORCHID), etc.), computer systems (IP addresses), or other artifacts (e.g., IBANs, DOIs).

The *Location* (LOC) category subsumes *StreetName (STREET)*, *StreetNumber (STREETNO)*, *ZipCode (ZIP)*, and *CityName (CITY)* which stands for villages, towns, cities, larger metropolitan areas (e.g., *'Larger Manchester'*) and regions smaller than a state (e.g., *'Bay Area'*); it also includes derivations of these names (e g., *'Roman'*).

Finally, *Address (ADD)* encompasses *EmailAddress (EMAIL)*, *PhoneNumber (PHONE)*, including fax numbers, and *URL (URL)*, as well as other forms of domain names.

Unlike some approaches from the field of clinical NLP (Stubbs et al., 2015b), we did not take ages or professions into account, because our use case is not that sensible and ages or professions probably are mentioned far more often in clinical reports than in emails. Furthermore, unspecific dates like *'Christmas'* or *'next week'* and geographical information such as landmarks, rivers or lakes were not tagged for de-identification since their contribution to possible re-identification is fairly limited due to their generality.

4 Email Corpus and Entity Annotation

Our experiments are based on 1,390 German emails from the CODE ALLTAG$_{S+d}$ corpus (Krieg-Holz et al., 2016), which were collected on the basis of voluntary email donation. The donors have provided their explicit consent that, *after de-identification*, their emails may be made publically available. Sharing the corpus would create lots of opportunities for NLP research, since public access to private emails is generally forbidden.[7]

For the manual annotation campaign,[8] we set up a team of three annotators who tagged equally sized parts of the corpus, according to the privacy-bearing (pi) categories described in Section 3 (Figure 1). Annotation was performed on entity level. Therefore, we did not have to care about token boundaries in the surrogate generation step and, thus, no special handling for compounds and multi-token entities is required.

In order to measure the inter-annotator agreement (IAA), the annotators worked on 50 identical emails randomly selected from the corpus within the same annotation phase as the entire corpus.

[7] One of the rare exceptions is the ENRON corpus (Klimt and Yang, 2004) whose non-anonymized contents was released for open inspection by order of US judges in the course of the destruction of the Enron company as a consequence of criminal financial transactions of the Enron management. For yet another example, cf. the *Avocado Research Email Collection*, available from LDC2015T03.

[8] We used BRAT (`http://brat.nlplab.org/`) for annotation (Stenetorp et al., 2012).

Table 1 shows Cohen's Kappa (Cohen, 1960) as a measure for IAA for the pairs of annotators calculated on the entities represented by the BIO annotation scheme.[9] Hence, not only the token label itself but also matching starting and ending points of an entity are taken into account, as well. The agreement is quite high, especially between annotator 1 and 3.

A1 - A2	A2 - A3	A3 - A1
0.925	0.933	0.958

Table 1: Cohen's Kappa for BIO tags on CODE ALLTAG$_{S+d}$

Based on these 50 emails the annotators also examined and discussed differences of their annotations and decided on the gold standard by majority vote, which we applied for further evaluation in order to measure precision (Prec), recall (Rec) and F$_1$ score (F$_1$). Table 2 shows the outcomes regarding BIO tags per annotator and the overall result calculated over the joint annotations of the annotators. We took the averages from the outcomes of the single categories weighted by the number of true instances for each label.

	Prec	Rec	F$_1$
A1	98.06	95.63	96.72
A2	87.67	77.92	80.36
A3	94.14	84.79	86.82
A1+A2+A3	93.37	86.11	88.66

Table 2: Weighted average of precision, recall and F$_1$ score with respect to the gold standard for BIO tags of CODE ALLTAG$_{S+d}$

An error analysis revealed that, besides mostly accidental errors, a higher disagreement on tagging ORGs (due to overlap or confusion with city or product names and rather generic organizations)[10] and an uncertainty regarding DATEs could be observed. The latter problem was solved by the decision to treat all dates as *pi* regardless of their specificity. As a consequence, one annotator worked through the entire corpus to re-tag each DATE and, if necessary, also re-tag ORGs according to the findings of the error analysis. The outcome of this overhaul constituted the final gold standard annotations of CODE ALLTAG$_{S+d}$ for the de-identification task.

5 Surrogate Generation

Once *pi*-relevant named entities have been recognized they undergo a replacement process where original identifiers are substituted by synthetic, though natural, surrogates. For this step, we distinguish language-independent criteria from those which are intrinsically language-specific. Only the latter have to be re-specified for languages other than German.

5.1 Language-Independent Criteria

Personal information belonging to the categories ADD (EMAIL, PHONE or fax number, URL), FID (PASSword, UFID), USER name and ZIP code are relatively simple to replace. Each digit of the string is substituted with a randomly generated alternative digit, each alphabetic character is replaced with a randomly generated alternative letter of the same case and alphabet. Other characters like '*@*' or punctuation marks are left as is. For URLs, we also keep the subdomain *'www'* and commonly used URL schemes like *'http'*, *'https'*, *'ftp'*, *'file'* and *'mailto'*. In contrast to Stubbs et al. (2015b) we did not implement any other restrictions for the selection of characters. As a consequence, the resulting surrogates may have an unrealistic appearance.

To maintain temporal ordering in the document we generate a time shift separately for each text to make re-identification difficult, if not impossible. We shifted the dates by a random interval between 365 days forward or backward. As we try to keep the original language-dependent formats, the user has to determine the formats to be generated.

In order to maintain coreferences between *pi* entities, we replace multiple occurrences of an entity by the same surrogate. To account for different spellings of names regarding lower and upper case we treat different possibilities of combinations, the original spelling, lower case, upper case and a normalized format which is language-specific. We decided not to consider misspellings, because checking for slightly different names, e.g., employing the Levenshtein distance, could also lead to coreference breaks since quite a few names differ only in one letter (like *'Lena'* and *'Lina'*).

[9] 'B' preceding a token's tag stands for the Beginning of an entity, 'I' for its continuation (Inside), and 'O' for any stretch of text not belonging to an entity (Outside).

[10] Stubbs and Uzuner (2015) also report confusions of organizations with other subcategories from their location class (department, hospital) and Stubbs et al. (2017) witness an uncertainty for tagging quasi-generic organizations.

For resolving initials and abbreviations of GIVEN, FAMILY and CITY names, we adopt the approach by Stubbs et al. (2015b) and use letter-to-letter mappings generated for each document. This means that each name of the respective category starting with a certain character is replaced with a name with its first letter corresponding to the mapping of this character or in case of initials and abbreviations with the mapping solely. For example *'Gandalf'* and *'Gimli'* would be substituted by names starting with the same character like *'Bilbo'* and *'Boromir'*. Hence, an initial *'G.'*, will also be replaced with *'B.'*; it does not have to be disambiguated and assigned to any of the previously occurring names (a task left to a coreference resolution module we currently do not provide).

We also try to account for frequency distributions of GIVEN, FAMILY, CITY, ORG and STREET names by constraining these random letter-to-letter-mappings to map first letters to letters with a similar frequency. In this way, mappings of very common first letters, such as *'A'* in case of German first names, to rare ones, like *'X'*, can be avoided. This approach still allows rare substitutes for common names. However, we circumvent the problem of adding ambiguity to the text, if we only have few names starting with *'X'* (Stubbs et al. (2015b) also mentioned but did not implement this idea). As we map distributions in quite a rough way, we do not think that this could cause a leak of information of the original text, but distributional mappings are optional and the user may choose the granularity and distribution in his or her own module or language extension.

5.2 Language-Dependent Criteria for German

Since the categories DATE, STREET, CITY, GIVEN and FAMILY name, and ORG are affected by language-specific influences we implemented a German language module for these named entity types. In contrast to other surrogate generation systems we know of, our solution takes inflection into account (relying on the NLP tool SPACY (Honnibal and Montani, 2017)).

Dates. The formats we handle include typical combinations of day, month and year according to German formatting style (e.g., days precede months), yet also account for different spellings (e.g., *'01.06.2019'* or *'1.6.19'*). Days, months, and years occurring in isolation are processed as well. If a month is given in letter format, it is substituted with an alternative month name trying to keep differences between standard varieties (e.g., *'Januar'* (standard German) vs. *'Jänner'* (Austrian German) for *'January'*) and also preserve abbreviations (e.g., *'Jan.'*).

Street names. For names of streets, we recognize the abbreviations *'S|-str(.)'* (for *'Straße'* (street)) and *'P|-pl(.)'* (for *'Platz'* (place)) for look-up and coreference. We do not handle inflections of street names like *'Ligusterweg(e)s'* (genitive) because, in emails, street names most often are part of an address and thus lack inflection.

The list of surrogates is built from large repositories of German street names[11] jointly with Austrian ones from different provinces[12] that do not contain special characters or are composed of more than two terms (e.g., *'Albert-Einstein-Straße'*). Furthermore, we restricted them to contain only names with standard street suffixes (*'-straße'*, *'-weg'*, *'-platz'*, etc.), because we had to get rid of village names that do not have any named streets (such as with *'Wegscheid 15'*).

Given Names and Family Names. Common German proper nouns are singular and do not change number. (Duden, 2009, p. 191) Hence, our system does not process any plural forms of forenames or surnames (which rarely may occur), yet handles the genitive singular case. Therefore, our system is also capable of resolving coreferences between uninflected and inflected genitive forms.

To acquire suitable look-up dictionaries we extracted female and male names with their associated nicknames from a list of first names.[13] These lists are restricted to more or less common forenames in German-speaking countries except for names with rare first letters, where we included less frequent names, too. An alternative list of German surnames[14] is frequency-independent, thus includes also lots of uncommon names.

City names. For the CITY category, the genitive singular case is handled too. While person names and city, town or village names are mostly

[11] http://www.datendieter.de/item/Liste_von_deutschen_Strassennamen_.csv

[12] http://www.statistik.at/strasse/suchmaske.jsp

[13] ftp://ftp.heise.de/pub/ct/listings/0717-182.zip

[14] http://www.namenforschung.net/fileadmin/user_upload/dfa/Inhaltsverzeichnisse_etc/Index_Band_I-V_Gesamt_Stand_September_2016.pdf

used without determiner, a few German names for regions (e.g., *'die Steiermark'*, *'das Drautal'*) always require a determiner (Duden, 2009, pp. 299ff.). These names can be of every gender and also pluralia tantum, whereas city, town or village names are neuter and singularia tantum (Duden, 2009, pp. 160f.). Unfortunately, we currently do not dispose of repositories for gender- and number-specific CITY names large enough, so we rather tolerate possible mistakes than a potential information leakage.

In contrast to person names, we consider derivations of locations and implemented rules for signifying inhabitants (*'die Klagenfurterin'*) and adjectivized toponyms ending on *'-(i)sch/-erisch/-er'* (*'kärntnerisch'* (carinthian), *'Wiener Dialekt'* (Viennese dialect)). We check for coreferences using Levenshtein distance on previously seen CITYs. To catch lemmata occurring later we form appropriate candidates in a rule-based manner together with a lexicon look-up. For naming inhabitants, we only treat the standard forms ending on *'-er'* such as *'Wiener'* (Viennese) and do not care about non-standard names like *'Hesse'* (Hessian).

The generation of derivations from the substitute lemma is restricted to the most common cases. We produce derivatives by concatenating the lemma and *'-er'* or *'-r'*, if the lemma ends with *'-e'*. For adjectivized forms ending with *'-isch/-sch/-erisch'* that often allow a variation of these suffixes (Duden, 2016, p. 685) we decided on generating derivatives with *'-erisch'* (e.g., *'Wienerisch'* (Viennese)). Our system produces the inflectional forms for derivations by copying the original inflectional suffix to the generated form.

To maintain local national information we employed separate lists of location names for the three major German-speaking countries[15] on which we perform a dictionary look-up to determine which country the location name is from. Admittedly, this approach fails, if the place is either not mentioned or occurs in multiple countries. For substitution, we provide cleaner lists containing only villages, towns and cities for Germany,[16] Austria,[17] and Switzerland.[18]

Organizations. Similarly, we consider the genitive case for organizations. As the same company or institution in a document might be denoted by different name forms, such as its full name (*'Stadtwerke GmbH'*), with or without the corporate form (*'Stadtwerke'*) or an acronym (*'STW'*), with respect to coreference chains a more sophisticated solution is required. For now, we only check for names without a list of corporate forms. The substitution is performed with a list of German company names,[19] restricted to names not containing any GIVEN or FAMILY name. Due to gender variability and the lack of a list of institutions, we here added fictional acronyms and randomly generated letter combinations.

5.3 System Architecture

Our system for surrogate generation (see Figure 2) accepts any type of text, not only emails, but requires BRAT annotations of pi-relevant entities as described in Section 3. It allows for an easy adaptation to languages other than German, since the base module of the surrogate generation system can implement a language module for alternative languages, too. While language-independent categories (Section 5.1) do not need any further consideration, this language module has to provide allowed date formats and lists with language- and category-appropriate substitutes for the language-dependent classes (Section 5.2). Furthermore, frequency mappings of first letters may be specified in order to take distributions of names with respect

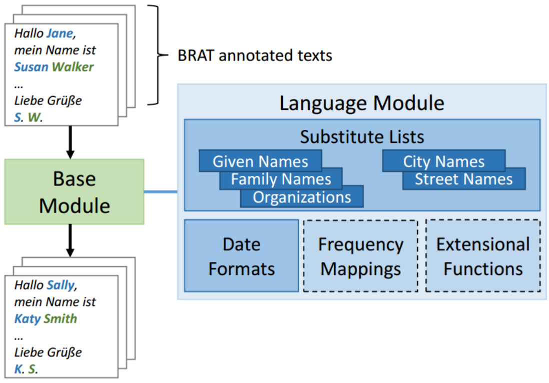


Figure 2: Schematic system architecture and surrogate generation workflow; dashed parts optional

[15] http://download.geonames.org/export/dump

[16] http://www.fa-technik.adfc.de/code/opengeodb/PLZ.tab

[17] http://www.statistik.at/strasse/suchmaske.jsp

[18] http://data.geo.admin.ch/ch.swisstopo-vd.ortschaftenverzeichnis_plz/PLZO_CSV_LV03.zip

[19] https://www.datendieter.de/item/Liste_von_deutschen_Firmennamen_.txt extracted from OpenStreetMap (http://www.openstreetmap.org/)

to their first letters into account. If a use case requires a special treatment of a category, extensional functions have to be defined, e.g., for inflection generation. Otherwise, the base system replaces the entities with default entries in the substitute lists, i.e., generally (non-inflected) lemmata.

6 Evaluation of Pseudonymization

6.1 Grammaticality and Acceptability Tests

In a first round of evaluations, we wanted to test whether the surrogates we had generated were well-formed in terms of grammaticality and semantically 'natural' in terms of acceptability. For privacy reasons, we refrained from explicitly evaluating whether coreference relations were preserved, because it is difficult to keep track of them without the original wording.

First, the pseudonymized emails were scored on both evaluation dimensions on a scale from 1 (worst) to 5 (best) in packages of about 30 emails by different annotators. Each email was annotated only once. While *grammaticality* refers to the agreement in number, gender and case between the generated surrogate and the sentence constituents the surrogate is embedded in, a surrogate is *acceptable* if it semantically fits into the surrounding context so that a reasonable semantic interpretation can be made. For example, *'We bought the car at Amici Pizza Express'* is considered as semantically inacceptable because cars normally cannot be bought at a pizzeria. In contrast common names replaced with rare ones (e.g., *'Paris'* with a small village name) are regarded as acceptable.

For the evaluation, the manually tagged 1,390 German-language emails of CODE ALLTAG$_{S+d}$ underwent the surrogate generation process twice to be sure not to reveal any original pi items to the annotators. The automatically generated output was checked manually, also substituting phrases not covered by our categories, such as course names (the corpus contains lots of university related emails donated by our students). After that it was fed again to the surrogate generation system.

With 4.90 for grammaticality and 4.73 for semantic acceptability the results are pretty sound. The lower outcome for semantic acceptability is mostly due to the surrogates for the ORG category. Occasionally, rather odd combination like *'Serial Knitters, IT solutions'* (for *'Institute for XX, YY-University'*) and substitutes with a different, inaccurate function were found.

6.2 Frequency Analysis

As Deléger et al. (2014) remark, large frequency imbalances between corpora may influence the performance of machine learning systems. Therefore, we also assessed frequency imbalances between different corpora: The original non-pseudonymized CODE ALLTAG$_{S+d}$ corpus with hand-annotated pi entities (referred to as ORIG), the pseudonymized[20] form of the original corpus (PSEUD) and pseudonymized[20] CODE ALLTAG$_{S+d}$ with automatically recognized pi entities (PSEUDPIR). For the latter, we retrieved the pi entities by training a model on $9/10$ of ORIG and applying it to the unseen part on ten different folds. We used a system for recognizing privacy-bearing information (PIR) based on NEURONER (Dernoncourt et al., 2017; Lee et al., 2016), with slight modifications of its neural network architecture from our side.

Table 3 shows that the number of tokens declines for both pseudonymized corpora compared to the original corpus. Taking a closer look, the category discrepancy almost entirely results from the category ORG and, to a much lesser extent though, also from STREET and CITY names. We conclude that our substitution dictionaries obviously contain shorter names, i.e., entities consisting of fewer tokens. Contrasting ORIG and the automatically annotated PSEUDPIR, we witness an increase of pi entities. For one thing, this can be explained by taking the tokens of one entity separately and thus splitting one single entity into multiple ones, which is also reflected in the smaller difference between the number of tokens. As this phenomenon especially occurs with URLs and PHONE or fax numbers (these categories are substituted randomly) it has no effect on surrogate generation. Again, ORGanizations play a major role here, because the PIR system achieves the worst results for this category.

	ORIG	PSEUD	PSEUDPIR
# token	151,229	150,166	150,425
# types[21]	21,159	22,320	22,455
# pi entities	8,866	8,866	9,427
# pi tokens	12,649	11,586	11,865

Table 3: Quantitative breakdown of the corpora used for evaluation; "#" stands for frequency count

[20] Processed by the surrogate generation system without any further reworking.

6.3 Automatic Recognition Performance

Further, we followed Yeniterzi et al. (2010) and Deléger et al. (2014) and tested the performance of the PIR system on the three corpora. Like Yeniterzi et al. (2010), we also found better results for training and testing on the PSEUD corpus than on ORIG,[22] while the performance difference decreases for PSEUDPIR and ORIG (see Table 4). Among others, this may be a consequence of the frequency imbalance, because the pseudonymized data contain fewer tokens especially those related to the hard to recognize ORGanizations.

Corpus	Prec	Rec	F_1	p-value
ORIG	83.02	82.26	82.52	
PSEUD	86.15	86.24	86.07	0.007
PSEUDPIR	82.26	85.79	83.86	0.063

Table 4: Results of the PIR system on different corpora (10-fold cross validation); significance difference (p-value) with respect to ORIG (paired t-test)

For training and testing on different corpora, we eliminated the emails found in the test set from the train set in a 10-fold cross-validation manner, even if they were pseudonymized in one of the datasets to avoid any overlap. Again, similar to Yeniterzi et al. (2010), training on ORIG and testing on PSEUD yields significantly better results than the other way round (see Table 5). Also the F_1 score plunges deeply when training on PSEUDPIR and testing on ORIG compared to ORIG and testing on PSEUDPIR.

If we subsume the language-independent categories, because they are randomized and treated similarly in surrogate generation, we get comparable outcomes for all experiments with the PIR system with an F_1 score around 90.00. In contrast, language-dependent categories, with the exception of DATEs, which accomplish equivalent results for nearly all experiments, too, consistently perform worse compared to training and testing on the same corpus, especially regarding ORGanizations, CITYs and STREETs. As the results in Table 3 reveal, the amount of word types is higher in the pseudonymized corpora; hence, they are more diverse. Further, they may contain rarer names. Both phenomena potentially lead to a decrease of performance on these data sets when trained on the ORIG corpus. Regarding the drop of the reverse experiment the fewer occurrences of 'I'-tags (from the BIO format) have an impact, too.

<table>
<tr><th>Train|Test</th><th>Prec</th><th>Rec</th><th>F_1</th><th>p</th></tr>
<tr><td>ORIG|PSEUD</td><td>77.97</td><td>73.12</td><td>74.93</td><td></td></tr>
<tr><td>PSEUD|ORIG</td><td>70.25</td><td>61.32</td><td>64.57</td><td>0.0</td></tr>
<tr><td>ORIG|PSEUDPIR</td><td>85.23</td><td>75.21</td><td>78.39</td><td></td></tr>
<tr><td>PSEUDPIR|ORIG</td><td>67.31</td><td>63.21</td><td>63.88</td><td>0.0</td></tr>
</table>

Table 5: Results of the PIR system trained on Train and tested on Test (10-fold cross validation without overlap); significance difference (p-value) over reverse setting (paired t-test)

In contrast to Yeniterzi et al. (2010) and our results, Deléger et al. (2014) report a smaller performance difference between training on original and testing on pseudonymized data and vice versa. This is probably due to the fact that they replaced pi entities with different entities of the same category taken from the same original corpus, thus almost retaining the original personal information. This approach bears the potential of causing a leak of personal information due to categories with limited occurrences.

7 Conclusion

In this paper, we moved the de-identification problem out of the medical domain (cf. also our work in this field described in Kolditz et al. (2019)) into the realm of user-generated content, emails in our use case. In particular, we focused on the surrogate generation step of that task, i.e., substituting named entities bearing privacy-relevant information by synthetic, yet mostly natural surrogates.

Our main contributions are the specification of a language-independent type hierarchy composed of named entities that carry privacy-relevant information, and the realization of the first German non-medical surrogate generation pipeline. It is composed of a language-dependent part for German input and a language-independent one which can readily be reused for languages other than German, without any changes.

We also ran a series of experiments on emails from the German-language CODE ALLTAG$_{S+d}$ corpus. In this evaluation of our surrogate generation system, we found high scores for the grammaticality and acceptability of the automatically

[21] Types of tokens excluding punctuation and stop words.

[22] When the k-fold splits are performed on sentences rather than on emails the results get notably better achieving 86.24 F_1 score for ORIG. But for reasons of comparability between non-pseudonymized and pseudonymized corpora we had to split on emails.

generated surrogates. A frequency analysis of different variants of CODE ALLTAG$_{S+d}$ revealed a quantitative imbalance between the original corpus, the pseudonymized one, and a de-identified variant that was built using an automatic recognizer for privacy-relevant named entities. Experiments on these three corpora further exposed differences in recognition performance already discussed in the literature.

Our future work will focus on a more adequate treatment of German derivational morphology and coreferences rooted in varying spellings. The main methodological desideratum concerns the investigation of ways to deal with organizations with different functions in order to improve semantic acceptability. Last but not least, we will have to demonstrate that the results from the small-scale corpus we currently dealt with (CODE ALLTAG$_{S+d}$) will scale up to much larger document collections (e.g., CODE ALLTAG$_{XL}$ as described in Krieg-Holz et al. (2016)).

Acknowledgments

We want to thank our RANLP reviewers for pointing out improvements of the original submission. These hints were gratefully acknowledged and incorporated in the final version of the paper.

This work was funded by the Austrian HRSM project *Kompetenznetzwerk Digitale Edition* (KONDE).

References

Eytan Adar. 2007. User 4XXXXX9: anonymizing query logs. In *Proceedings of the Workshop on Query Log Analysis: Social and Technological Challenges @ WWW 2007. Banff, Alberta, Canada, May 8, 2007*.

Alyaa Alfalahi, Sara Brissman, and Hercules Dalianis. 2012. Pseudonymisation of personal names and other PHIs in an annotated clinical Swedish corpus. In *BioTxtM 2012 — Proceedings of the 3rd Workshop on Building and Evaluating Resources for Biomedical Text Mining @ LREC 2012. Istanbul, Turkey, May 26, 2012*. pages 49–54.

David S. Carrell, Bradley A. Malin, John S. Aberdeen, Samuel Bayer, Cheryl Clark, Benjamin Wellner, and Lynette Hirschman. 2013. Hiding In Plain Sight: use of realistic surrogates to reduce exposure of protected health information in clinical text. *Journal of the American Medical Informatics Association* 20(2):342–348.

Tao Chen and Min-Yen Kan. 2013. Creating a live, public short message service corpus: the NUS SMS corpus. *Language Resources and Evaluation* 47(2):299–335.

Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. 2018. Privacy-preserving neural representations of text. In *EMNLP 2018 — Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium, October 31 - November 4, 2018*. pages 1–10.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Louise Deléger, Todd Lingren, Yizhao Ni, Megan Kaiser, Laura Stoutenborough, Keith Marsolo, Michal Kouril, Katalin Molnar, and Imre Solti. 2014. Preparing an annotated gold standard corpus to share with extramural investigators for de-identification research. *Journal of Biomedical Informatics* 50:173–183.

Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. 2017. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association* 24(3):596–606.

Duden. 2009. *Die Grammatik: Unentbehrlich für richtiges Deutsch*, volume 4 of *Der Duden in 12 Bänden*. Dudenverlag, Mannheim etc., 8th edition.

Duden. 2016. *Das Wörterbuch der sprachlichen Zweifelsfälle: Richtiges und gutes Deutsch*, volume 9 of *Der Duden in 12 Bänden*. Dudenverlag, Berlin, 8th edition.

Matthew Honnibal and Ines Montani. 2017. SPACY 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io/.

Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. 2019. The European Union general data protection regulation: what it is and what it means. *Information & Communications Technology Law* 28(1):65–98.

Bryan Klimt and Yiming Yang. 2004. The ENRON corpus: a new dataset for email classification research. In *ECML 2004 – Proceedings of the 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004*. Springer, pages 217–226.

Tobias Kolditz, Christina Lohr, Johannes Hellrich, Luise Modersohn, Boris Betz, Michael Kiehntopf, and Udo Hahn. 2019. Annotating German clinical documents for de-identification. In *MedInfo 2019 — Proceedings of the 17th World Congress on Medical and Health Informatics. Lyon, France, 25-30 August 2019*. IOS Press.

Ulrike Krieg-Holz, Christian Schuschnig, Franz Matthies, Benjamin Redling, and Udo Hahn. 2016. CODE ALLTAG: a German-language e-mail corpus.

In *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation. Portorož, Slovenia, 23-28 May 2016*. pages 2543–2550.

Ji Young Lee, Franck Dernoncourt, Özlem Uzuner, and Peter Szolovits. 2016. Feature-augmented neural networks for patient note de-identification. In *ClinicalNLP 2016 — Proceedings of the Clinical Natural Language Processing Workshop @ COLING 2016. Osaka, Japan, December 11, 2016*. pages 17–22.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. Towards robust and privacy-preserving text representations. In *ACL 2018 — Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics. Melbourne, Victoria, Australia, July 15-20, 2018*. volume 2, pages 25–30.

Zengjian Liu, Buzhou Tang, Xiaolong Wang, and Qingcai Chen. 2017. De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of Biomedical Informatics* 75(Supplement):S34–S42.

Harald Lüngen, Michael Beißwenger, Laura Herzberg, and Cathrin Pichler. 2017. Anonymisation of the Dortmund Chat Corpus 2.1. In *cmccorpora17 — Proceedings of the 5th Conference on CMC and Social Media Corpora for the Humanities. Bolzano, Italy, October 3-4, 2017*. pages 21–24.

Ben Medlock. 2006. An introduction to NLP-based textual anonymisation. In *LREC 2006 — Proceedings of the 5th International Conference on Language Resources and Evaluation. Genoa, Italy, 22-28 May, 2006*. pages 1051–1056.

Stéphane M. Meystre. 2015. De-identification of unstructured clinical data for patient privacy protection. In Aris Gkoulalas-Divanis and Grigorios Loukides, editors, *Medical Data Privacy Handbook*, Springer International Publishing, pages 697–716.

Stephen P. Mulligan, Wilson C. Freeman, and Chris D. Linebaugh. 2019. Data protection law: an overview. Technical Report CRS Report R45631, Congressional Research Service.

Kostas Pantazos, Sören Lauesen, and Sören Lippert. 2011. De-identifying an EHR database: anonymity, correctness and readability of the medical record. In *MIE 2011 — Proc. of the 23rd Conference of the European Federation of Medical Informatics. Oslo, Norway, August 28-31, 2011*. pages 862–866.

Namrata Patel, Pierre Accorsi, Diana Z. Inkpen, Cédric Lopez, and Mathieu Roche. 2013. Approaches of anonymisation of an SMS corpus. In *CICLing 2013 — Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing. Karlovasi, Samos, Greece, March 24-30, 2013*. Springer, pages 77–88.

Frances Rock. 2001. Policy and practice in the anonymisation of linguistic data. *International Journal of Corpus Linguistics* 6(1):1–26.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a Web-based tool for NLP-assisted text annotation. In *EACL 2012 — Proc. of the 13th Conf. of the European Chapter of the Association for Computational Linguistics: Demonstrations. Avignon, France, April 25-26, 2012*. pages 102–107.

Amber Stubbs, Michele Filannino, and Özlem Uzuner. 2017. De-identification of psychiatric intake records: overview of 2016 CEGS NGRID Shared Tasks Track 1. *Journal of Biomedical Informatics* 75(Supplement):S4–S18.

Amber Stubbs, Christopher Kotfila, and Özlem Uzuner. 2015a. Automated systems for the de-identification of longitudinal clinical narratives: overview of 2014 i2b2/UTHealth Shared Task Track 1. *Journal of Biomedical Informatics* 58(Supplement):S11–S19.

Amber Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: the 2014 i2b2/UTHealth corpus. *Journal of Biomedical Informatics* 58(Supplement):S20–S29.

Amber Stubbs, Özlem Uzuner, Christopher Kotfila, Ira Goldstein, and Peter Szolovits. 2015b. Challenges in synthesizing surrogate PHI in narrative EMRs. In Aris Gkoulalas-Divanis and Grigorios Loukides, editors, *Medical Data Privacy Handbook*, Springer International Publishing, pages 717–735.

Latanya Sweeney. 1996. Replacing personally-identifying information in medical records, the SCRUB system. In *AMIA '96 — Proceedings of the 1996 AMIA Annual Fall Symposium. Washington, D.C., USA, October 26-30, 1996*. pages 333–337.

Latanya Sweeney. 2002. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems* 10(5):557–570.

Maaske Treurniet, Orphée De Clercq, Henk van den Heuvel, and Nelleke Oostdijk. 2012. Collecting a corpus of Dutch SMS. In *LREC 2012 — Proceedings of the 8th International Conference on Language Resources and Evaluation. Istanbul, Turkey, May 21-27, 2012*. pages 2268–2273.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. 2007. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association* 14(5):550–563.

Shomir Wilson et al. 2016. The creation and analysis of a website privacy policy corpus. In *ACL 2016 — Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin, Germany, August 7-12, 2016*. pages 1330–1340.

Reyyan Yeniterzi, John S. Aberdeen, Samuel Bayer, Benjamin Wellner, Lynette Hirschman, and Bradley A. Malin. 2010. Effects of personal identifier resynthesis on clinical text de-identification. *Journal of the American Medical Informatics Association* 17(2):159–168.

Lexical Quantile-Based Text Complexity Measure

Maksim Eremeev
AITHEA
m.eremeev@aithea.com

Konstantin Vorontsov
National University of Science
and Technology MISIS
voron@aithea.com

Abstract

This paper introduces a new approach to estimating the text document complexity. Common readability indices are based on average length of sentences and words. In contrast to these methods, we propose to count the number of rare words occurring abnormally often in the document. We use the reference corpus of texts and the quantile approach in order to determine what words are rare, and what frequencies are abnormal. We construct a general text complexity model, which can be adjusted for the specific task, and introduce two special models. The experimental design is based on a set of thematically similar pairs of Wikipedia articles, labeled using crowdsourcing. The experiments demonstrate the competitiveness of the proposed approach.

1 Introduction

Automated text complexity measurement tools have been proposed in order to help teachers to select textbooks that correspond to the students' comprehension level and publishers to explore whether their articles are readable. Thus, plenty of readability indexes were developed. Measures like *Automated Readability Index* (Senter and Smith, 1967), *Flesch-Kincaid readability tests* (Flesh, 1951), SMOG index (McLaughlin, 1969), Gunning fog (Gunning, 1952) and etc. use heuristics based on simple statistics such as total number of words, mean number of words per sentence, total number of sentences or even number of syllables to evaluate how complex given text is. By combining these statistics with different weighting factors, readability indexes assign the given document a *complexity score*, which is, in most cases, the approximate representation of the US grade level needed to comprehend the text. For instance, an Automated Readability Index (ARI) has the following form for the document d:

$$ARI(d) = 4.71 \times \frac{c}{w} + 0.5 \times \frac{w}{s} - 21.43 \quad (1)$$

where c refers to the total number of letters in the document d, w is the total number of words and s denotes the total number of sentences in d.

Since readability indexes rely on a few basic factors, precise assessment requires aggregation of many scores. Thus, Coh-Metrix-PORT tool (Aluisio et al., 2010) includes more than 50 different indexes for Portuguese language. The tool is based on Coh-Metrix (Graesser et al., 2004) principles to estimate complexity and cohesion not only for explicit text, but for the mental representation of the document.

Readability indexes are interpretable and easy to implement. However, the great number of constants tuned specifically for the English language texts, lack of the semantics consideration and tailoring to the US grade level system restrains the number of possible applications.

As for the non-English languages, several lexical and morphological features for Italian to solve text simplification problem were presented (Brunato et al., 2015), supervised approach in readability estimations was introduced (vor der Brck et al., 2008) and the complexity estimations for legal documents in Russian were explored (Dzmitryieva, 2017).

In this paper we introduce a new approach to gauge the complexity of the documents based on their lexical features. Our research is motivated by information retrieval applications such as exploratory search for learning or editorial purposes (Marchionini, 2006; White and Roth, 2009; Palagi et al., 2017). In the exploratory search, the user needs a hint which of the found documents to read first, gradually moving from simple to more complex documents. Reading order optimization is an alternative way to content consumption that departs from the typical ranked lists of documents based on their relevance (Koutrika et al., 2015). The more specific terms document contains, and the more rare they are, the more complex the document is. To formalize this consideration, we estimate the complexity of each term in the document and then aggregate them to get the complete document complexity score. We use Wikipedia as a *reference collection* of moderately complex texts in order to determine what term frequencies are abnormal.

In section 2 we describe quantile approach to estimate the single term complexity. We present highly flexible general model in section 3 and models in subsections 3.1 and 3.2. The way of evaluating the proposed methods is introduced in section 4 and the experiments result are provided in section 5.

2 Single Term Complexity Estimation

Reference collection: Let D denote a reference collection. Let document $d \in D$ consist of terms $t_1, t_2, \dots t_{n_d}$, where n_d refers to the length of document d. Each term can be either a single word or a key phrase.

Quantile approach: In general case each term can occur in different complexity states, which may depend on a position in text or context surrounding the term. Each complexity state of the term t_i standing in position i is described with a *term complexity score* $c(t_i)$. Consider a complexity scores empirical distribution for each term over the reference collection. Assume that term t_i is *in complex state* if its complexity $c(t_i)$ in current text position i is greater than γ-quantile $C_\gamma(t_i)$ of the distribution over $c(t_i)$, where γ is a hyperparameter, responsible for the complexity level. Therefore, when estimating complexity score of the document, we count $c(t_i)$ only for terms t_i which are in the complex state, defined by the γ parameter.

For instance, $c(t_i)$ can be a constant, which means all terms have identical complexity, or can be set equal to 0 if it occurs in the reference collection and 1 otherwise. In this case, we count new terms (for the reference collection) as complex and all other terms as simple.

3 General Document Complexity Model

Document d complexity $W(d)$ can be calculated by aggregating complexity scores of terms that form d. In this paper we propose a weighed sum over the complex terms to be the aggregate function.

$$W(d) = \sum_{i=1}^{n_d} w(t_i)[c(t_i) > C_\gamma(t_i)] \qquad (2)$$

where $[\,]$ refers to the Iverson notation (i.e. $[true] = 1, [false] = 0$).

By defining weights $w(t_i)$ and complexity scores $c(t_i)$ for all terms t_i specialize the complexity model.

Some examples of interpretable weights $w(t_i)$ are presented in Table 1.

$w(t_i)$	**Meaning of** $w(t_i)$
1	number of complex terms
$1/n_d \times 100\%$	complex terms percentage
$c(t_i)$	total complexity
$c(t_i)/n_d$	mean complexity
$c(t_i) - C_\gamma(t_i)$	excessive complexity
$(c(t_i) - C_\gamma(t_i))/n_d$	mean excessive complexity

Table 1: Weights $w(t_i)$ examples.

3.1 Distance-Based Complexity Model

The following model relies on the assumption, proposed in (Birkin, 2007). Consider an arbitrary document d which is the sequence of terms $t_1, t_2, \dots t_{n_d}$. Let $r(t_i)$ be a distance in terms to the previous occurrence of the same term t_i in document d. Formally,

$$r(t_i) = \min_{1 \le j < i} \{i - j \mid t_i = t_j\}. \qquad (3)$$

If i is the first occurrence of term t_i in document d, it means that $r(t_i)$ is undefined. In such cases we take $r(t_i)$ equal to n_d. Hence, for terms with the only occurrence in d complexity scores are the greatest.

If term t does not appear in the reference collection, we set C_γ equal to $-\infty$, therefore counting it as a constantly complex term.

Assume that term t in the position i is more complex than the same term in the position j if $r(t_i) > r(t_j)$. Consider there are no separators between documents in the reference collection, so it becomes a single document d_{all}. Thus, it is possible to count distributions of $r(t)$ of each unique term t in d_{all} and corresponding γ-quantiles $C_\gamma(t)$ of these distributions.

For the document d, which complexity we try to estimate, we calculate $r_d(t_i)$ values for all terms $t_i \in d$.

We define mean distance $r_{d,i}(t_i)$ for term t_i in i-th position in the document d as

$$\bar{r}_{d,i}(t_i) = \frac{\sum_{j=1}^{i} r_d(t_i)[t_i = t_j]}{\sum_{j=1}^{i}[t_i = t_j]} \tag{4}$$

which aggregates all occurrences of the term t_i from the document start.

Finally $c(t_i)$ has the form:

$$c(t_i) = \bar{r}(t_i) - \bar{r}_{d,i}(t_i) \tag{5}$$

where $\bar{r}(t_i)$ is the mean distance of the reference collection scores $r(t_i)$ for the term t_i.

Intuitively, this means, that term is more complex if it occurs less in reference collection and occurs more in document d.

Figures 1 and 2 show distributions of distances $r(t)$ for the simple term 'algebra' and the complex term 'nlp', calculated over the reference collection containing 1.5M documents of the Russian Wikipedia. For the 'algebra' term most occurrences are relatively close to each other, whether 'nlp' occurrences have fairly greater distance scores.

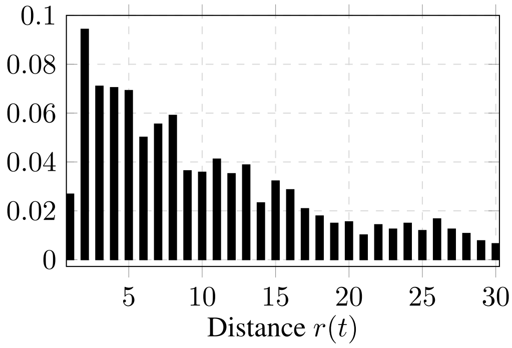


Figure 1: Distribution of distances $r(t)$, calculated over the complete Wikipedia dataset for the word 'algebra'.

So, using the formula for $c(t_i)$ as above and choosing weights $w(t_i)$ we get the distance-based complexity model.

3.2 Counter-Based Complexity Model

The second model presented in this paper is based on the assumption that each term has an independent fixed complexity in the whole language. Thus, in this section we consider not the complexity distribution of a single term, but the general complexity distribution over all terms in the language. Hence, each term t is assigned the only

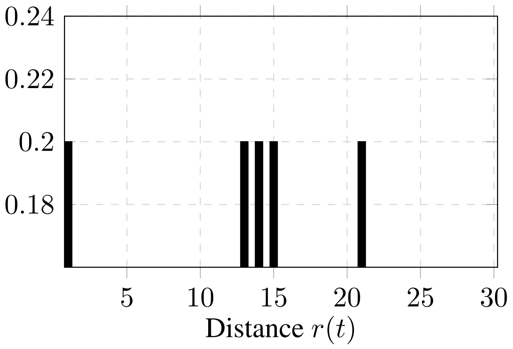


Figure 2: Distribution of distances $r(t)$, calculated over the complete Wikipedia dataset for the word 'nlp'.

complexity score $c(t)$ and the γ-quantile we count is now a constant C_γ.

Hence, the model has the following form:

$$W(d) = \sum_{i=1}^{n_d} w(t_i) \left[\frac{1}{count(t_i)} > C_\gamma\right] \tag{6}$$

where $w(t_i)$ corresponds to the term weights introduced before.

Assume the term t_1 is more complex than the term t_2 if number of occurrences in the reference collection of the term t_1 is lesser than the number of occurrences of the term t_2.

Let $count(t)$ denote number of occurrences of the term t in the reference collection. Thus, the complexity score function can be defined as

$$c(t) = \frac{1}{count(t)} \tag{7}$$

so the assumption above is satisfied.

For each term t we calculate counters $count(t)$ and complexity scores $c(t)$ over the reference collection. Having the distribution of $c(t)$, we obtain γ-quantiles C_γ. The described distribution for the Russian Wikipedia reference collection is shown on Figure 3.

Thus, we have defined $c(t)$ for all terms possible and the distribution necessary to count the C_γ. By varying weights $w(t_i)$ described in section 3, we obtain the counter-based model for the complexity estimation.

4 Quality Metric

To measure the quality of proposed algorithms, we asked assessors to label 10K pairs of Russian Wikipedia articles. Assessors were asked to carefully read both articles and to choose which was

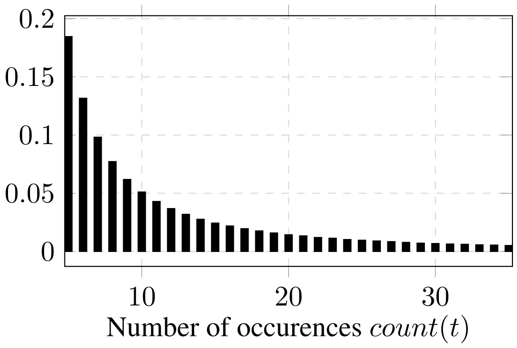


Figure 3: Distribution of $count(t)$, calculated over complete Wikipedia articles dataset.

more difficult to comprehend. If person cannot determine which document is more complex, then he was asked to choose 'documents are equal' option. If documents in the given pair are from different scientific domains, then we ask assessor to choose 'invalid pair' option.

Documents were chosen from math, physics, chemistry and programming areas. Clustering was performed using the topic modeling technique (Hofmann, 1999). BigARTM open-source library was used to perform the clustering (Vorontsov et al., 2015). Pairs were formed so that both documents belong to a single topic and their lengths are almost identical. Examples of document pairs to assess are introduced in Table 2.

Document 1	**Document 2**	**Result**
Matrix	Tensor	RIGHT
Neural network	Linear regression	LEFT
Electric charge	Molecule	EQUAL
Mac OS X	Convex Hull	INVALID

Table 2: Examples of labeled document pairs.

Each pair was labeled twice in order to avoid human factor mistakes. We assume that the pair was labeled correctly if labels were not controversial, i.e. first assessor labeled the first document as more complex, while second assessor chose the second document. If one or both grades were 'documents are equal' then we assume the pair to be correctly labeled.

8K pairs out of 10K were labeled correctly and were used to compare for the different versions of algorithms. For each we calculated the accuracy score, which is the rate of correctly chosen document in the pair.

5 Experiments

Two types of experiments were done. In first case we used full Russian Wikipedia articles dataset (1.5M documents) as a reference collection. In second type we used only Wikipedia articles from the math domain. To do that, we built a topic model using ARTM (Additive Regularization of Topic Models) technique (Vorontsov and Potapenko, 2015), which clusters documents into monothematic groups.

5.1 Complete Wikipedia Dataset

Preprocessing: All Wikipedia articles were lemmatized (i.e. reduced to normal form). In this experiment we assume term to be either a single word or a bigram (i.e. two words combination). To extract them, RAKE algorithm (Rose et al., 2010) was used. Hence, each document in the collection was turned into the sequence of such terms.

Reference collection: Preprocessed Wikipedia articles were used as a reference collection. $r(t)$ for every term position and $count(t)$ for every unique term were counted.

Documents to estimate complexity on: We used the labeled pairs described in Section 4 to evaluate the models. Accuracy was used as a quality metric.

Models to evaluate: Models introduced in 3.1 and 3.2 with different $w(t_i)$ parameters were tested. We took ARI and Flesch-Kincaid readability test as benchmarks.

The results of the experiments are introduced in Table 3. Also we tested how the bigrams extraction affects final quality with fixed weight function $w(t) = c(t)/n_d$. The results are given in Table 4.

Model	$w(t)$	**Accuracy**
ARI	-	**46%**
Flesch-Kincaid	-	**57%**
Distance-based	$c(t)$	**68%**
Distance-based	$c(t)/n_d$	**71%**
Counter-based	$c(t)$	**77%**
Counter-based	$c(t)/n_d$	**81%**

Table 3: Results of experiment 1 with different weight function.

Results show that both distance- and counter-based approaches work twice as well as readability indexes. Counter-based model with $w(t) = c(t)/n_d$ weights show the best results.

Model	Terms	Accuracy
Distance-based	Words	**63%**
Distance-based	Words+Bigrams	**71%**
Counter-based	Words+Bigrams	**74%**
Counter-based	Bigrams	**81%**

Table 4: Results of experiments 1 with terms differently defined.

5.2 Single Topic Wikipedia Dataset

In experiment 2 we shortened the reference collection to include only documents from specific topic.

ARTM model: To divide documents into single-topic clusters, topic modeling is used. Topic Models are unsupervised machine learning models and perform soft clustering (i.e. assign each document a distribution over topics). The set of such vectors for all documents form a matrix, which is usually denoted by Θ. *ARTM model* was trained on the preprocessed Wikipedia dataset. ARTM features dozens of various types of regularizers and allows to treat modalities (i.e. types of terms) differently.

In this specific experiment we used regularizers to sparse Θ matrix and make each topic distribution over terms more different. Words and bigrams (i.e. pairs of words) modalities were used with weights 1 and 5 respectively. Using this model, we detect the most likely topic for each document.

Experiment setup: In the following experiment we chose math and physics documents to be the reference collection. Documents were preprocessed in the same way as they were in the previous experiment. We also divided labeled pairs into same single-topic groups to test models configured with different reference collections on various single-topic groups of labeled pairs.

Math collection included 200K documents in reference collection and 3.5K labeled pairs, while for the physics collection it was 250K documents in reference collection and 1.5K labels. The results are shown in Table 5 and Table 6.

As it can be seen from results, using tailored reference collection improves the score. Indeed, that solves terms ambiguity problem and eliminates terms unrelated to the topic from the reference collection, so they are treated complex in the estimating document, which is fairly logical.

Model	$w(t)$	Accuracy
ARI	-	**41%**
Flesch-Kincaid	-	**49%**
Distance-based	$c(t)$	**55%**
Distance-based	$c(t)/n_d$	**61%**
Counter-based	$c(t)$	**79%**
Counter-based	$c(t)/n_d$	**84%**

Table 5: Results of experiment 2 on math collection of Wikipedia articles with different weights.

Model	$w(t)$	Accuracy
ARI	-	**52%**
Flesch-Kincaid	-	**58%**
Distance-based	$c(t)$	**65%**
Distance-based	$c(t)/n_d$	**63%**
Counter-based	$c(t)$	**82%**
Counter-based	$c(t)/n_d$	**81%**

Table 6: Results of experiment 2 on physics collection of Wikipedia articles with different weights.

6 Conclusions

We have presented an approach to estimating text complexity based on lexical features. Document complexity is an aggregation of terms' complexities. Introduced general model is highly flexible, it can be adjusted by tuning weights $w(t)$ and choosing proper reference collection.

Complexity score can only be count with respect to the reference collection. Reference collection can be a large set of documents on different topics or just contain single-topic texts.

The proposed complexity measures are used in AITHEA exploratory search system (http://aithea.com/exploratory-search) for ranking search results in complexity-based reading order.

Acknowledgements

Application of topic modeling in this research was supported by the Russian Research Foundation grant no. 19-11-00281. The work of K.Vorontsov was partially supported by the Government of the Russian Federation (agreement 05.Y09.21.0018) and the Russian Foundation for Basic Research (grants 17-07-01536).

References

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability assessment for text simplification.

A.A. Birkin. 2007. *Speech Codes*. Hippocrat, Saint-Peterburg.

Dominique Brunato, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. 2015. Design and annotation of the first italian corpus for text simplification. pages 31–41.

Tim vor der Brck, Sven Hartrumpf, and Hermann Helbig. 2008. A readability checker with supervised learning using deep syntactic and semantic indicators.

Aryna Dzmitryieva. 2017. The art of legal writing: A quantitative analysis of russian constitutional court rulings. *Sravnitel'noe konstitucionnoe obozrenie*, 3:125–133.

R. Flesh. 1951. How to test readability. *New York, Harper and Brothers*.

Arthur Graesser, Danielle McNamara, Max Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, computers : a journal of the Psychonomic Society, Inc*, 36:193–202.

Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill, New York.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA. ACM.

Georgia Koutrika, Lei Liu, and Steven Simske. 2015. Generating reading orders over document collections. In *2015 IEEE 31st International Conference on Data Engineering*, pages 507–518.

Gary Marchionini. 2006. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46.

G. H. McLaughlin. 1969. Smog grading: A new readability formula. *Journal of Reading*, 12(8):639–646.

Emilie Palagi, Fabien Gandon, Alain Giboin, and Raphaël Troncy. 2017. A survey of definitions and models of exploratory search. In *ESIDA17 - ACM Workshop on Exploratory Search and Interactive Data Analytics, Mar 2017, Limassol, Cyprus*, pages 3–8.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. *Automatic Keyword Extraction from Individual Documents*.

R.J. Senter and E.A. Smith. 1967. Automated readability index. *AMRL-TR*, 66(22).

K. V. Vorontsov and A. A. Potapenko. 2015. Additive regularization of topic models. *Machine Learning, Special Issue on Data Analysis and Intelligent Optimization with Applications*, 101(1):303–323.

Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Petr Romov, and Marina Suvorova. 2015. Bigartm: Open source library for regularized multimodal topic modeling of large collections. In *AIST'2015, Analysis of Images, Social networks and Texts*, pages 370–384. Springer International Publishing Switzerland, Communications in Computer and Information Science (CCIS).

Ryen W. White and Resa A. Roth. 2009. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan and Claypool Publishers.

Demo Application for LETO: Learning Engine Through Ontologies

Suilan Estevez-Velarde[1], **Andrés Montoyo**[2], **Yudivián Almeida-Cruz**[1],
Yoan Gutiérrez[3], **Alejandro Piad-Morffis**[1], and **Rafael Muñoz**[2]

[1]School of Math and Computer Science, University of Havana, Cuba
{sestevez,yudy,apiad}@matcom.uh.cu
[2]Department of Languages and Computing Systems, University of Alicante, Spain
[3]U.I. for Computer Research (IUII), University of Alicante, Spain
{montoyo,ygutierrez,rafael}@dlsi.ua.es

Abstract

The massive amount of multi-formatted information available on the Web necessitates the design of software systems that leverage this information to obtain knowledge that is valid and useful. The main challenge is to discover relevant information and continuously update, enrich and integrate knowledge from various sources of structured and unstructured data. This paper presents the Learning Engine Through Ontologies (LETO) framework, an architecture for the continuous and incremental discovery of knowledge from multiple sources of unstructured and structured data. We justify the main design decision behind LETO's architecture and evaluate the framework's feasibility using the Internet Movie Data Base (IMDB) and Twitter as a practical application.

1 Introduction

In recent years, research in machine learning, knowledge discovery, data mining and natural language processing, among others, have produced many approaches and techniques to deal with the large amount of information available on the Internet to carry out a variety of tasks, such as, for example building search (Brin and Page, 1998) and recommendation systems (Davidson et al., 2010) that could be used to improve business, health-care and political decisions (Ferrucci et al., 2013).

The purpose of our proposal is to present LETO: Learning Engine Through Ontologies, a framework to automatically and gradually extract knowledge from different sources (both structured and unstructured), building internal representations that can be adapted to and integrated in multiple domains. The current state of LETO's implementation is a computational prototype that illustrates the different components of its architecture and demonstrate its feasibility. Inspired by the different processes that occur during human learning, we design the framework's architecture as a learning pipeline that gradually builds more complex knowledge.

In a simplified view, the human learning process can be modeled as a continuous loop that transforms sensorial data into knowledge (see Figure 1) (Gross, 2015). Humans collect information about the environment through senses, where the human brain attempts to detect relations between individual signals to form a more structured representation of reality. By relating as many signals as possible, humans build a much richer semantic representation of the environment, which is unconsciously filtered storing only the most relevant part. In order to achieve this, the brain is able to access to stored experiences about what has been important before, and what is already known. This feedback loop also evaluates previously known facts, and modifies them at the light of new experiences. In time, humans not only learn new facts, but also learn better ways of learning.

The challenge of building computational knowledge discovery systems is an active research problem in the field of artificial intelligence, specifically in emerging areas such as ontology learning (Cimiano et al., 2009) and learning by reading (Barker et al., 2007). Modern systems employ a combination of knowledge-based techniques (i.e., using rules handcrafted by domain experts (Chandrasekaran, 1986)) and statistical approaches (i.e., based on pattern recognition with statistical and probabilistic models (Kevin, 2012)).

Given the large amount of information available online, several knowledge discovery systems fo-

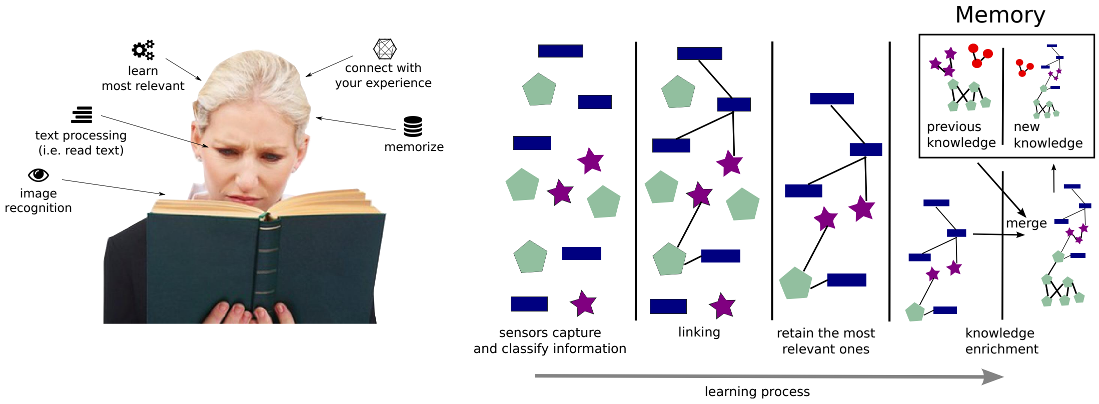


Figure 1: Simplified representation of the human learning process.

cus on extracting knowledge and exploiting the semi-structured format of web resources , e.g, ARTEQUAKT (Alani et al., 2003), SOBA (Buitelaar et al., 2006) and WEB->KB (Craven et al., 2000). In order to extract relevant knowledge from natural language text, NLP techniques have been introduced in systems such as OPTIMA (Kim et al., 2008) and ISODLE (Weber and Buitelaar, 2006). Natural language features can be used to build rule-based systems (e.g., OntoLT (Buitelaar and Sintek, 2004)) or systems based on statistical or probabilistic models trained on NLP corpora, such as LEILA (Suchanek et al., 2006) or Text2Onto (Cimiano and Völker, 2005). Some systems address the issue of inferring more abstract knowledge from the extracted facts, often using unsupervised techniques to discover inherent structures. Relevant examples of this approach are OntoGain (Drymonas et al., 2010), ASIUM (Faure and Poibeau, 2000) and BOEMIE (Castano et al., 2007).

Most of the mentioned systems focus on one iteration of the extraction process. However, more recent approaches, like NELL (Mitchell et al., 2018), attempt to learn continuously from a stream of web data, and increase over time both the amount and the quality of the knowledge discovered.

One of the main characteristics of LETO, in contrast to similar proposals in the literature (such as NELL (Mitchell et al., 2018) or BOEMIE (Petasis et al., 2011)), is the explicit management of separated pieces of knowledge. By isolating the knowledge for different domains, it is possible to apply different techniques and/or parameters as appropriate. Besides, this allows the temporal existence of contradictions or unreliable information that can be crosschecked in the future.

The rest of the paper is organized as follows to facilitate a detailed description of our proposal: Section 2 describes the proposed architecture of a general framework for knowledge discovery. In Section 3 we present an application of LETO to a specific knowledge discovery problem combining Twitter and IMDB. Finally, in Section 4 we present the main conclusions of the research and outline possible future works.

2 Learning Engine Through Ontologies (LETO)

In this section we present LETO, a general architecture for a framework designed to discover relevant knowledge from a variety of data sources, both structured and unstructured.

The LETO framework is divided into 6 modules, which are interrelated. Each module has a specific responsibility defining the inputs and outputs that establish the intercommunication among the rest of the modules within the framework. Figure 2 shows a general overview of the framework.

As shown in Figure 2 the top layer (Data Sources) represents the sources of data that serve as input for the framework. The middle layer contains the Main Modules, which perform the processing of the input data to extract and discover the relevant knowledge latent in this data. Figure 2 also shows the subprocesses that occur inside each module. The main modules always communicate with each other by sharing ontologies. The following sections 2.1, 2.2 and 2.3 explain in detail the inner workings of the main modules. The bottom layer (Backend) contains modules used by the rest of framework:

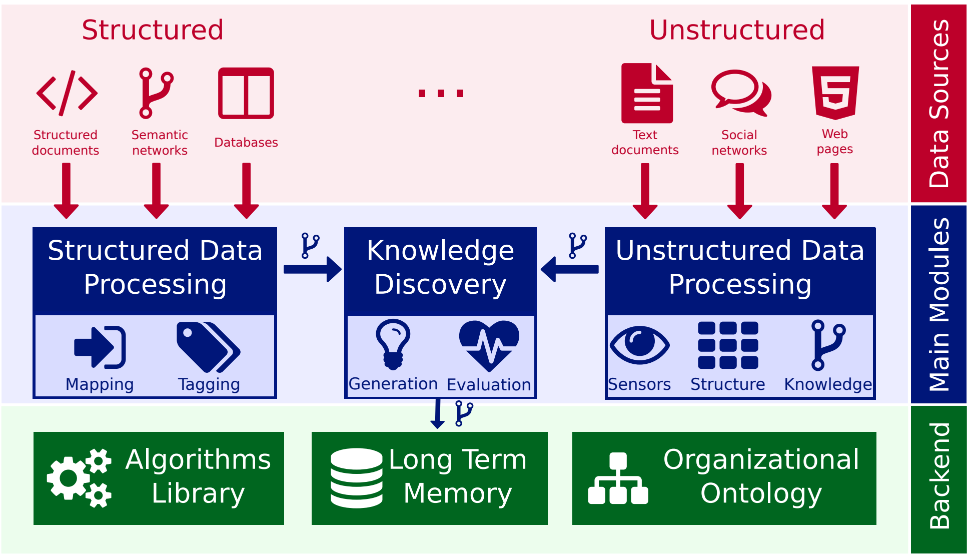


Figure 2: Overview of the architecture of LETO.

Algorithms Library: Contains different algorithms and mathematical models for solving specific problems, along with associated metadata.

Long Term Memory: Contains all the knowledge accumulated by the other modules, in the form of individual ontologies with metadata that describes their content.

Organizational Ontology: An internal representation of the framework's components in an ontological format which enables the automatic construction of the user interface.

2.1 Structured Data Processing

This module is responsible for processing structured data. Sources for structured data are available online in different formats. Among the different types of structures for representing information, such as relational databases, concept maps, knowledge graphs, and others, LETO proposes the use of ontologies for their semantic richness. Ontologies were chosen because they are more expressive than other DTO (Data Transfer Object) formats.

The general pipeline that this module performs can be thought of as a classic Extract, Transform and Load process (ETL) (Vassiliadis, 2009; Hermida et al., 2012). Afterwards, the normalized and tagged block of knowledge (stored as an ontology) is handled to the knowledge processing module, for further refinement and storage. Figure 3 shows an schema of this module. This module performs two main tasks:

Mapping: Since there are many different structured formats, the first stage of this module is to convert any of these representations into a standard representation for internal use, in the form of an ontology, using a mapping process (Choi et al., 2006; Y. An and Mylopoulos, 2006; Noy and Musen, 2003). The current implementation infers classes and relations from CSV or TSV input files using a rule-based approach, and outputs and ontology in OWL format.

Tagging: This step attaches several tags, such as source, domain, topic and reliability to the mapped ontology. This tags can be either inferred automatically (e.g., the domain and reliability) or provided by the user (e.g., the source). The current implementation requires a manual input by a domain expert.

2.2 Unstructured Data Processing

The sources for unstructured data are extremely varied in format and computational representation. Text is one of the most common forms for storing and communicating human knowledge, but pictures, sound files, and videos are also interesting and increasingly popular forms of communication. Also, in contrast with structured sources, there is a lot of variety in the level of reliability and completeness of unstructured sources.

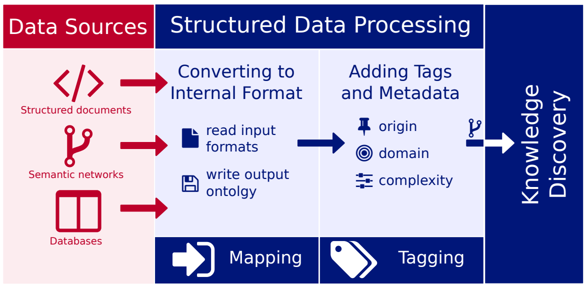


Figure 3: A schema of the Structured Data Module, and a representation of the processes that occur in each of the two main tasks performed by this module.

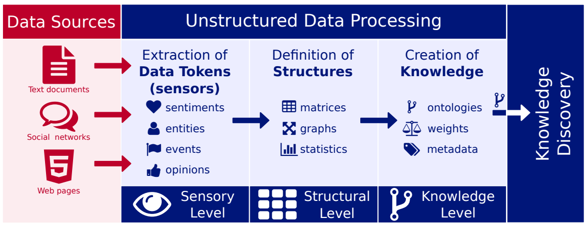


Figure 4: A schema of the three stages of the Unstructured Data Module and its relation with the rest of the framework.

Besides these factors, contrary to structured sources, there is no predefined structure of concepts and relations inside a block of unstructured data. Hence, the module for processing unstructured data is designed as a pipeline through which simple concepts are processed and transformed into more complex ones. Figure 4 shows a schema of this module, as well as an example of the type of processes that occur inside and their relation with each other. The module is organized in a three-levels pipeline as follows:

Sensory Level: Contains a number of processing units called "sensors", which extract different chunks of data. Among the implemented sensors, LETO includes named entity recognition) (Gattani et al., 2013), sentiment analysis (Montoyo et al., 2012), and detection of subject-actions-target triplets (Estevez-Velarde et al., 2018). In general, each of these sensors performs a specific analysis and produces a stream of *data tokens* of a particular type. Each of these data tokens represents a single unit of semantic information, for instance, the existence of a particular entity, or the association between an entity and an event, and are not interrelated.

Structural Level: The data tokens extracted from the original source are processed as a group to find an underlying structure. Techniques implemented in this stage include Latent Semantic Analysis (LSA) (Hofmann, 2017), Principal Components Analysis (PCA) (Guo et al., 2002), Word Embeddings (Turian et al., 2010) and clustering techniques. The output of this stage is either a graph, a correlation matrix, or some statistical description that represents the underlying structure of the data tokens that were previously extracted.

Knowledge Level: The structured information that was previously built is analyzed to refine, remove noise, and extract the relevant pieces of knowledge, based on clustering techniques. This allows synthesizing the knowledge discovered so far according to the context defined by the relations between the semantic units extracted in the previous stage. The output of this stage is always an ontology, which is then passed to the Knowledge Discovery Module for further integration with the stored knowledge. The resulting ontology then becomes part of the stored knowledge of the framework, which is iteratively refined, corrected and enhanced with new knowledge extracted from different sources.

2.3 Knowledge Discovery

The knowledge discovery module receives the output from unstructured data processing and structured data processing, always in the form of an ontology. Each of these ontologies represents a collection of knowledge assets from a particular domain or a general domain. Some of them may overlap, containing the same knowledge facts, even if labeled as different entities or relations. Others may have contradictions or inconsistencies, either within themselves or with one another, see Figure 5. For this purpose, this module performs two main tasks:

Generation: The generation of knowledge involves two processes, namely the merging of ontologies (Noy et al., 2000; Noy and Musen, 2003), and the generation of new (or more general-domain) ontologies from other ontologies (Aussenac-Gilles and Jacques, 2006; Blomqvist, 2009). Merging ontologies requires this module be able to undertake a matching among entities, relations and instances in two or more ontologies that are deemed similar (Shvaiko and Euzenat, 2013).

Evaluation: After the new ontology is created, this step provides quality evaluation metrics that assert the reliability, completeness or soundness of the new knowledge. These metrics are based on comparing the new ontology with the existing knowledge.

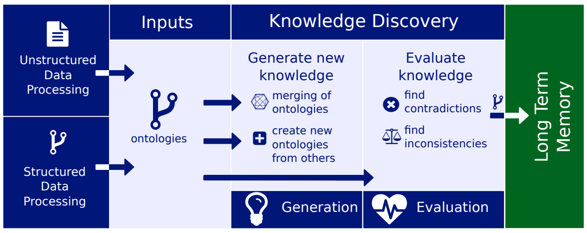


Figure 5: A schema of the architecture of Knowledge Discovery Module, showing the internal tasks performed and its relations with the rest of the framework.

3 Application of LETO to Knowledge Discovery

This section shows the use of the LETO system through a practical scenario that involves the processing of both unstructured and structured data sources. This application illustrates the types of processes (i.e. processing of both unstructured and structured data sources) that our framework performs. We select the Internet Movie Database (IMDB) that contains information about films and actors. We aim to enrich this knowledge with opinions expressed in social networks. Opinions can be extracted from a specific Twitter *hashtag* feed (i.e., `#Oscars`). Figure 6 shows a schematic representation of the whole process.

The first step consist of obtaining the IMDB data (in CSV format) and mapping it to an OWL ontology. Data from IMDB was obtained in tab-separated files, processed by LETO's generic mapping pipeline which infers class names and relation names from the CSV structure. This results in a total of 4,807,262 film instances and 8,427,043 person instances, related by 27,044,985 tuples in 12 different relation types. After the mapping process, the resulting ontology is tagged with relevant metadata. In this case, the domain is **Cinema**, and a high confidence can be assigned since this source is known to be of high quality. These steps are represented in the figure with the numbers *1a* and *1b* and performed in LETO using the *Structured Data Processing* module (see Fig. 3).

The next step involves the processing of a continuous stream of Twitter messages (*2a*). These are obtained through the standard Twitter query API, filtering with the hashtag `#Oscars`, which returned 3375 messages that span a period of 2 weeks. Using standard NLP techniques, each tweet is processed to obtain named entities (Nadeau and Sekine, 2007) and an opinion label (Pang et al., 2008; Liu, 2012) (*2b*). The entity sensor was implemented using spaCy (Honnibal and Montani, 2017), which returned 524 unique PERSON instances, from a total of 1961 PERSON mentions. The document level emotion sensor was implemented through the use of the SAM[1] project (Fernández et al., 2015). An example output of the entity sensor is shown in Figure 7. Similar interfaces are available in LETO for interacting with all the components of the framework, but are not shown for space restrictions.

Afterwards, the different mentions of the same entities across multiple tweets are matched together (*2c*). The least relevant mentions (e.g., those with very few appearances) are filtered out (*2d*). through the clustering technique Affinity Propagation (Pedregosa et al., 2011). Finally, the filtered entities with their associated opinions (*2e*) are tagged and stored in an ontology (*2f*). These steps are performed using components from the *Unstructured Data Processing* module.

After the processing of both structured and unstructured data is completed (*3a*), both sources are selected for a knowledge integration process (*3b*). An ontology mapping technique (Choi et al., 2006) is applied, which maps relevant instances of the IMDB ontologies to their corresponding mentions in the tweets (*3c*). The result of this mapping process is an ontology in the same format as IMDB, but with additional aggregated opinion labels for each instance (of those found in Twitter). This enriched knowledge is tagged (*e3*) and stored for future use. These steps are performed using components from the *Knowledge Discovery* module. The resulting ontology can be visualized in LETO, as shown in Figure 9. This visualization tool shows both the classes and instances, enabling an interactive exploration of the ontology.

[1] http://wiki.socialisingaroundmedia.com/

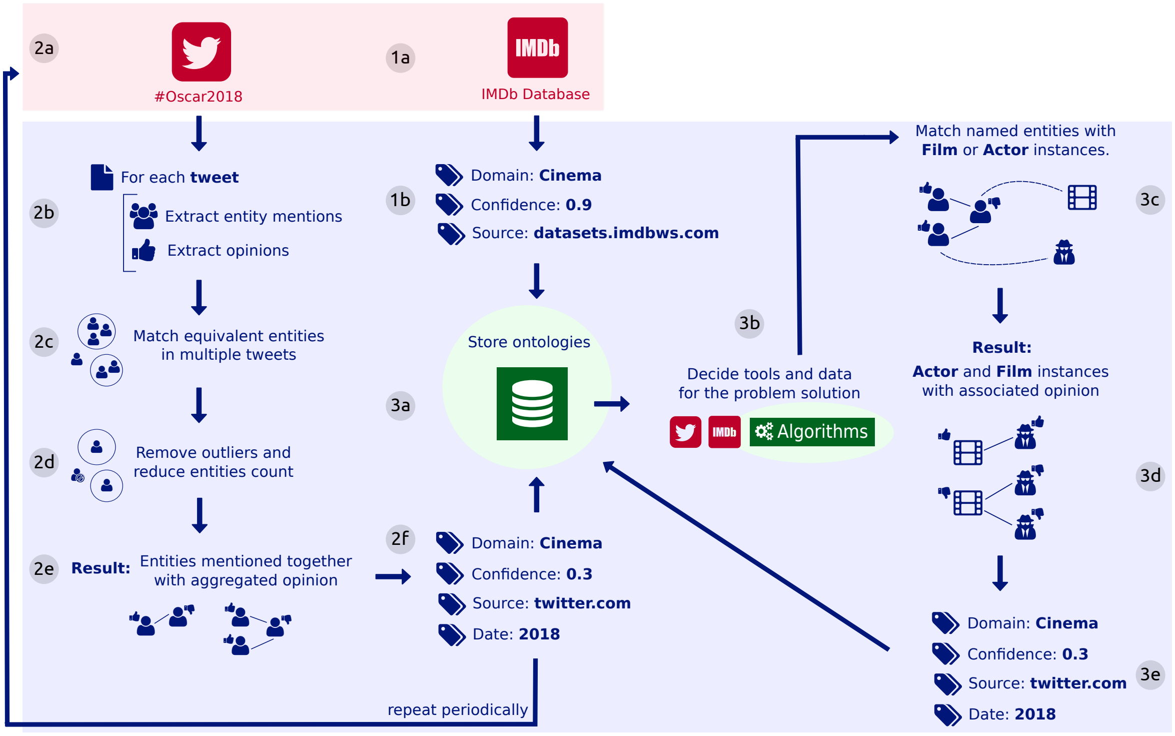
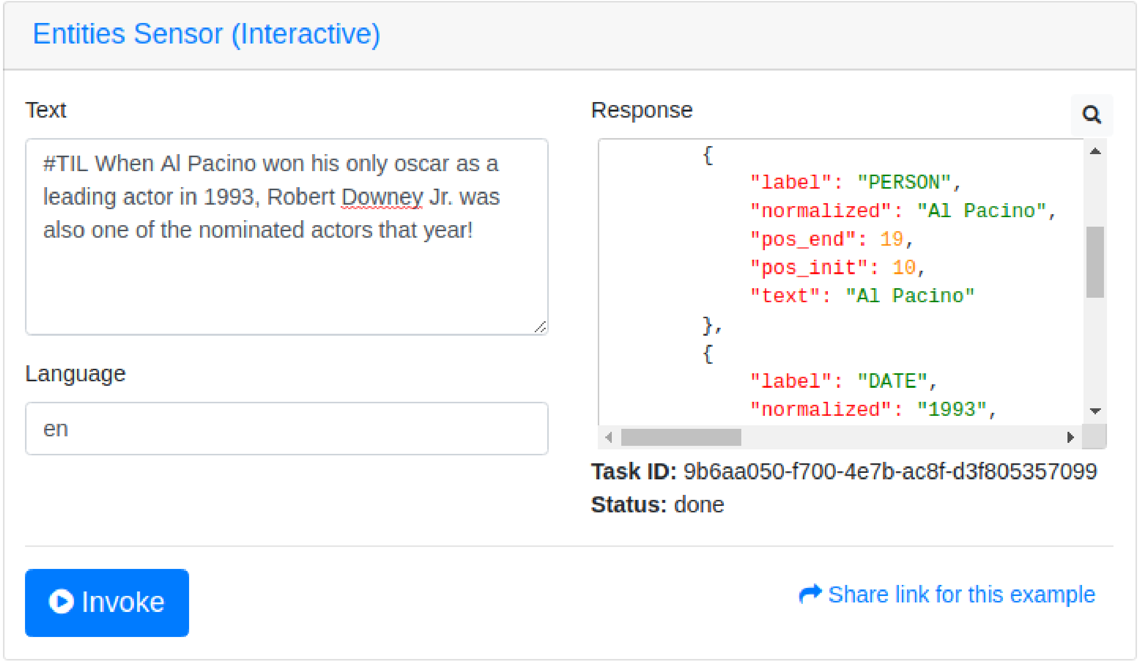

2a
#Oscar2018
1a
IMDb
IMDb Database
2b
For each **tweet**
Extract entity mentions
Extract opinions
1b
Domain: **Cinema**
Confidence: **0.9**
Source: **datasets.imdbws.com**
Match named entities with **Film** or **Actor** instances.
3c
2c
Match equivalent entities in multiple tweets
Store ontologies
3b
Decide tools and data for the problem solution
IMDb
Algorithms
Result: **Actor** and **Film** instances with associated opinion
3a
2d
Remove outliers and reduce entities count
3d
2f
2e
Result: Entities mentioned together with aggregated opinion
Domain: **Cinema**
Confidence: **0.3**
Source: **twitter.com**
Date: **2018**
Domain: **Cinema**
Confidence: **0.3**
Source: **twitter.com**
Date: **2018**
3e
repeat periodically

Figure 6: Schematic representation of the process for mapping emotions in reaction to movie reviews with IMDB.


Entities Sensor (Interactive)

Text

#TIL When Al Pacino won his only oscar as a leading actor in 1993, Robert Downey Jr. was also one of the nominated actors that year!

Language

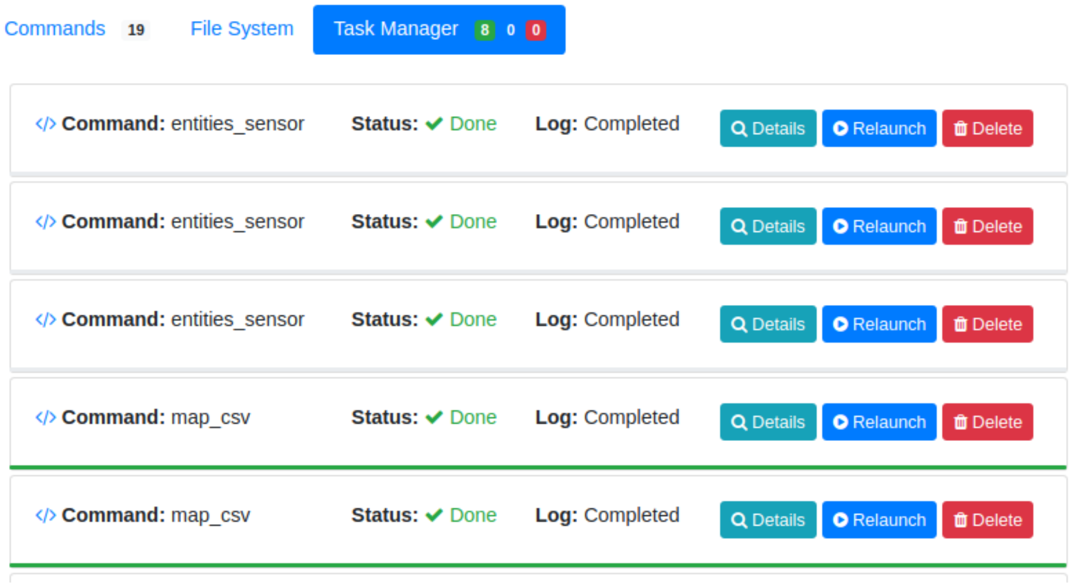
en

Response

```
        {
            "label": "PERSON",
            "normalized": "Al Pacino",
            "pos_end": 19,
            "pos_init": 10,
            "text": "Al Pacino"
        },
        {
            "label": "DATE",
            "normalized": "1993",
```

Task ID: 9b6aa050-f700-4e7b-ac8f-d3f805357099
Status: done

Figure 7: Example execution of the Entity Sensor for one tweet. A similar interface allows the batch execution for a collection of tweets.



Commands 19 File System Task Manager 8 0 0

</> **Command:** entities_sensor **Status:** ✓ Done **Log:** Completed

</> **Command:** entities_sensor **Status:** ✓ Done **Log:** Completed

</> **Command:** entities_sensor **Status:** ✓ Done **Log:** Completed

</> **Command:** map_csv **Status:** ✓ Done **Log:** Completed

</> **Command:** map_csv **Status:** ✓ Done **Log:** Completed

Figure 8: Main UI of LETO, specifically the Task Management view.

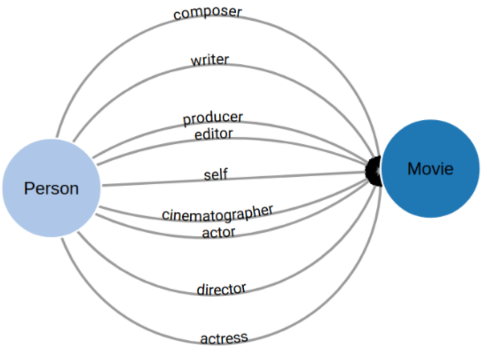


Figure 9: Visualizing the structure of the IMDB ontology in LETO.

The knowledge generation process involved matching Twitter PERSON instances with IMDB instances and attaching an average of the emotions found in each mention of the corresponding instance. A total of 212 instances were matched, which indicates a 40.45% of accuracy for the Twitter entity extractor. A manual review of the 542 recognized instances was performed, to evaluate the reasons for the mistakes. All entities appearing in Twitter where searched in Google and the first result was used as ground truth. Table 1 summarizes these results.

The current implementation of LETO provides an interactive application where researchers can apply the different algorithms and techniques implemented in each module, both interactively (i.e., using a single input example) or in batch mode.

Metric	Value	Percent
Correct matches	212	**40.45**
Correct mismatch	19	**3.62**
Matching error	118	22.52
Extraction error	165	31.48
Knowledge error	10	1.91
Context missing	2	0.38
Total errors	293	55.92

Table 1: Summary of results of the knowledge discovery process.

LETO supports multiple processes running in parallel, and provides tools for running and monitor long-term processes that can take hours or days. Figure 8 shows a overall view of LETO's main user interface, specifically the view for task management.

4 Conclusion and Future Works

In this research work, the aim was to design and implement a framework for automatic knowledge discovery from different data sources. We considered the discovery of knowledge from structured and unstructured sources of information. This framework has been designed as a modular set of components that perform specific tasks and communicate with each other. An open-source prototype implementation of LETO is currently available[2], which already contains several of the main components. In future lines of development, we will pursue the implementation of more var-

[2] https://github.com/knowledge-learning/leto

ied sensors, and more complex mechanisms for knowledge integration (e.g., ontology merging and mapping processes). Another line for future research is related to context mismatch and recognition, specifically in the *Unsupervised Processing Module*. This process is necessary for accurately matching portions of unstructured text to sections of an already stored ontology. We will also focus on extending the automation processes currently available in LETO.

Acknowledgments

This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, it has also been partially funded by both aforementioned universities, the Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVINGLANG (RTI2018-094653-B-C22) and INTEGER (RTI2018-094649-B-I00).

References

Harith Alani, Sanghee Kim, David E Millard, Mark J Weal, Wendy Hall, Paul H Lewis, and Nigel R Shadbolt. 2003. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems* 18(1):14–21.

Nathalie Aussenac-Gilles and Marie-Paule Jacques. 2006. Designing and evaluating patterns for ontology enrichment from texts. In *International Conference on Knowledge Engineering and Knowledge Management*. Springer, pages 158–165.

Ken Barker, Bhalchandra Agashe, Shaw Yi Chaw, James Fan, Noah Friedland, Michael Glass, Jerry Hobbs, Eduard Hovy, David Israel, Doo Soon Kim, et al. 2007. Learning by reading: A prototype system, performance baseline and lessons learned. In *AAAI*. volume 7, pages 280–286.

Eva Blomqvist. 2009. Ontocase-automatic ontology enrichment based on ontology design patterns. In *International Semantic Web Conference*. Springer, pages 65–80.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30(1-7):107–117.

Paul Buitelaar, Philipp Cimiano, Stefania Racioppa, and Melanie Siegel. 2006. Ontology-based information extraction with soba. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Paul Buitelaar and Michael Sintek. 2004. Ontolt version 1.0: Middleware for ontology extraction from text. In *Proc. of the Demo Session at the International Semantic Web Conference*.

Silvana Castano, Sofia Espinosa, Alfio Ferrara, Vangelis Karkaletsis, Atila Kaya, Sylvia Melzer, Ralf Möller, Stefano Montanelli, and Georgios Petasis. 2007. Ontology dynamics with multimedia information: The boemie evolution methodology. In *International Workshop on Ontology Dynamics (IWOD-07)*. page 41.

Balakrishnan Chandrasekaran. 1986. Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design. *IEEE expert* 1(3):23–30.

Namyoun Choi, Il-Yeol Song, and Hyoil Han. 2006. A survey on ontology mapping. *ACM Sigmod Record* 35(3):34–41.

Philipp Cimiano, Alexander Mädche, Steffen Staab, and Johanna Völker. 2009. Ontology learning. In *Handbook on ontologies*, Springer, pages 245–267.

Philipp Cimiano and Johanna Völker. 2005. text2onto. In *International Conference on Application of Natural Language to Information Systems*. Springer, pages 227–238.

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. 2000. Learning to construct knowledge bases from the world wide web. *Artificial intelligence* 118(1-2):69–113.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, pages 293–296.

Euthymios Drymonas, Kalliopi Zervanou, and Euripides GM Petrakis. 2010. Unsupervised ontology acquisition from plain texts: the OntoGain system. In *International Conference on Application of Natural Language to Information Systems*. Springer, pages 277–287.

S. Estevez-Velarde, Y. Gutierrez, A. Montoyo, A. Piad-Morffis, R. Munoz, and Y. Almeida-Cruz. 2018. Gathering object interactions as semantic knowledge (accepted). In *Proceedings of the 2017 International Conference on Artificial Intelligence (ICAI'17)*.

David Faure and Thierry Poibeau. 2000. First experiments of using semantic knowledge learned by asium for information extraction task using intex. In *Proceedings of the ECAI workshop on Ontology Learning*.

Javi Fernández, Yoan Gutiérrez, José M Gómez, and Patricio Martínez-Barco. 2015. Social rankings: análisis visual de sentimientos en redes sociales. *Procesamiento del Lenguaje Natural* 55:199–202.

David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T Mueller. 2013. Watson: beyond jeopardy! *Artificial Intelligence* 199:93–105.

Abhishek Gattani, Digvijay S Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. 2013. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. *Proceedings of the VLDB Endowment* 6(11):1126–1137.

Richard Gross. 2015. *Psychology: The science of mind and behaviour 7th edition*. Hodder Education.

Q Guo, W Wu, DL Massart, C Boucon, and S De Jong. 2002. Feature selection in principal component analysis of analytical data. *Chemometrics and Intelligent Laboratory Systems* 61(1-2):123–132.

Jesús M Hermida, Santiago Meliá, Jose-Javier Martínez, Andrés Montoyo, and Jaime Gómez. 2012. Developing semantic rich internet applications with the s m 4ria extension for oide. In *International Conference on Web Engineering*. Springer, pages 20–25.

Thomas Hofmann. 2017. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*. ACM, volume 51, pages 211–218.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* .

Murphy Kevin. 2012. Machine learning: a probabilistic perspective.

Sang-Soo Kim, Jeong-Woo Son, Seong-Bae Park, Se-Young Park, Changki Lee, Ji-Hyun Wang, Myung-Gil Jang, and Hyung-Geun Park. 2008. Optima: An ontology population system. In *3rd Workshop on Ontology Learning and Population (July 2008)*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5(1):1–167.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. *Commun. ACM* 61(5):103–115. https://doi.org/10.1145/3191513.

Andrés Montoyo, Patricio MartíNez-Barco, and Alexandra Balahur. 2012. Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Natalya F Noy and Mark A Musen. 2003. The prompt suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies* 59(6):983–1024.

Natalya Fridman Noy, Mark A Musen, et al. 2000. Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. 2011. Ontology population and enrichment: State of the art. In *Knowledge-driven multimedia information extraction and ontology evolution*. Springer-Verlag, pages 134–166.

Pavel Shvaiko and Jérôme Euzenat. 2013. Ontology matching: state of the art and future challenges. *IEEE Transactions on knowledge and data engineering* 25(1):158–176.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Leila: Learning to extract information by linguistic analysis. In *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. pages 18–25.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, pages 384–394.

Panos Vassiliadis. 2009. A survey of extract–transform–load technology. *International Journal of Data Warehousing and Mining (IJDWM)* 5(3):1–27.

Nicolas Weber and Paul Buitelaar. 2006. Web-based ontology learning with isolde. In *Proc. of the Workshop on Web Content Mining with Human Language at the International Semantic Web Conference, Athens GA, USA*. volume 11.

A. Borgida Y. An and J. Mylopoulos. 2006. Building semantic mappings from databases to ontologies. volume 21st National Conference on Artificial Intelligence (AAAI 06).

Sentence Simplification for Semantic Role Labelling and Information Extraction

Richard Evans
Research Institute in Information
and Language Processing
University of Wolverhampton
United Kingdom
r.j.evans@wlv.ac.uk

Constantin Orăsan
Research Institute in Information
and Language Processing
University of Wolverhampton
United Kingdom
c.orasan@wlv.ac.uk

Abstract

In this paper, we report on the extrinsic evaluation of an automatic sentence simplification method with respect to two NLP tasks: semantic role labelling (SRL) and information extraction (IE). The paper begins with our observation of challenges in the intrinsic evaluation of sentence simplification systems, which motivates the use of extrinsic evaluation of these systems with respect to other NLP tasks. We describe the two NLP systems and the test data used in the extrinsic evaluation, and present arguments and evidence motivating the integration of a sentence simplification step as a means of improving the accuracy of these systems. Our evaluation reveals that their performance is improved by the simplification step: the SRL system is better able to assign semantic roles to the majority of the arguments of verbs and the IE system is better able to identify fillers for all IE template slots.

1 Introduction

Sentence simplification is one aspect of text simplification, which is concerned with the conversion of texts into a more accessible form. In many cases, text simplification is performed to facilitate subsequent human or machine text processing. This may include processing for human reading comprehension (Canning, 2002; Scarton et al., 2017; Orăsan et al., 2018) or for NLP tasks such as dependency parsing (Jelínek, 2014), information extraction (Jonnalagadda et al., 2009; Evans, 2011; Peng et al., 2012), semantic role labelling (Vickrey and Koller, 2008), and multidocument summarisation (Blake et al., 2007; Siddharthan et al., 2004).

In previous research, Caplan and Waters (1999) noted a correlation between sentence comprehension difficulty for human readers and the numbers of propositions expressed in the sentences being read.[1] Evans and Orăsan (2019) presented an iterative rule-based approach to sentence simplification which is intended to reduce the per sentence propositional density of input texts by converting sentences which contain compound clauses and complex NPs[2] into sequences of simpler sentences.

Evaluation of text simplification systems is difficult, especially when such evaluations need to be conducted repeatedly for development purposes and cost is a critical factor. In general, the choice of evaluation method depends on the purpose of the simplification task. Various types of evaluation are currently used, but these are problematic. In previous work, evaluation of sentence simplification systems (including Evans and Orăsan's (2019) system, which is extrinsically evaluated in our current paper) has relied on one or more of three main approaches: the use of overlap metrics such as Levenshtein distance (Levenshtein, 1966), BLEU score (Papineni et al., 2002) and SARI (Xu et al., 2016) to compare system output with human simplified texts (e.g. Wubben et al., 2012; Glavas and Stajner, 2013; Vu et al., 2014); automated assessments of the readability of system output (Wubben et al., 2012; Glavas and Stajner, 2013; Vu et al., 2014); and surveys of human opinions about the grammaticality, readability, and meanings of system output (Angrosh et al., 2014; Wubben et al., 2012; Feblowitz and Kauchak, 2013). In previous work, researchers have also used methods such as

[1] Propositions are atomic statements that express simple factual claims (Jay, 2003). They are considered the basic units involved in the understanding and retention of text (Kintsch and Welsch, 1991).

[2] NPs which contain finite nominally bound relative clauses.

eye tracking (Klerke et al., 2015; Timm, 2018), and reading comprehension testing (Orăsan et al., 2018) to evaluate text simplification systems.

There are several challenges in these approaches to evaluation. The development of gold standards in text simplification is problematic because they are difficult to produce and numerous variant simplifications are acceptable. As a result, existing metrics may not accurately reflect the usefulness of the simplification system being evaluated. Even when there are detailed guidelines for the simplification task, there is still likely to be a variety of means by which a human might simplify a text to produce a reference simplification. Further, due to the difficulty of the human simplification task, it may be that evaluation measures such as BLEU and SARI are unable to exploit a sufficiently large set of reference simplifications.

Evaluation of text simplification methods using automatic readability metrics is problematic because the extent to which all but a handful of readability metrics correlate with human reading comprehension is uncertain. Evaluation via opinion surveys of readers is difficult because participants may have varying expectations about the upper and lower limits of sentence complexity, making responses to Likert items unreliable. Participants also vary in terms of linguistic ability and personal background knowledge. These variables, which affect reading behaviour and may affect responses to opinion surveys, are difficult to control.

When using methods such as eye tracking to evaluate text simplification, previous work has shown that differences in reading behaviour depend on participants' reading goals (Yeari et al., 2015). This variable is usually controlled by asking participants to respond to text-related opinion surveys or multiple choice reading comprehension questions. One adverse effect of this is that these evaluations may be of limited validity when considering the usefulness of system output for other purposes. While we may learn whether a sentence simplification method improves participants' performance in answering short reading comprehension questions, it is not clear whether similar benefits would be obtained in terms of readers' abilities to be entertained by the text or to understand it well enough to be able to summarise it for friends.

Given that text simplification is usually made for a particular purpose, the evaluation method should offer insights into the suitability of the text simplification system for this purpose. Extrinsic evaluation offers the possibility of meeting this requirement. Text simplification has also been claimed to improve automatic text processing (e.g. Vickrey and Koller, 2008; Evans, 2011; Hasler et al., 2017), though the evidence for this has been fairly limited. In this paper, we explore whether syntactic simplification can facilitate two NLP tasks: semantic role labelling (SRL) and information extraction (IE).

In Section 2 of this paper, we present an overview of previous related work. In Section 3, we present an overview of Evans and Orăsan's (2019) method for sentence simplification, which is the simplification method used in our current paper. In Section 4, we present each of the extrinsic evaluation experiments based on SRL (Section 4.1) and IE (Section 4.2). Each of these sections describes the task, the test data used, the NLP system whose output is used for extrinsic evaluation of the sentence simplification system, our motivation for considering that accuracy of the NLP system may be improved via a preprocessing step in which sentence simplification is performed, the evaluation method, our results, and a discussion of the results. In Section 5, we draw conclusions and consider directions for future work.

2 Related Work

Chandrasekar and Srinivas (1997) hypothesised that approaches to sentence simplification may evoke improvements in subsequent text processing tasks. In previous work, researchers have sought to determine whether or not a preprocessing step based on text simplification can facilitate subsequent natural language processing. In the current paper, our concern is to investigate the impact of a system simplifying sentences which contain compound clauses. Hogan (2007) and Collins (1999) observed that, for dependency parsers, dependencies involving coordination are identified with by far the worst accuracy of any dependency type (F_1-*score* $\approx 61\%$). This is one factor motivating our research in this direction.

Sentence simplification has also been applied as a preprocessing step in neural machine translation and hierarchical machine translation (Hasler et al., 2017). In their approach, the approach to sentence simplification was sentence compression. One contribution of our current paper is an investigation of the use of an information preserving approach to sentence simplification as a preprocessing step in the NLP applications.

Vickrey and Koller (2008) applied their sentence simplification method to improve performance on the CoNLL-2005 shared task on SRL.[3] For sentence simplification, their method exploits full syntactic parsing with a set of 154 parse tree transformations and a machine learning component to determine which transformation operations to apply to an input sentence. They find that a SRL system based on a syntactic analysis of automatically simplified versions of input sentences outperforms a strong baseline. In their evaluation, Vickrey and Koller (2008) focus on the overall performance of their SRL system rather than on the particular contribution made by the sentence simplification method. As noted earlier, in our current paper, we isolate sentence simplification as a preprocessing step and investigate its impact on subsequent NLP tasks.

3 Sentence Simplification System

Evans and Orăsan (2019) presented an iterative rule-based method for sentence simplification based on a shallow syntactic analysis step. Their system transforms input sentences containing compound clauses and complex NPs into sequences of simpler sentences that do not contain these types of syntactic complexity.

The first stage of sentence simplification is a shallow syntactic analysis step which tags textual markers of syntactic complexity, referred to as *signs*, with information about the syntactic constituents that they coordinate or of which they are boundaries. The signs of syntactic complexity are a set of conjunctions, complementisers, wh-words, punctuation marks, and bigrams consisting of a punctuation mark followed by a lexical sign. In the analysis step, syntactic constituents are not identified. It is only the signs which are tagged. The automatic sign tagger was developed by Dornescu et al. (2013). In their scheme, clause coordinators are tagged CEV[4] while the left boundaries of subordinate clauses are tagged SSEV.[5]

After shallow syntactic analysis of the sentence, an iterative algorithm is applied to sentences containing compound clauses and complex NPs. The algorithm (Algorithm 1) integrates a sentence transformation function which implements the transformation schemes listed in Table 1.

```
   Input: Sentence $s_0$, containing at least one
          sign of syntactic complexity of class $c$,
          where $c \in \{$CEV, SSEV$\}$.
   Output: The set of sentences $A$ derived from
           $s_0$, that have reduced propositional
           density.
1  The empty stack $W$;
2  $O \leftarrow \emptyset$;
3  $push(s_0, W)$;
4  while $isEmpty(W)$ is $false$ do
5      $pop(s_i, W)$;
6      if $s_i$ contains a sign of syntactic
         complexity of class c (specified in Input)
         then
7          $s_{i_1}, s_{i_2} \leftarrow transform_c(s_i)$;
8          $push(s_{i_1}, W)$;
9          $push(s_{i_2}, W)$;
10     else
11         $O \leftarrow O \cup \{s_i\}$
12     end
13 end
```

Algorithm 1: Sentence simplification algorithm

In its original implementation, the *transform* function (line 7 of Algorithm 1) included 28 sentence simplification rules to implement one transformation scheme simplifying compound clauses and 125 rules implementing three transformation schemes simplifying sentences which contain complex NPs. Evaluation of the method revealed that simplification of sentences containing complex NPs was significantly less reliable than simplification of sentences containing compound clauses. For this reason, in the extrinsic evaluations presented in this paper, we deactivated the rules simplifying sentences that contain complex NPs. Each of the remaining implemented rules includes a rule activation pattern which, when detected in the input sentence, triggers an associated transformation operation. Table 1 presents the transformation scheme used to simplify compound clauses and an example of the sentence transformation that it makes. Input sentences are transformed if they match any of the rule activation patterns, which are expressed in terms of particular words, parts of speech, and tagged signs of syntactic complexity. Each application of a rule transforms a single input sentence into two sim-

[3] http://www.lsi.upc.edu/~srlconll/spec.html. Last accessed 14th May 2019.

[4] Coordinator of Extended projections of a Verb.

[5] Start of Subordinate Extended projection of a Verb.

Scheme	Input Sentence	Output Sentence 1	Output Sentence 2
A [B -**CEV** C] D. → A B D. A C D.	{They were formally found not guilty by the recorder Michael Gibbon QC after}$_A$ [a witness, who cannot be identified, withdrew from giving evidence$_B$ and**CEV** prosecutor Susan Ferrier offered no further evidence$_C$]{}$_D$.	{They were formally found not guilty by the recorder Michael Gibbon QC after}$_A$ a witness, who cannot be identified, withdrew from giving evidence$_B$ {}$_D$.	{They were formally found not guilty by the recorder Michael Gibbon QC after}$_A$ prosecutor Susan Ferrier offered no further evidence$_C$ {}$_D$

Table 1: *Sentence transformation scheme used to simplify sentences containing compound clauses*

pler sentences which are added to the working set (stack W in Algorithm 1).

The iterative nature of the algorithm enables it to convert complex sentences containing multiple signs of syntactic complexity such as (1) into the sequence of simple sentences (2).

(1) Kattab, of Eccles, Greater Manchester, was required to use diluted chloroform water in the remedy, but the pharmacy only kept concentrated chloroform, which is 20 times stronger.

(2) a. Kattab, of Eccles, Greater Manchester, was required to use diluted chloroform water in the remedy.
b. The pharmacy only kept concentrated chloroform.
c. Concentrated chloroform is 20 times stronger.

4 Experimental Setup

We evaluated the sentence simplification method extrinsically via two NLP applications. In each case, the application was treated as a black box. We compared performance of the system when processing input in its original form and in an automatically simplified form generated by the simplification method. As noted in Section 3, our approach to sentence simplification is syntactic rather than lexical. As they are based to some extent on exact string matching, the experiments described in this paper would be unsuitable for evaluation of lexical simplification systems.

4.1 Semantic Role Labelling

Semantic role labelling (SRL) is the task of automatically detecting the different arguments of predicates expressed in input sentences. We evaluated a system performing SRL in accordance with the Propbank formalism. In this scheme, an "individual verb's semantic arguments are numbered, beginning with zero. For a particular verb, [A0] is generally the argument exhibiting features of a Prototypical Agent (Dowty, 1991), while [A1] is a Prototypical Patient or Theme. No consistent generalizations can be made across verbs for the higher-numbered arguments"[6] (Palmer et al., 2005). The scheme includes semantic roles for "general, adjunct-like arguments" providing information on the verb's cause [AMCAU], direction [AMDIR], discourse relations [AMDIS], location [AMLOC], manner [AMMNR], modal function[7] [AMMOD], negation [AMNEG], purpose [AMPNC], and time [AMTMP], among others. For extrinsic evaluation of the sentence simplification method, we focused on verbal predicates[8], their arguments, and the nine listed adjunct-like argument types.

Table 2 provides an example of SRL to analyse sentence (3).

(3) When Disney offered to pay Mr. Steinberg a premium for his shares, the New York investor didn't demand the company also pay a premium to other shareholders.

The table contains a row of information about the semantic roles associated with each of the four main verbs occurring in the sentence. For example, it encodes information about the agent (*the New York investor*), patient or theme (*the company also pay a premium to other shareholders*), time (*When Disney offered to pay Mr. Steinberg a premium for his shares*), and negation (*n't*) of the verb *demand*.

Test Data. No suitable test data exist to evaluate a SRL system as a means of extrinsically evaluating the sentence simplification method. Although annotated data from the CONLL-2004/5[9] shared tasks on SRL are available, this test data is available only for the original versions of input sentences and not for simplified versions which may be generated using sentence simplification systems. Given that it is difficult to map verbs,

[6] Such as [A2], etc.

[7] Applicable to verbs.

[8] As opposed to prepositional, adjectival, or other types of predicate.

[9] http://www.lsi.upc.edu/~srlconll/home.html. Last accessed 23rd May 2019.

A0	V	A1	A2	A3	AMDIS	AMNEG	AMTMP
Disney	offered	to pay Mr. Steinberg a premium for his shares					
Disney	pay	his shares	Mr. Steinberg	a premium			
the New York investor	demand	the company also pay a premium to other shareholders				n't	When Disney offered to pay Mr. Steinberg a premium for his shares
the company	pay		other shareholders	a premium	also		

Table 2: *Example of semantic role labelling of Sentence (3)*

their arguments, and the semantic labels of these arguments from sentences in their original form to groups of sentences in their automatically generated simplifications, we developed a new set of test data for this purpose. We used a 7270-token collection of news articles from the METER corpus (Gaizauskas et al., 2001) to derive a new manually annotated data set. The original version of this dataset contains 265 sentences while the automatically simplified one contains 470 sentences.

NLP System. We made our extrinsic evaluation of the sentence simplification method using *Senna* (Collobert et al., 2011), a SRL system which tags predicates and their arguments in accordance with the formalism used in *Propbank*

Motivation. In our previous work (Evans and Orăsan, 2019), we used six metrics to assess the readability of the original and simplified versions of texts which include those that we use as test data for the SRL task. We found that the automatically simplified news texts have a lower propositional density (0.483 vs. 0.505) and reading grade level (5.4 vs. 10.3) and greater syntactic simplicity (89.07 vs. 46.81) and temporal consistency, assessed in terms of tense and aspect (30.15 vs. 27.76) than the original news texts. We determined the scores for these readability metrics using the CPIDR tool (Covington, 2012)[10] and the Coh-Matrix Web Tool (McNamara et al., 2014). As a task dependent on accurate syntactic parsing, we would expect that automatic SRL would be more accurate when processing the simplified versions of the input texts.

Evaluation Method. We applied Senna to the original and automatically simplified versions of the test data. Table 3 contains an example of the semantic roles labelled in one of the test sentences that we used. In this table, arguments identified more accurately in simplified sentences are underlined. For cases in which the SRL performed by Senna differed when processing the original and automatically simplified versions of input sentences, we manually inspected the two analyses, and recorded the number of cases for which SRL of the original sentence was superior to that of the simplified sentence, and vice versa. The inspection was made by a single annotator. In future work, we will seek to employ additional annotators for this task.

Results. Our manual evaluation of output from Senna revealed that 86.39% (1707) of the arguments identified in the two versions of the texts were identical. Of the remaining arguments, 5.31% (105) of those correctly identified in the original versions of the texts were not identified in the simplified versions, while 8.29% (164) of the arguments correctly identified in the simplified versions of the texts were not identified in the original versions. Of the 269 arguments identified in only one of the versions of the texts, 60.97% were arguments identified more accurately in the simplified version, while 39.03% were arguments identified more accurately in the original versions of the texts.

Table 4 shows the number of semantic roles labelled more accurately, by type, when Senna processes the original (*Orig*) and the automatically simplified (*Simp*) versions of news articles. To illustrate, when processing the original versions of the news texts, Senna correctly identifies the agents (arguments with semantic role label A0) of 14 verbs that it did not identify when process-

[10] `http://ai1.ai.uga.edu/caspr/CPIDR-3.2.zip`. Last accessed 31st May 2019.

Original sentence: But Smith had already been arrested - her clothing had been found near his home and DNA tests linked him to it.						
A0	**V**	**A1**	**A2**	**AMDIS**	**AMLOC**	**AMTMP**
	arrested	Smith		But		already
	found	her clothing			near his home and DNA tests linked him to it	
his home and DNA tests	linked	him	to it			
Simplified sentence: But Smith has already been arrested - her clothing had been found near his home. DNA tests linked him to it.						
A0	**V**	**A1**	**A2**	**AMDIS**	**AMLOC**	**AMTMP**
	arrested	Smith		But		already
	found	her clothing			near his home	
DNA tests	linked	him	to it			

Table 3: *Example of more accurate semantic role labelling in automatically simplified text.*

Role	**Orig vs. Simp**	**Simp vs. Orig**
A0 (agent)	14	23
A1 (patient/theme)	45	77
A2 (less prominent than A1)	14	13
AMCAU (cause)	0	1
AMDIR (direction)	4	0
AMDIS (discourse relation)	0	3
AMLOC (location)	3	13
AMMNR (manner)	4	6
AMNEG (negation)	0	1
AMPNC (purpose)	1	6
AMTMP (time)	12	27
V (verb)	2	3
Total	**99**	**173**

Table 4: *Positive differences in numbers of true positives obtained for semantic role labelling of original and simplified versions of input texts*

ing the automatically simplified versions of those texts. Conversely, when processing the automatically simplified versions, Senna correctly identified the agents of 23 verbs that it did not identify when processing the original versions.

Discussion. Overall, while there are advantages to performing SRL on each version of input texts, the greatest improvement in performance arises from processing the automatically simplified versions. A larger-scale evaluation is necessary but this observation constitutes some evidence that the sentence simplification method facilitates the NLP task of SRL.

4.2 Information Extraction

Information extraction (IE) is the automatic identification of selected types of entities, relations, or events in free text (Grishman, 2005). In this paper, we are concerned with IE from vignettes which provide brief clinical descriptions of hypothetical patients.

The discourse structure of these vignettes consists of six elements: *basic information* (patient's gender, profession, ethnicity, and health status); *chief complaint* (the main concern motivating the patient to seek medical intervention); *history* (a narrative description of the patient's social, family, and medical history); *vital signs* (a description of the patient's pulse and respiration rates, blood pressure, and temperature); *physical examination* (a narrative description of clinical findings observed in the patient); and *diagnostic study and laboratory study* (the results of several different types of clinical test carried out on the patient).

Each element in the discourse structure is represented by a template encoding related information. For example, the template for physical examinations holds information on each clinical finding/symptom (FINDING) observed in the examination, information on the technique used to elicit that finding (TECHNIQUE), the bodily location to which the technique was applied (LOCATION), the body system that the finding pertains to (SYSTEM), and any qualifying information about the finding (QUALIFIER). In this article, we focus on automatic extraction of information pertaining to physical examinations. The goal of the IE system is to identify the phrases used in the clinical vignette that denote findings and related concepts and add them to its database entry for the vignette.

Test Data. Our test data comprises a set of 286 clinical vignettes and completed IE templates, encoding information about TECHNIQUES, LOCATIONs, SYSTEMs, and QUALIFIERs, associated with the 719 FINDINGs that they contain. This test data was developed in the context of an earlier project and is based on clinical vignettes owned by the National Board of Medical Examiners.[11]

NLP System. For the experiments described in this paper, we used a simple IE system in which input texts are tokenised and part of speech tagged, domain-specific gazetteers are used to identify references to medical concepts and a simple set of finite state transducers (FSTs) is used to group adjacent references to concepts into multiword terms. The gazetteers and FSTs were developed in previous work presented by Evans (2011).

After tagging references to clinical concepts in the vignettes, IE is performed using a small number of simple rules. To summarize, vignettes are processed by considering each sentence in turn. Every mention of a clinical FINDING or SYMPTOM is taken as the basis for a new IE template. The first tagged TECHNIQUE, SYSTEM, and LOCATION within the sentence containing the focal SYMPTOM or FINDING is considered to be related to it.[12] QUALIFIERS (e.g. *bilateral* or *peripheral*) are extracted in the same way, except in sentences containing the word *no*. In these cases, the QUALIFIER related to the FINDING is identified as *none*.

The sentences in the test data were simplified using the method presented in Section 3. We then ran the IE system in two settings. In the first (IE_{ORIG}), it processed the original collection of vignettes. In the second (IE_{SIMP}), it processed the automatically simplified vignettes which contain a reduced number of compound clauses.

Motivation. An analysis of the readability of the original and simplified versions of the clinical vignettes did not provide a strong indication that the automatic sentence simplification method would improve the accuracy of the IE system. The 286 original clinical vignettes in the test data have a mean propositional density of 0.4826 ideas per word and 5.499 ideas per sentence. The values of these metrics for the simplified versions of the vignettes are 0.4803 ideas per word and 5.269 ideas per sentence, respectively. Although they are of the correct polarity, these differences are not statistically significant ($p = 0.5327$ and $p = 0.1407$, respectively). However, previous work in sentence simplification for IE (Jonnalagadda et al., 2009; Evans, 2011; Peng et al., 2012; Niklaus et al., 2016) has demonstrated that automatic sentence simplification can improve the accuracy of IE systems. This motivated us to evaluate the impact of the automatic sentence simplification method in this task.

Evaluation Method. For the IE task, our evaluation metric is based on F_1-score averaged over all slots in the IE templates and all templates in the test data. Identification of true positives is based on exact matching of system-identified slot fillers with those in the manually completed IE templates in our test data.

Results. The accuracy scores obtained by each variant of the IE system are presented in Table 5. Inspection of this table reveals that FINDINGs and all related concepts are identified more accurately in the simplified versions of the input texts.

Sentence (4) and its automatically simplified variant (5) provide an example of the difference in performance obtained by the two systems. In these examples, identified FINDINGs are italicised and associated concepts are underlined. Multiword terms appear in square brackets.

(4) She has $\underline{\text{truncal}}_{LOC}$ *obesity* and $\underline{\text{pigmented}}_{QUAL}$ $\underline{\text{abdominal}}_{LOC}$ *striae*.

(5) a. She has $\underline{\text{truncal}}_{LOC}$ [*obesity striae*].
b. She has $\underline{\text{pigmented}}_{QUAL}$ $\underline{\text{abdominal}}_{LOC}$ *striae*.

In (5-a), the FINDING *obesity* is not tagged correctly because the SYMPTOM *striae* is erroneously grouped with *obesity* to form a new FINDING, *obesity striae* which does not match the FINDING listed in the gold standard. By contrast, LOCATIONS in (5) are identified with greater accu-

[11] https://www.nbme.org/. Last accessed 31st May 2019.

[12] Versions of the system in which the closest tagged concept was extracted in each case, rather than the first, were significantly less accurate in both cases (overall accuracy of 0.6542 for IE from the original vignettes, and 0.6567 for IE from vignettes automatically simplified using the system described in Section 3). See Table 5 for results obtained using the superior IE system.

	IE$_{ORIG}$		IE$_{SIMP}$		
Template slot	Acc	95% CI	Acc	95% CI	Best Performer
FINDING	0.8819	[0.847, 0.914]	0.8861	[0.853, 0.917]	0.5486
TECHNIQUE	0.8514	[0.814, 0.886]	0.8903	[0.858, 0.922]	0.9344
SYSTEM	0.8097	[0.769, 0.850]	0.8431	[0.806, 0.881]	0.873
QUALIFIER	0.7431	[0.697, 0.786]	0.7708	[0.728, 0.814]	0.794
LOCATION	0.8431	[0.806, 0.881]	0.8611	[0.825, 0.894]	0.735
All	0.8258	[0.808, 0.843]	0.8503	[0.834, 0.867]	0.976

Table 5: *Performance of the IE systems processing our test data.*

racy than those in (4) because IE$_{ORIG}$ erroneously extracts the same LOCATION (*truncal*) for both FINDINGs in (4).

We applied a bootstrapping method to obtain confidence intervals for accuracy of extraction of each of the IE template slots. For this purpose, 50% of the the output of each system was randomly sampled in each of 100 000 evaluations. The confidence intervals are presented in the *95% CI* columns of Table 5. The figures in the *Best Performer* column of this table indicate the proportion of evaluations for which the IE_{SIMP} system was more accurate than the IE_{ORIG} system. Differences in the accuracy of IE were found to be statistically significant in all cases, using McNemar's test ($p < 0.00078$), with the exception of differences when extracting FINDINGs ($p = 0.6766$).

Discussion. Chinchor (1992) notes that assessment of the statistical significance of differences in accuracy between different IE systems is challenging. In our evaluation experiment, Dos Santos et al. (2018) framed the comparison between IE$_{ORIG}$ and IE$_{SIMP}$ using a binomial regression model. Given that such models apply only when the variables being considered are independent, dos Santos et al. (2018) included a latent variable in the analysis to represent the effect of the text on the performance of the two systems (the two evaluations are not independent because both systems process the same text). They showed that the odds ratio of agreement between IE$_{SIMP}$ and the gold standard is 1.5 times greater than that between IE$_{ORIG}$ and the gold standard. For all slots in the IE template, the probability of agreement between IE$_{ORIG}$ and the gold standard is 0.937. The probability of agreement between IE$_{SIMP}$ and the gold standard is 0.957. This difference is statistically significant. They conclude that IE$_{ORIG}$ and IE$_{SIMP}$ differ in their performance on the information extraction task. The probability of agreement with our gold standard is greater for IE$_{SIMP}$ than for IE$_{ORIG}$, although the probability of agreement is already large for IE$_{ORIG}$. This evaluation indicates that the automatic sentence simplification method facilitates IE.

5 Conclusions

As a result of various difficulties identified in current approaches to intrinsic evaluation of sentence simplification methods, we performed an extrinsic evaluation of one information-preserving sentence simplification method via three NLP tasks. We found that the sentence simplification step brings improvements to the performance of IE and SRL systems. In a third experiment, not described here due to space restrictions, we evaluated the sentence simplification method extrinsically with respect to a multidocument summarisation task using MEAD (Radev et al., 2006) to summarise clusters of documents developed for Task 2 of DUC-2004.[13] We found that the simplification step had no impact on this task. As a result, although the findings reported in our current paper seem promising, it is difficult to know the extent to which they are applicable to other NLP tasks or to tasks which differ only with respect to the test data used. This is one issue that we are interested in exploring in future work. Another is a test of whether extrinsic evaluation methods sensitive to information about the types of changes made in the simplification step would perform better than the black box methods used in the current paper.

[13] Information about the DUC conferences is accessible from https://www-nlpir.nist.gov/projects/duc/index.html (last accessed 22nd August 2018). Guidelines about the tasks presented in DUC-2004 are available at https://www-nlpir.nist.gov/projects/duc/guidelines/2004.html (last accessed 22nd August 2018).

References

Mandya Angrosh, Tadashi Nomoto, and Advaith Siddharthan. 2014. Lexico-syntactic text simplification and compression with typed dependencies. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Association for Computational Linguistics, Dublin, Ireland, pages 1996–2006.

Catherine Blake, Julia Kampov, Andreas K. Orphanides, David West, and Cory Lown. 2007. Uncch at duc 2007: Query expansion, lexical simplification and sentence selection strategies for multidocument summarization. In *Proceedings of the Document Understanding Conference (DUC-2007)*. National Institute of Standards and Technology.

Yvonne Canning. 2002. *Syntactic Simplification of Text*. Ph.d. thesis, University of Sunderland.

David Caplan and Gloria S. Waters. 1999. Verbal working memory and sentence comprehension. *Behavioural and Brain Sciences* 22:77–126.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems* 10:183–190.

Nancy Chinchor. 1992. The statistical significance of the muc-4 results. In *Proceedings of the Fourth Message Understanding Conference*. McLean, Virginia, pages 30–50.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.d thesis, University of Pennsylvania.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186.

Michael A. Covington. 2012. CPIDR® 5.1 user manual. Technical report, Institute for Artificial Intelligence, University of Georgia, Athens, Georgia, U.S.A.

Iustin Dornescu, Richard Evans, and Constantin Orasan. 2013. A Tagging Approach to Identify Complex Constituents for Text Simplification. In *Proceedings of Recent Advances in Natural Language Processing*. Hissar, Bulgaria, pages 221 – 229.

Larissa Sayuri Futino Castro dos Santos, Marcos Oliveira Prates, Gisele de Oliveira Maia, Guilherme Lucas Moreira Dias Almeida, Daysemara maria Cotta, Ricardo Cunha Pedroso, and Aurélio de Aquino araújo. 2018. Assessing if an automated method for identifying features in texts is better than another: discussions and results. Technical report, Department of Statistics, Universidade Federal de Minas Gerais. https://bit.ly/2xUD2BI.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language* 67:547–619.

Richard Evans. 2011. Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and Linguistic Computing* 26 (4):371–388.

Richard Evans and Constantin Orăsan. 2019. Identifying signs of syntactic complexity for rule-based sentence simplification. *Natural Language Engineering* 25 (1):69–119.

Dan Feblowitz and David Kauchak. 2013. Sentence simplification as tree transduction. In *Proceedings of the 2nd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1–10.

Robert Gaizauskas, Jonathan Foster, Yorick Wilks, John Arundel, Paul Clough, and Scott Piao. 2001. The meter corpus: A corpus for analysing journalistic text reuse. In *Proceedings of Corpus Linguistics 2001 Conference*. Lancaster University Centre for Computer Corpus Research on Language, pages 214–223.

Goran Glavas and Sanja Stajner. 2013. Event-centered simplication of news stories. In *Proceedings of the Student Workshop held in conjunctuion with RANLP-2013*. RANLP, Hissar, Bulgaria, pages 71–78.

Ralph Grishman. 2005. The Oxford Handbook of Computational Linguistics. Oxford University Press, chapter Information Extraction, pages 545–559.

Eva Hasler, Adrià de Gispert, Felix Stahlberg, and Aurelien Waite. 2017. Source sentence simplification for statistical machine translation. *Computer Speech & language* 45:221–235.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 680–687.

Timothy B. Jay. 2003. *The psychology of language*. Pearson, Upper Saddle Rive, NJ.

Tomáš Jelínek. 2014. Improvements to dependency parsing using automatic simplification of data. In *Proceedings of LREC-2014 the 22nd International Conference on Computational Linguistics (Coling 2008)*. European Language Resources Association, Reykjavik, Iceland, pages 73–77.

Siddhartha Jonnalagadda, Luis Tari, Jorg Hakenberg, Chitta Baral, and Graciela Gonzalez. 2009. Towards effective sentence simplification for automatic processing of biomedical text. In *Proceedings of NAACL HLT 2009: Short Papers*. Association for Computational Linguistics, Boulder, Colorado, pages 177–180.

Walter Kintsch and David M. Welsch. 1991. The construction–integration model: A framework for studying memory for text. In W. E. Hockley and S. Lewandowsky, editors, *Relating theory and data: Essays on human memory*, Hillsdale, NJ: Erlbaum, pages 367–385.

Sigrid Klerke, Héctor Martínez Alonso, and Anders Søgaard. 2015. Looking hard: Eye tracking for detecting grammaticality of automatically compressed sentences. In *NODALIDA*. Linköping University Electronic Press / ACL, pages 97–105.

Vladimir Iosifovich Levenshtein. 1966. Binary Codes Capable of Correcting Deletions and Insertions and Reversals. *Soviet Physics Doklady* 10 (8):707–710.

Danielle S. McNamara, Arthur C. Graesser, Philip M. McCarthy, and Zhiqiang Cai. 2014. *Automated Evaluation of Text and Discourse with Coh-Metrix*. Cambridge University Press.

Christina Niklaus, Bernhard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Osaka, Japan, pages 170–174. https://www.aclweb.org/anthology/C16-2036.

Constantin Orăsan, Richard Evans, and Ruslan Mitkov. 2018. Intelligent text processing to help readers with autism. In Khaled Shaalan, Aboul Ella Hassanien, and Mohammed F. Tolba, editors, *Intelligent Natural Language Processing: Trends and Applications*, Springer, pages 713–740.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106. https://doi.org/10.1162/0891201053630264.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, pages 311–318.

Yifan Peng, Catalina O. Tudor, Manabu Torii, Cathy H. Wu, and K. Vijay-Shanker. 2012. iSimp: A sentence simplification system for biomedical text. In *Proceedings of the 2012 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE, Philadelphia, PA, pages 1–6.

Dragomir Radev, John Blitzer, Adam Winkel, Tim Allison, and Michael Topper. 2006. MEAD documentation v3.10. Technical report, University of Michigan.

Carolina Scarton, Alessio Palmero Aprosio, Sara Tonelli, Tamara Martín Wanton, and Lucia Specia. 2017. MUSST: A Multilingual Syntactic Simplification Tool. In *The Companion Volume of the IJCNLP 2017 Proceedings: System Demonstrations*. Taipei, Taiwan, pages 25–28.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '04. https://doi.org/10.3115/1220355.1220484.

Linnea B. Timm. 2018. *Looking at text simplification - Using eye tracking to evaluate the readability of automatically simplified sentences*. Bachelor thesis, Institutionen fr datavetenskap, Linkpings universitet.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*. Columbus, Ohio, pages 344–352.

Tu Thanh Vu, Giang Binh Tran, and Son Bao Pham. 2014. *Learning to Simplify Children Stories with Limited Data*, Springer, Bangkook, Thailand, pages 31–41.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-12)*. Association for Computational Linguistics, Jeju, Republic of South Korea, pages 1015–1024.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL* 4:401–415. https://bit.ly/2Sj5mag.

Menahem Yeari, Paul van den Broek, and Marja Oudega. 2015. Processing and memory of central versus peripheral information as a function of reading goals: evidence from eye-movements. *Reading and Writing* 28 (8):1071–1097.

OlloBot - Towards A Text-Based Arabic Health Conversational Agent: Evaluation and Results

Ahmed Fadhil
Department of Computer Science
Universita Degli Studi di Trento
Trento, Italy
ahmed.fadhil@unitn.it

Ahmed AbuRa'ed
Information & Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
ahmed.aburaed@upf.edu

Abstract

We introduce OlloBot, an Arabic conversational agent that assists physicians and supports patients with the care process. It doesn't replace the physicians, instead provides health tracking and support and assists physicians with the care delivery through a conversation medium. The current model comprises healthy diet, physical activity, mental health, in addition to food logging. Not only OlloBot tracks user daily food, it also offers useful tips for healthier living. We will discuss the design, development and testing of OlloBot, and highlight the findings and limitations arose from the testing.

1 Introduction

According to World Health Organisation (WHO), poor diet and physical inactivity have tremendous implications on individuals health (Michie et al., 2009; Hard et al., 2012). Healthy diet helps protect against malnutrition in all its forms, as well as noncommunicable diseases (NCDs) (Michie et al., 2009). Middle eastern countries are among the mostly affected nations by poor diet and physical inactivity and its effect on cardiovascular diseases (Organization et al., 2010, 2012). According to WHO report, total deaths caused by NCDs are among the highest in countries such as, Saudi Arabia, Iraq, Qatar, and Bahrain (See Table-1). This is an evident for a global risk and there is a need for nationwide approach to help mitigate or prevent this escalation.

By 2025, AI systems could be involved in everything from population health management, to virtual assistants capable of answering specific patient queries. Overall, AI has the potential to improve outcomes by 30 to 40 percent while cutting treatment costs by as much as 50 percent (Koh et al., 2011). Conversational agents, an example of AI powered systems can communicate with users through an intelligent conversation using a natural language, they can aid doctors in enhancing productivity and enabling them to respond to patients quickly. For example, doctors could interact with a diet chatbot to recommend the appropriate food to their patients. Patients could also engage with this chatbot to get instant information about their dietary choices.

The NCDs and Middle East (WHO)	
Countries	% of total deaths by NCDs
Saudi Arabia	*Cardiovascular diseases = 46%*
Qatar	*Cardiovascular diseases = 24%*
Iraq	*Cardiovascular diseases = 33%*
Bahrain	*Cardiovascular diseases = 26%*

Table 1: Middle East Countries and NCD Burden.

Currently, some ways to use healthcare chatbots include: scheduling doctor appointments based on the severity of the symptoms, monitoring the health status and notifying a human nurse immediately if the parameters are out of control, helping home-care assistants stay informed about patients evolution. However, current health conversational agents are mature for English language but still in their infancy for specific demographics and languages. For example, Hebrew and Arabic are far more complex languages for conversational agents. To ensure timely health information delivery to Arabic users, the patient should be able to tell the bot what symptoms they are experiencing and receive medical advice. However, this is not possible with current approaches, since they merely focus on generic approaches. Hence, in this research we highlight the role of chatbots to provide health services to individuals with language barriers. The approach targets demographics who are not bilingual and can speak only Arabic language. This is important, since from Table-1 we see the burden of NCDs in Arabic speaking countries. We propose OlloBot, an Arabic conversational agent able to converse with users and handle daily tasks about diet, physical activity, mental wellness and coping. The bot can also track users daily food and keep a record of their daily dietary habits. To our knowledge, few studies exists on conversational agents that supports Arabic language, able to flexibly converse and handle dialogues. However, no study exists that have considered the health benefit achievable with language specific conversational agents. Few platforms provide support to Arabic language technology to build interactive and intelligent conversational agents. This might be due to the complexity of the language and the resource scarcity available to support it. Our approach relies on IBM Watson Conversation API (IBM bluemix)[1] to handle the dialogue structure and Telegram Bot Platform to build the chatbot. We tested OlloBot with 43 Arabic speaking users and presented the findings in this paper.

2 Background Research

Although cognitive behavioral therapeutic (CBT) apps have demonstrated efficacy, still they are characterized by low adherence rate (Fitzpatrick et al., 2017). Conversational agents, on the other hand, may offer a convenient and engaging alternative of giving support at any time. A work by Graf et al., (Graf et al., 2015) built a chatbot that assimilates into daily routines. The bot communicates with user and gathers nutrition data. Keeping interaction with the bot final is a good design practice, however building a great dialogue flow is also essential to ensure smooth user interaction. This includes the way the bot handles several user interactions and requests. Zaghouani et al., (Zaghouani et al., 2015) presented a correction annotation guidelines to create a manually corrected nonnative (L2) Arabic corpus. The work extends a large scale Arabic corpus and its manual corrections to include manually corrected non-native Arabic learner essays. The approach uses annotated corpus to develop components for automatic detection and correction of language error to help standard Arabic learners and improve the quality of Arabic text produced. Chatbots could be used as language learning tools, to access information, to visualise the context of a corpus and to give answers to specific questions (Abu Shawar, 2005). A work by Shawar et al., (Abu Shawar, 2005) built a general language chatbot that supported different language among which is Arabic and English. The study also used different corpora structure, such as dialogue, monologue and structured text to build the dialogue model.

A work by Ali et al., (Ali and Habash, 2016) presented BOTTA, an Arabic dialect chatbot that communicates with users using the Egyptian Arabic dialect. Another work by Shawar et al., (Shawar and Atwell, 2009) described a technique to access Arabic information using chatbot with natural language processing (NLP) tools. The bot provides responses to capture the logical ontology of a given domain using a set of pattern-template matching rules. The literature shows no studies considering conversational agent as assistive tool for Arabic speakers. Which could provide instant access to health information and data, and assist physicians in providing a follow up to the patients. Conversational agents are great in handling repetitive tasks which consumes most of the healthcare providers time. Sundermeyer et al., (Sundermeyer et al., 2014) build a two translation recurrent neural models. The first one is a word-based approach using word alignments, while the second presents phrase-based translation models that are more consistent with phrase-based decoding. The models are capable of improving strong baselines in BOLT task for Arabic to English.

3 OlloBot Architecture

To provide the required level of Natural Language Understanding (NLU), we built the Arabic dialogue states on IBM Watson Conversation[2], which defines the conversation flow and dialogue states. The platform provides Artificial Intelligence support to catch different user intents and entities. The OlloBot starts with the user sending a message to the bot (OlloBot) running on Telegram Bot Platform[3]. The application provides the user with the topics they can chat about, then based on user selection the bot forwards the request to IBM Watson Conversation cloud. The conversation slots takes user input and provides them with

[1] https://www.ibm.com/cloud-computing/bluemix/it

[2] https://www.ibm.com/watson/services/conversation/

[3] https://telegram.org/blog/bot-revolution

relevant answer after checking their intent, entity and condition of the conversion. The logical parts of the dialogue are handled by a Node.js wrapper to handle unmatched dialogues by the dialogue flow (Figure-1 illustrates the high-level architectural view of OlloBot).

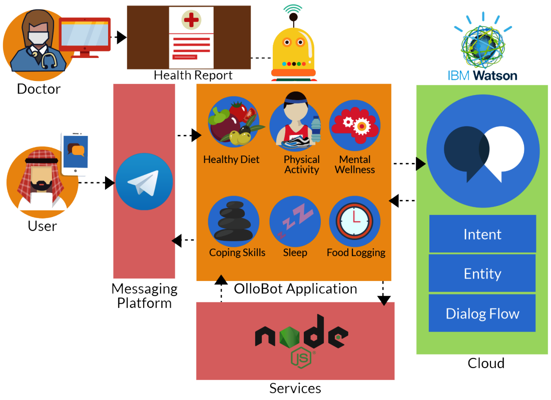


Figure 1: OlloBot High-level Architecture.

The bot detects patterns in user phrasing and correlates them with the intents defined. For example, conversing about users daily meal and asking users to enter their breakfast, they can either provide a list of food items or send an emojis as a list. The bot can extract the intend from each item or emoji and build a list of food items. In principle, this induces the system to recognize food items inserted by users. The need for conversational agents rises due to the on-demand 24/7 access to care. However, with millions of people globally who struggle with poor diet, poor exercise, anxiety and depression, there isn't enough healthcare provider or psychologists to provide the necessary care. Moreover, in some parts, there is limited technological services supporting native languages and sometimes services are practically nonexistent. Due to the associated cost for care, providing conversational agents as a novel technology that is cost effective, efficient and targets the right demographics is essential. OlloBot bridges these gaps and provides Arabic users with a quick and accessible health services. The bot converses about diet, exercise, emotion and provide coping skills. It can also ask the user to log their daily food and gather data about their preferred style of diet and exercise day, exercise, sleep. The fact that OlloBot is supported by Watson and easily accessible by many users makes it able to support many users at the same time.

The chatbot gathers users interaction data, and performance indicators from their interaction. These data are saved in a database in the form of accessible reports by the caregivers. OlloBot acts as a health assistants, meaning that it offers help and support rather than treatment.

3.1 User Intents

The intents refers to what the users want from the bot or what's the objective behind a user's input. For instance, the user intents hi, hello, hi OlloBot are translated to the intent "Greetings", whereas the intents food logging, daily food, logging data...etc are all translated into "Food Logging". We carefully sketched the dialogue scope the bot covers and those that it does not (see Table-2 for a list of user intents). The dialog flow structure was designed in collaboration with a healthcare clinic. After defining the flows for key intents, we defined the bot responses or follow up, once the task is performed. When the user starts a conversation, the bot can either track user dialogue or leave it at resolution and reset. The bot can switch between intends, based on user input. Since the interaction medium is conversational, users can switch intents on the chatbot. For example, while the bot waits for the user to provide relevant information about their physical activity, the user can ask the bot to insert their breakfast. OlloBot switches between the topics given in the above table, by detecting user intent from the conversation. To accurately handle this switching, we designed additional flows represented by conditions. Although this adds flexibility to the interaction, however, it can also create additional cognitive load.

User Intents	
Greetings	*Coping Skills*
Diet and nutrition	*Daily meals Logging*
Mood	*Jokes*
Physical activity	*Conditions*
Anything else	*Farewells*

Table 2: The User Intents

The next bot response depends on what the user will say or choose from the buttons. For example, after the user provides their daily food logging, the bot can move on and ask them about their suitable time to be notified again. Users can come back to the chatbot at a later time, with no recollection of the task they were trying to accomplish, therefore tracking the dialogue flow is essential to offer a more flexible conversation to the users. Finally, to handle fallbacks where the bot has no clue about the respond, we designed the “anything else” intent, to handle unhandled intents.

3.2 Health Report

The chatbot application provides a health report generated by users interaction with the chatbot. This report contains user activity data, their overall interaction with the application and indications about their overall health. The health measure is mainly about their diet, exercise and stress pattern. All these data are structurally generated by the chatbot and saved into the database. These data can then be accessed by the caregiver to obtain relevant user specific data.

3.3 Entities

Entities are the pieces of valuable information hidden in users input. They’re important keywords extracted from a sentence. For example, in the utterance “I want to talk about physical activity”, the word “physical activity” is detected as an entity, and hence the bot switches to the physical activity topic. Entities focus on defining the topic the user is talking about. This is important to provide the right respond to user questions. We defined a big range of entities to build our dialogue model. These entities represented the four main topics and the daily food logging. In Table-3 we provide the entities listed to structure the dialogue flow.

Topic Entities	
Topics	*Meal times*
Daily times	*Weekly times*
Quantity	*Food*
Numbers	*Sport*

Table 3: The Topic Entities

The above table lists the “Topics” entity, namely diet, physical activity, mood, and coping that the bot converses about. The topics entity also covers users daily food logging. Other entities include the “Meal times” which defines the meal periods per day that includes: breakfast, lunch, snacks and dinner. The “Daily times” refers to the periods of the day, namely morning, noon, afternoon, evening and night. Whereas, “Weekly times” entity includes the period of the day, namely yesterday, today and tomorrow. The “Quantity” entity refers to the quantity measurements the user might mention in the conversation (i.e., kg, g, tea spoon, bread loaf), whereas the “Numbers” entity refers to the countable number the user might mention. We have also defined a comprehensive list of common Arabic food items and included them in the “Food” entity. Finally, we defined a list of any kind of sports in the “Sport” entity. The entity list was accompanied by a list of synonym to add flexibility for the entity detection. This is important to detect user intents and sentiments from their conversation. While there has been a lot of research on sentiment analysis in English, the amount of research and datasets for Arabic language is still limited. A work by Alayba et al., (Alayba et al., 2017) built a sentiment analysis dataset in Arabic from Twitter data. The study applied machine learning algorithms (i.e., Naive Bayes, Support Vector Machine and Logistic Regression) for the analysis. Another work by Ismail et al., (Ismail and Ahmad, 2004) proposed a new type of recurrent neural network architecture for speech recognition and Backpropagation Through Time (BPTT) learning algorithm to observe differences in alphabet “alif” until “ya”.

3.4 Dialogue Engine

The dialogue structure was designed for each of healthy diet, physical activity, and mental wellness topics by referencing the Cognitive-behavioural therapy (CBT) (Rothbaum et al., 2000). We based the chatbot dialogue on techniques suggested by the CBT.

The dialogue was then developed on IBM Watson Conversation, where we listed each of the intents and entities and built the dialogue structure and flow. The tasks were intentionally built to be simple to interact with, so to decrease the time and amount of physical and mental efforts needed, and increase user’s ability. For example, when asked to log their food, users can either write the list or send an emoji of the food item. Moreover, button replies were provided to further simplify the data logging. Simplifying food tracking process will increase users ability and decrease the learning curve associated to health tracking. This is because interaction with the bot shouldn’t be only conversational, since some interactions are better with Graphical UI and others with Conversational UI.

3.4.1 Healthy Diet

Around 2 billion people are overweight, but many are ready to change (Van Itallie, 1985). However, according to studies (Mann et al., 2007) temporary fixes to old habits makes people to regain their weight. OlloBot acts as an interactive AI-powered diet tracking bot that converses with individuals in a friendly way directly through the Telegram messaging application. The goal is to give instant advices with each meal eaten and help improve the eating habits on the go. Once the conversation is executed, OlloBot starts conversing about user's diet and asks them questions about their eating habits and highlights the values associated with healthy diet. For example, OlloBot stresses the fact that following a diet rich in vegetables and fruits helps decrease escalation into overweight, obesity or even chronic conditions, and the detrimental effects associated otherwise.

3.4.2 Physical Activity

Studies (Cooney et al., 2014; Mead et al., 2009; Artal et al., 1998; Byrne and Byrne, 1993) have shown that exercise can treat mild to moderate depression as effectively as antidepressant medication and with no associated side-effects. Our conversational agent chats with users about their physical activity and daily energy level, and can provide personalized plans and keep track of workout progress by storing relevant user inputs. Whether the user wants to stay fit, loose weight, or get toned, OlloBot can later provide an efficient and consistent workout plan while keeping track of the progress. Although the bot is not considered as a tool to loose weight through exercise, but rather a supportive tool to help individuals track their exercise and improve in on the go, making users conscious about their health habits also makes them more likely to set aside time for physical activities in the future.

3.4.3 Mental Wellness

Mental health refers to our overall psychological well-being. This includes the way we feel about ourselves, the quality of our relationships and the ability to manage our feeling. OlloBot chats with users about mental health and asks them questions about their stress, sleep and other measures relevant to their mental wellbeing. There exists several mental health chatbots that provide support at different stages of mental illness. For example, X2AI[4] created a set of chatbots for mental health applications. Their flagship AI, Tess, helps patients in tandem with their doctors by providing resources on cognitive-behavioral therapy, medication side effects, and questionnaire automation. Woebot[5] is a mood tracking chatbot with personality. Backed by scientific research, Woebot can help reduce depression, share CBT resources, and learn from conversations over time. Finally, Joy[6] uses a chatbot approach to mental health. She offers options for both individuals and therapists.

3.4.4 Coping Skills

Being mentally or emotionally healthy is more than being free of depression, anxiety, or other psychological issues. Rather than the absence of mental illness, mental health refers to the presence of positive characteristics. With OlloBot, we handle this by providing coping skills and mindfulness support through clever motivational quotes, relevant workout suggestions mixed with some health facts to help users understand the benefit of health and wellness. The bot provides the uses with tips about coping with stress, diet, exercise, sleep, and mindfulness. The quotes and suggestions are all based on best recommendations provided by the World Health Organisation (WHO) (Michie et al., 2009). Coping skills are important, since finding a moment to take a few deep breaths and quiet your mind is a great way to relieve stress and improve your overall health.

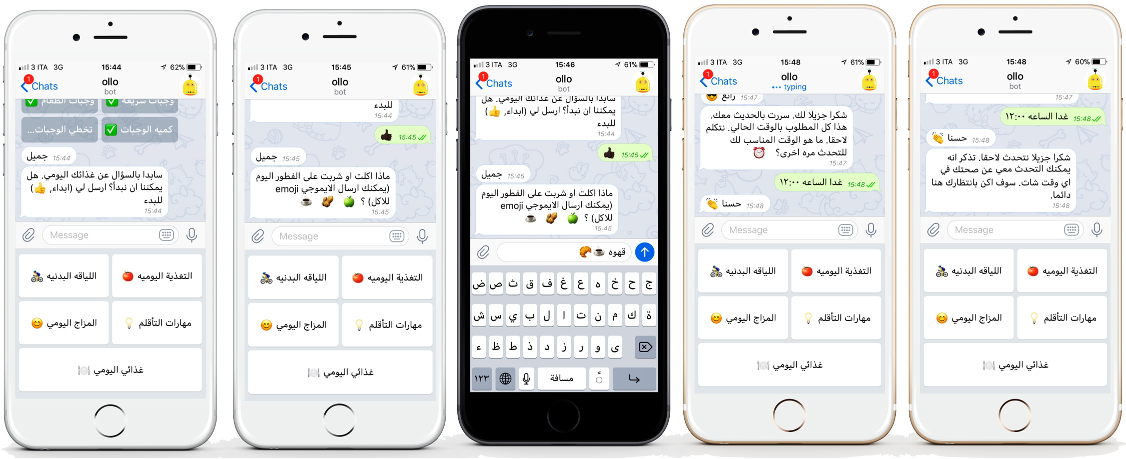
3.4.5 Daily Food Logging

While many food-logging tools are already on the market, most are either too complicated or boring for the average individual. OlloBot keeps a food diary, tracks calories, and provides basic nutritional tips based on user's eating habits. We wanted to make the food logging process simpler, more engaging, and more informative for the users. The bot tracks user meals, by asking them to insert each meal items. This helps to know more about users diet. With the NLP capabilities offered by IBM Watson Conversation, the bot supports any food combinations. The user can even log their food by sending emojis of the food items. Once the food logging is done, the bot asks the user about their preferred time to recheck with them. OlloBot is in its early stages, due to the time

[4] https://x2.ai/

[5] https://woebot.io/

[6] http://www.hellojoy.ai/



وجبات الطعام ✅ وجبات سريعة ✅
تخطي الوجبات... كميه الوجبات ✅

جميل 15:44

سابدا بالسؤال عن غذائك اليومي. هل يمكننا ان نبدأ؟ ارسل لي (ابداء, 👍) للبدء 15:44

للبدء 15:44

👍 15:45

جميل 15:45

ماذا اكلت او شربت على الفطور اليوم (يمكنك ارسال الايموجي emoji للاكل) ؟ 🍏 🥜 ☕ 15:45

سابدا بالسؤال عن غذائك اليومي. هل يمكننا ان نبدأ؟ ارسل لي (ابداء, 👍) للبدء 15:44

👍 15:45

جميل 15:45

ماذا اكلت او شربت على الفطور اليوم (يمكنك ارسال الايموجي emoji للاكل) ؟ 🍏 🥜 ☕ 15:45

قهوه ☕ 🥐

رائع 😎 15:47

شكرا جزيلا لك. سررت بالحديث معك. هذا كل المطلوب بالوقت الحالي. نتكلم لاحقا. ما هو الوقت المناسب لك للتحدث مره اخرى؟ ⏰ 15:47

غدا الساعه ١٢:٠٠ 15:48

حسنا 👏 15:48

15:47

غدا الساعه ١٢:٠٠ 15:48

حسنا 👏 15:48

شكرا جزيلا نتحدث لاحقا. تذكر انه يمكنك التحدث معي عن صحتك في اي وقت شات. سوف اكن بانتظارك هنا دائما. 15:48

Figure 2: Daily Food Logging Conversation with OlloBot.

required to build and integrate new features, then validate it through testing with real users. We understand that existing food logging apps are now far ahead of us in terms of calorie tracking, precision, and various interface features. However, we aim to close this gap and simplify the food tracking/logging and feedback providing process in the future (see Figure-2 for the daily food logging with OlloBot).

4 OlloBot Platform

The dialogue flow in OlloBot were handled with IBM Watson Conversation, whereas the logical part of the dialogue were handled with a Node.js implementation. Our choice of IBM Watson was due to its language support, to our knowledge it was the only conversational agent dialogue building platform that supports Arabic. We deployed the bot on Telegram Bot Platform and built the components to handle user requests and responses. We used various UI elements made available by Telegram and integrated them into our dialogue model.

5 OlloBot - USE Questionnaire

After building and integrating OlloBot with Telegram Bot Platform, we conducted a user experiment with 43 Arabic speaking users. The dialogue and various questions the bot asks were all based on a WHO questionnaire we extracted about health and wellbeing (Kessler and Üstün, 2004; Parslow and Jorm, 2000; , WHO et al.(2017; Organization et al., 2006). After testing the chatbot, we performed a survey analysis to test various aspects of the bot. The questionnaire is based on a standard framework, namely USE Questionnaire (Lund, 2001) to measure the usability of the chatbot. The framework tests four items of usability within a product. It consists of 30 questions to test the usefulness, easy of use, ease of learning and satisfaction with the application. The questionnaires are organised in a scale of 1 - 5, where 1 is *“strongly disagree”* and 5 is *“strongly agree”*. We describe the experiment settings and results in the following sections.

5.1 Participants Demographics

The user demographics consisted of both male (n=26) and female (n=17) participants with an age range of 20 - 65 years old for both gender. We checked the participants familiarity with tracking devices (e.g., any diet, sleep, physical activity, or mood tracking application) and chatbot applications (see Table-4 for the results). There was no significant differences in both cases with either tracking devices (e.g., wearable, sensor, or mobile applications) or chatbot applications. The majority of participants have shown no familiarity with tracking devices (n=26) and chatbot applications (n=27). We provided additional questions at the end of the survey to evaluate their overall experience with the chatbot. A reimbursement of 5€was given to all participants in the form of Amazon coupon.

5.2 Experiment

We distributed the chatbot to all the Arabic speaking participants recruited from Iraq. After carrying out the experiment for 1 week, we collected data

Gender	Female	*17*
	Male	*26*
Age	Mean	*29.8*
	Std. Dev	*9.28*
Tracking devices familiar	No	*26*
	Yes	*17*
Chatbot familiar	No	*27*
	Yes	*16*

Table 4: Participants Demographics.

through questionnaire to test the four main points relevant to the usability of OlloBot.

5.2.1 Usefulness

This item helps to test how useful the participants perceived the application. The usefulness includes measuring whether the bot helps the user to be more effective and productive and enhance their control over their daily life activities. For example, *"I believe is effective to track my health"* evaluates whether the user thinks the bot is effective in tracking their health.

Dimensions	**Mean**	**Std. Dev.**	**t value (df=42)**	**p value**
Usefulness	*3.381*	*0.373*	*6.695*	*p <.01*
Ease Of Use	*3.626*	*0.3944*	*10.405*	*p <.01*
Learning	*3.942*	*0.5341*	*11.564*	*p <.01*
Satisfaction	*3.505*	*0.3667*	*9.031*	*p <.01*

Table 5: The Results from the USE Questionnaire.

5.2.2 Ease of Use

This point helps understand the ease of use aspect. It measures easiness, simplicity and user friendliness of the application (chatbot). This point also measures the steps and effort required to achieve the goal set and whether its easy to recover from mistakes. For example, to measure the effort required for each step, we asked users to provide their scale for *"It requires the fewest steps possible to accomplish what I want to do with it"*. We performed a descriptive statistics about the four items checked in the overall experience (see Table-6). Figure-3 below lists each of the overall bot reliability (Figure-3a), and the participants overall experience (Figure-3b, Figure-3c and Figure-3d).

5.2.3 Ease of Learning

This point measures the learnability of the tool. We measure whether its quick to learn and easy to remember each time. For example, the question *"I quickly became skilful with it"* checks for how quickly a skill is learned using the chatbot.

5.2.4 Satisfaction

This step checks for user's overall satisfaction with the chatbot. It checked whether the users are satisfied and would recommend the tool to others and how entertaining they perceived it. For example, the question *"I would recommend it to a friend or family member"* checks whether the user would recommend the bot to their relatives or friends.

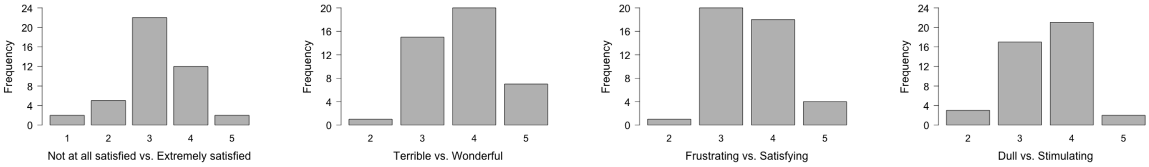
We performed descriptive statistics on the results and reported them in Table-5. One Sample t test showed that averages for all four scales were statistically significantly different from the middle value (value = 3, see Table-5). Further analyses were performed considering gender, familiarity with tracking technology, and familiarity with chatbot as between factors. No differences among the four usability dimensions were observed between male and female participants, and no differences emerged between participants who frequently use or not use tracking devices. A significant difference for the "ease of learning" scale was observed when comparing participants who were familiar in interacting with chatbot application and participants who were not. The latter group reported lower scores for learnability compared to the former group (t(41)= -2.46, p <.01).

5.2.5 Overall Experience

This part evaluated the overall experience with OlloBot and is not part of the USE questionnaire. We tested user satisfaction with the reliability of OlloBot, their overall experience with the bot from different perspectives (see Table-7 for the experience comparison). We have checked whether users like chatbots or rather use a mobile application. Finally, we considered the list of most positive and negative aspects the users mentioned during the survey. In addition, we asked the users about the features they would like to see/use in feature versions of OlloBot.

6 Discussion

This work was the first to evaluate the application of NLP powered health tools into language and context scarce domains. The work involved initial design phase, which involved researchers and health experts in the context of healthy lifestyle



(a) Overall Reliability Satisfaction.

(b) Participants Overall experience.

(c) Participants Overall experience.

(d) Participants Overall experience.

Figure 3: The Overall Experience with OlloBot.

	App vs. Chatbot	**Not at all satisfied vs. Extremely satisfied**	**Terrible vs. Wonderful**	**Frustrating vs. Satisfying**	**Dull vs. Stimulating**
Valid	43	43	43	43	43
Mean	3.860	3.163	3.767	3.581	3.512
Std. Dev.	0.9900	0.8710	0.7508	0.6980	0.70028
Min	1.000	1.000	2.000	2.000	2.000
Max	5.000	5.000	5.000	5.000	5.000

Table 6: The Descriptive Statistics for Chatbot Experience.

Overall Experience		
Questions	Scale = 1	Scale = 5
How satisfied are you with the reliability of this chatbot?	*Not at all satisfied*	*Extremely satisfied*
Can you rate your experience with the chatbot:	*Terrible*	*Wonderful*
Can you rate your experience with the chatbot:	*Frustrating*	*Satisfying*
Can you rate your experience with the chatbot:	*Dull*	*Stimulating*
Would you rather use the chatbot or prefer a mobile app to track your everyday health ?	*Use an app*	*Use a chatbot*

Table 7: Users Overall Experience With OlloBot.

promotion. Involving the health expert was necessary to guarantee the real context applicability of the application. The dialog structure was built in collaboration with a dietitian who described the steps necessary when building a dialog with a patient. We then applied this design paradigm into our dialog engine. The existing work has several limitations worth mentioning. Since most of the research work was carried out in Italy in collaboration with Spain, the dietitian involved in designing the dialog was a native Italian. Yet, the guidance obtained from the expert was translated into Arabic language and integrated into the Watson dialog engine. We also acknowledge that the size of the participants and the period of experiments, although provide good indications, are not enough to conclude any long-term effect. Future work will consider building a standalone dialog model and including a larger user samples and over an extended period.

7 Conclusion

Chatbots can be a trustworthy assistant, like a caring nurse, which provides registration services and patient follow up. Medical practices can rely on chatbots to capture leads and provide 24/7 support to existing patients, answering their simple, repetitive questions using a pre-designed answers. It will not offer a diagnosis, but it can remind patients to take their drugs or help them check for an unusual side effect. Most users showed interest towards social intelligence of the bot. Hence, tone and empathy matters, people can be turned off if the experience is too robot-like or casual, so we should opted for a polite tone. In summary, chatbots offer a great user experience to patients by just chatting with the bot to get relevant answers to their queries. Future work will integrate OlloBot into a health coaching system and make it specific to support Arabic speaking, diabetic patients.

References

Bayan Aref Abu Shawar. 2005. *A corpus based approach to generalise a chatbot system*. Ph.D. thesis, University of Leeds.

Abdulaziz M Alayba, Vasile Palade, Matthew England, and Rahat Iqbal. 2017. Arabic language sentiment analysis on health services. *arXiv preprint arXiv:1702.03197* .

Dana Abu Ali and Nizar Habash. 2016. Botta: An arabic dialect chatbot. In *COLING (Demos)*. pages 208–212.

Michal Artal, Carl Sherman, and Nicholas A DiNubile. 1998. Exercise against depression. *The Physician and sportsmedicine* 26(10):55–70.

A Byrne and DG Byrne. 1993. The effect of exercise on depression, anxiety and other mood states: a review. *Journal of psychosomatic research* 37(6):565–574.

Gary Cooney, Kerry Dwan, and Gillian Mead. 2014. Exercise for depression. *Jama* 311(23):2432–2433.

Kathleen Kara Fitzpatrick, Alison Darcy, and Molly Vierhile. 2017. Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial. *JMIR Mental Health* 4(2):e19.

Bettina Graf, Maike Krüger, Felix Müller, Alexander Ruhland, and Andrea Zech. 2015. Nombot: simplify food tracking. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. ACM, pages 360–363.

Eat Smart Play Hard, Healthy Kansas Schools, In-Class Physical Activity Breaks, and Blast Off Game. 2012. Physical activity. *Fitness and health: Internatio* .

Saliza Ismail and A Ahmad. 2004. Recurrent neural network with back propagation through time algorithm for arabic recognition. *Proceedings of the 18th ESM Magdeburg, Germany* pages 13–16.

Ronald C Kessler and T Bedirhan Üstün. 2004. The world mental health (wmh) survey initiative version of the world health organization (who) composite international diagnostic interview (cidi). *International journal of methods in psychiatric research* 13(2):93–121.

Hian Chye Koh, Gerald Tan, et al. 2011. Data mining applications in healthcare. *Journal of healthcare information management* 19(2):65.

Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8(2):3–6.

Traci Mann, A Janet Tomiyama, Erika Westling, Ann-Marie Lew, Barbra Samuels, and Jason Chatman. 2007. Medicare's search for effective obesity treatments: diets are not the answer. *American Psychologist* 62(3):220.

Gillian E Mead, Wendy Morley, Paul Campbell, Carolyn A Greig, Marion McMurdo, Debbie A Lawlor, et al. 2009. Exercise for depression. *Cochrane Database Syst Rev* 3.

Susan Michie, Charles Abraham, Craig Whittington, John McAteer, and Sunjai Gupta. 2009. Effective techniques in healthy eating and physical activity interventions: a meta-regression. *Health Psychology* 28(6):690.

World Health Organization et al. 2006. Global strategy on diet, physical activity and health: a framework to monitor and evaluate implementation .

World Health Organization et al. 2010. *WORLD HEALH REPORT (The): Health Systems Financing: the path to universal Coverage (Arabic)*. World Health Organization.

World Health Organization et al. 2012. World health day 2012: ageing and health: toolkit for event organizers .

Ruth A Parslow and Anthony F Jorm. 2000. Who uses mental health services in australia? an analysis of data from the national survey of mental health and wellbeing. *Australian and New Zealand Journal of Psychiatry* 34(6):997–1008.

Barbara Olasov Rothbaum, Elizabeth A Meadows, Patricia Resick, and David W Foy. 2000. Cognitive-behavioral therapy. .

B Abu Shawar and Eric Atwell. 2009. Arabic question-answering via instance based learning from an faq corpus. In *Proceedings of the CL 2009 International Conference on Corpus Linguistics. UCREL*. volume 386.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *EMNLP*. pages 14–25.

Theodore B Van Itallie. 1985. Health implications of overweight and obesity in the united states. *Annals of internal medicine* 103(6_Part_2):983–988.

World Health Organization (WHO et al. 2017. Obesity and overweight factsheet from the who. *Health* .

Wajdi Zaghouani, Nizar Habash, Houda Bouamor, Alla Rozovskaya, Behrang Mohit, Abeer Heider, and Kemal Oflazer. 2015. Correction annotation for non-native arabic texts: Guidelines and corpus. In *LAW@ NAACL-HLT*. pages 129–139.

Developing the Old Tibetan Treebank

Christian Faggionato
SOAS, University of London
cf36@soas.ac.uk

Marieke Meelen
University of Cambridge
mm986@cam.ac.uk

Abstract

This paper presents a full procedure for the development of a segmented, POS-tagged and chunk-parsed corpus of Old Tibetan. As an extremely low-resource language, Old Tibetan poses non-trivial problems in every step towards the development of a searchable treebank. We demonstrate, however, that a carefully developed, semi-supervised method of optimising and extending existing tools for Classical Tibetan, as well as creating specific ones for Old Tibetan, can address these issues. We thus also present the very first Tibetan Treebank in a variety of formats to facilitate research in the fields of NLP, historical linguistics and Tibetan Studies.

1 Introduction

In historical linguistics, there are currently two types of morpho-syntactically annotated corpora or 'Treebanks', one based on constituency parses, the other on dependency parses. The former includes pioneering work in the Penn-Helsinki tradition, resulting in the Old and Middle English (Taylor and Kroch, 1994), Icelandic (IcePaHC) (Rögnvaldsson et al., 2012) and Portuguese (Tycho Brahe) (Galves, 2018) treebanks. The latter represents syntactic structure in the form of dependencies and is often used for applied NLP tasks and early Indo-European languages, e.g. the PROIEL Treebank family (Eckhoff et al., 2018). In this paper we present a constituency-based treebank for an under-resourced language: Old Tibetan (see Green et al. (2012) and El-Haj et al. (2015) for similar examples of creating under-resourced treebanks).

Old Tibetan is an extremely under-resourced and under-researched language from an NLP point of view. We chose to focus our attention on the Old Tibetan corpus (7-11th c.) since it consists of a small collection of documents compared to the vast amounts of translated and original Classical Tibetan texts. Nonetheless, the Old Tibetan corpus is still heterogeneous enough to represent natural language. The majority of Old Tibetan texts (known to date; new inscriptions and texts are still being discovered) has now been digitised in one way or another (images, OCR and/or transcribed) and annotating this data is fundamental for the understanding of diachronic and synchronic issues in Tibeto-Burman languages. As the first attempt to create a Tibetan Treebank, developing a segmented, POS-tagged and chunk-parsed corpus of Old Tibetan provides new opportunities in Tibetan scholarly history, literature and linguistics.

Old Tibetan was the language spoken in the Yarlung Valley from where the Tibetan empire started its initial expansion. Writing was mainly introduced to facilitate administrative tasks, and the earliest Old Tibetan texts represent the most detailed sources for the history of early Tibet (Hill, 2010). The earliest currently available, securely datable Old Tibetan document dates to ca. 763 CE. However, the digital resources for Old Tibetan are inadequate (problematic transcriptions, transliterations and no digitised secondary resources such as dictionaries, etc.).

The core of the Old Tibetan corpus is available as plain e-texts (without segmentation or any kind of annotation) on the Old Tibetan Documents Online (OTDO) website.[1] We first focus on the Old Tibetan *Annals* (5.9k tokens)

[1] http://otdo.aa.tufs.ac.jp/

Proceedings of Recent Advances in Natural Language Processing, pages 304–312,
Varna, Bulgaria, Sep 2–4 2019.

and Old Tibetan *Chronicles*, since these are the best sources that we have at our disposal in terms of length and linguistic variety (many other Old Tibetan text are short inscriptions or more fragmentary). The Old Tibetan *Annals* are Tibet's earliest extant history. The Old Tibetan *Chronicle*, written in the early 9th century, is more narrative and includes historical accounts and songs related to the Yarlung dynasty and the Tibetan empire.

In this paper we present our annotation procedure that addresses all issues of pre-processing, segmentation, POS tagging and parsing in detail. Our semi-supervised method, resulting in the first Old Tibetan Treebank, can furthermore serve as an example of how to overcome challenges of low-resource and under-researched languages in general.

2 The Annotation Procedure

Since Old Tibetan is an extremely under-resourced language the procedure to develop an annotated corpus needs to be developed with great care. Additional steps are necessary at each of the normal stages, from pre-processing to POS tagging and parsing and finally post-processing. In the pre-processing stage, for example, the normalisation is not a trivial task because of a range of issues with the Tibetan script and the way it is digitised, in either Unicode or a variety of transliteration formats. In addition to solving these script issues, our core solution is to transform our Old Tibetan texts through a 'conversion/normalisation process' with a Constraint Grammar (Cg3) to a form of Tibetan that is closer to Classical Tibetan, for which at least some NLP tools are available.

Before we can move on to the annotation stage, we need to solve a further non-trivial issue of finding word and sentence boundaries. Since there are no Gold Standards or training data available for Old Tibetan, we resort to the little material and tools available for Classical Tibetan and then do a rigorous error analysis checking specific Old Tibetan features that we know differ from Classical Tibetan. Our annotation method is thus supervised in various ways to overcome the obstacles building a treebank of an extremely low-resourced language like Old Tibetan.

3 Pre-Processing

The Old Tibetan texts we work with to start this corpus are already transcribed from the original manuscripts or digitised images. For the present paper we thus only address the issues concerning encoding of transcriptions and transliterations and the issues of tokenising a language without word or sentence boundaries.

3.1 Transliteration Issues

One of the first challenges we encountered in creating the Old Tibetan corpus was the conversion from Tibetan Unicode script (see Hill 2012) to the Wylie transliteration system. There are few reliable tools available and in addition, we have to take the peculiar orthographic features of Old Tibetan into consideration. The Tibetan Unicode script for the Old Tibetan documents was obtained from a modified version of the Wylie transliteration system that is used for the Old Tibetan Documents Online (OTDO) website, through the BDRC conversion tool.[2]

However, this tool only partially addresses the issue, because we also want to transform Old Tibetan into a form of Tibetan that looks more similar to Classical Tibetan in terms of orthography. Therefore, the Wylie transliteration used by the OTDO website had to be modified. As an example the reverse 'i' vowel mark, ྀ - called *gigu* - is transliterated with 'I' on the OTDO website. We substituted 'I' with 'i', which is the standard Wylie transliteration for this character, as shown in (1):

(1) *rgyal po'I* > *rgyal po'i* 'of the king'

3.2 Normalisation

The Old Tibetan script furthermore presents a set of features that need to be 'normalised' or converted to a form that looks like Classical Tibetan. We therefore created a set of rules translated into the Constraint Grammar (Cg3) formalism. Most of the Cg3 rules used to normalise Old Tibetan are simple replacement rules. For example, In Old Tibetan there are many instances of the

[2] https://www.tbrc.org/

above-mentioned reverse *gigu* such as ཀྱྀ *kyI*. These two forms of *gigu*, ྀ and ི are phonetically indistinguishable and mark no difference in Classical Tibetan. The Cg3 `SUBSTITUTE` rule to normalise the reverse *gigu* is:

```
SUBSTITUTE (``([[\^{ }<]*)\u0F80(.*)"r)
(“$1ྀ$2v) TARGET (σ)
```

Two additional problems encountered in the normalisation of Old Tibetan are represented by the alternation between aspirated and unaspirated voiceless consonants and the difficulty of splitting merged syllables. This aspiration, however, was probably not phonemic in Old Tibetan (Hill, 2007, 471). Therefore, a set of string replacement rules in the Cg3 formalism was created to normalise and convert these instances to their equivalent reading in Classical Tibetan.

Furthermore in Classical Tibetan, syllables are separated by a punctuation marker called *tsheg*: ་. In Old Tibetan texts, syllable margins are not so clear and syllables are often merged together with the following case marker or converb, e.g. Old Tibetan བཀུམོ *bkumo* > Clas. Tib. བཀུམ་མོ *bkum mo* ‘kill, destroy’:

(2) བཀུམོ > བཀུམ་མོ

These types of merged syllables were also converted to their classical forms, using a set of three regular expressions in the Cg3 formalism through the rule `SPLITCOHORT`. Considering the complexity of the Tibetan syllable, in order to generate the rules, we took the maximum number of its constituents into account (in terms of vowels and consonants) as well as their order.

Generic Rule:

```
([^aeiouI\s]+[aeiouI][^aeiouI\s]*)
([^aeiouI\s'])([aeiouI][^aeiouI\s']*)
> $1$2 $2$3
```

Cg3 rule:

```
SPLITCOHORT (
  "<$1>"v "$1$3 "v
  "<$3$4>"v "$3$4"v
)("<(.{2,6})(([^\\u0FB2\\u0FB1])
([\\u0F7C\\u0F7A\\u0F74\\u0F72\\u0F80]
 ?))>"r)(NOT 0 (split) or (genitive)
 or (diphthongs));
```

Through these conversions and normalisations, we could apply existing tools for Classical Tibetan to our Old Tibetan corpus to avoid manually creating our treebank from scratch completely. The full Cg3 grammar is discussed in detail in our forthcoming research.

3.3 Segmenting Sentences

Segmenting sentences is necessary since there are no obvious sentence boundaries in Old Tibetan. The Tibetan scripts does have a punctuation marker that sometimes (but not always) indicates meaningful phrases, a so-called *shad*, ། or double *shad*, །།. Since without any further annotation, there is no way of knowing where sentences begin or end, we used the single and double *shad* as sentence boundaries and automatically inserted utterance boundaries indicators (`<utt>`) after every instance. This greatly facilitates subsequent annotation tasks that depend on sentence boundaries, such as POS tagging and chunkparsing.

3.4 Tokenisation

The Tibetan script furthermore does not indicate word boundaries. Tokenisation is therefore a tremendous issue, not only for scholars of Tibetan (who often disagree on what the word boundaries should be), but even more so for any Tibetan NLP tasks. The Classical Tibetan script does have a way of indicating syllable boundaries though, by using the above-mentioned *tsheg* marker ་, e.g. བྲག་མར transliterated *brag mar* ‘Dagmar’ with spaces between every syllable according to the conventions of the Wylie transliteration.

For Classical Tibetan, Meelen and Hill (2017) addressed this tokenisation issue by recasting it as a classification task with a memory-based tagger (Daelemans et al., 2003) giving ‘beginning’, ‘middle’ or ‘end’ labels to every syllable (automatically split based on the aforementioned *tsheg* and *shad* markers. With our supervised learning method first normalising and then converting our Old Tibetan corpus to a form of Tibetan that is much closer to Classical Tibetan, we were able to use this existing segmentation tool for Classical Tibetan and extend and modify them after manually correcting part of our Old Tibetan data.

4 POS Tagging

Since there was no Old Tibetan POS-tagged Gold Standard either, here too we started from the Classical Tibetan training data[3] and tagging method developed by Meelen and Hill (2017). We tested a number of ways to get and improve results for the Old Tibetan corpus, e.g. developing a new, reduced tag set, changing scripts (Unicode vs. Wylie) as well as generating new taggers, based on the manually corrected Old Tibetan only and, finally, adding the manually corrected Old Tibetan to the existing Classical Tibetan Gold Standard.

4.1 Small vs Large Tag Set

The tag set used for the Classical Tibetan Gold Standard, developed by Garrett et al. (2014) is with 79 morpho-syntactic tags rather large. This causes major issues for the out-of-vocabulary items, especially for languages without insightful morphological suffixes like Tibetan. For this first attempt of developing an Old Tibetan Treebank, we therefore decided to reduce the amount of tags to a small and simplified version of the standard Universal Dependency POS set, consisting of 15 tags only (De Marneffe et al., 2014). We transformed the existing Classical Tibetan training data, which is our Gold Standard, in the following way: `interj > INTJ`, `punc > PUNCT`, `n.prop > PROPN`, `skt, dunno > X`, `adj, num.ord > ADJ`, `n.v.cop, v.cop, v.cop.neg > AUX`, `n.count, n.mass, n.rel > NOUN`, `num.card, numeral > NUM`, `cl.focus, cv.fin, cv.imp, cv.ques, neg > PART`, `p.indef, p.interrog, p.pers, p.refl > PRON`, `d.dem, d.det, d.emph, d.indef, d.plural, d.tsam > DET`, and, finally, all verb remaining verb forms in all tenses `> VERB`, all remaining converbs `> SCONJ`, all post-positional case markers `> ADP` and all adverbs `> ADV`. A 10-fold cross-validation with the exact same parameter settings of the memory-based tagger[4] on the $>318k$ Classical Tibetan Gold Standard, yielded better results compared to those of the large tag set reported by (Meelen and Hill, 2017) (increase from 95.0% to 96.3% in Global Accuracy; Known Words increased from 96.8% to 97.8%; Unknown Words from 53.4% to 59.7%).

All tags with a very low number of tokens in the out-of-vocabulary set (ranging from $n = 1$-92) have a Precision and Recall close or equal to zero. These items are always very short (one or two characters only), which makes predicting the tag for new items in this category an almost impossible task for the tagger. With the newly trained small tag set tagger, we tagged the Old Tibetan *Annals* and manually corrected the first $3.5k$ tokens as a start. We then evaluated the tagger again with another 10-fold cross-validation, first on this small Old Tibetan corpus and then again adding this manually corrected Old Tibetan data to the existing Classical Tibetan Gold Standard. This yielded a better Global Accuracy for the combination of Old and Classical Tibetan (96.1%) compared to Old Tibetan alone (92.8%). However, the results for Unknown Words are significantly lower (decrease from 71.1% to 58.5%).

Since these two new Gold Standards differ significantly in size it is impossible to do a fair comparison until we manually correct more Old Tibetan. It is clear, however, that despite our efforts to normalise and convert the Old Tibetan into a form of the language that looks more like Classical Tibetan, it is still making a difference, shown in the lower accuracy (by more than 10%) of unknown words for this combined training data. Without adding the Classical Tibetan training data, however, the vocabulary list that the memory-based tagger builds would simply be too small to get any good results on unseen data. Despite the 10-fold cross-validation, the relatively high scores for the Old Tibetan corpus only are misleading, because of the small size of the corpus. Until we have more manually corrected Old Tibetan data, we therefore proceed with the Classical Tibetan Gold Standard and add an extra stage of error correction, see Section 6.

[3] `http://github.com/tibetan-nlp/soas-corpus/`

[4] These settings for Classical Tibetan are: `-p dwdwfWaw -P psssdwdwdwFawaw -M 1100 -n 5 -% 8 -O+vS -FColumns -G K: -a0 -k1 U: -a0 -mM -k17 -dIL.`

4.2 Unicode vs Wylie Transliteration

The above-mentioned taggers were trained and tested on Tibetan script in Unicode. The Unicode Tibetan script contains a lot of so-called 'stacked' characters that are centred before, above and below one single root letter. A typical example is བསྒྲུབས་ , which is transliterated in the official Wylie system as *bsgrubs* 'achieved'. In Tibetan Unicode, the order of these stacked characters can differ depending on the exact combinations of consonants and vowels. This varying order often yields unexpected problems when processing Tibetan Unicode text as our NLP algorithms do not recognise variants of the same order in the same word as the same type. This then increases the number of types and thus reduces the overall accuracy. For this reason, we converted the Classical Tibetan Gold Standard from Tibetan Unicode script to Wylie transcription as well. Some examples of Tibetan Unicode with Wylie transliterations are: བཅོམ་ལྡན་འདས་ *bcom-ldan-'das* 'Blessed One', ཤཱཀྱ་སེང་གེ་ *shAkya-seng-ge* 'Buddha', ཕྱག་ *phyag* 'arm, prostration'.

In a 10-fold cross-validation of the Classical Tibetan Gold Standard, this conversion to Wylie yields slightly better results. Global Accuracy was 95.0% for Tibetan Unicode vs. 96.5% for Wylie. We observed a major improvement in Unknown Words in particular from 53.4% in the Tibetan Unicode to 62.2% in the Wylie transliteration. Since the results with the Wylie transliteration are slightly better, especially for unknown, out-of-vocabulary items, converting all Unicode Tibetan to Wylie transliteration would appear to be a logical way forward. However, in practice, Unicode Tibetan script is far more widely used within the Tibetan community. To make the corpus more accessible, but also to get support from members of this community who are willing to correct segmentation and any further type of linguistic annotation, a Unicode Tibetan version is indispensable. It is therefore important to develop segmenters, taggers and parsers that work well for both, or develop tools that can automatically convert the Tibetan text (but not any type of annotation also in roman script) back from its Wylie transliteration to Unicode Tibetan script.

4.3 Memory-Based vs Neural-Network Tagging

Finally, we tested a BiLSTM-CNN-CRF tagger[5] to see if it would yield better results than the memory-based tagger. We chose this neural-network tagger, because it processes both word- and character-level representations automatically, using a combination of a bidirectional Long-Short-Term-Memory (LSTM), a Convolutional Neural-Network (CNN) and a Conditional Random Field (CRF). Although this tagger requires no pre-processing of the data or any further feature engineering, the results are better when the system can use word vectors for the specific language. Since the current number of manually corrected tokens in Old Tibetan is too small to train any neural-network-based tool, we again resorted to using Classical Tibetan instead. For Classical Tibetan, we used FastText[6] to create word embeddings with the aim of improving the results of the tagger with word vectors based on a large amount of Tibetan data digitised by the BDRC[7] and annotated by Meelen and Hill 2017: the Annotated Corpus of Classical Tibetan (ACTib) (version 1, (Meelen et al., 2017)). We then divided the above-mentioned $>318k$ token Classical Tibetan Gold Standard in training, test and developments sets (80/10/10), trained a tagger with these word embeddings and evaluated the results on the held-out test set. With its default settings,[8] this BiLSTM-CNN-CRF tagger yielded a result of 95.8% Global Accuracy (F1 score).[9]

These results are slightly better than those of the memory-based tagger (95% Global Accuracy). They are reasonable, but could be improved in a number of ways. Furthermore, at present they cannot easily be reproduced for our small corpus of the Old Tibetan *Annals* written in a very different style and genre.

[5] See `https://github.com/achernodub/targer` and Chernodub et al. (2019).

[6] `https://fasttext.cc/`

[7] `https://www.tbrc.org/`

[8] Batch size = 10; 100 epochs; dropout ration = 0.5 with the Bi-RNN-CNN-CRF model.

[9] Although it is not common practice (anymore) for POS tagging evaluations, we calculated the F1 instead of normal accuracy to make it directly comparable to the results presented by (Meelen and Hill, 2017). Actual accuracies are slightly higher than the Global Accuracies presented here.

These initial neural-network results thus look promising, but need further extension and refinement. In forthcoming work we address these issues by optimising the parameters, improving the segmentation and, with that, creating better word embeddings (Hill et al., ming).

4.4 Summary of POS Tagging

The below table summarises the results of our tests and evaluations discussed in the previous sections. There are some differences between the small and the larger tag sets and between the Unicode Tibetan script and the Wylie transliteration, with the smaller tag sets and the Wylie transliteration getting better results. The neural network tagger performs best overall with the larger tag set. With the smaller tag set, the Wylie transliteration is best for the smaller tag set.

	Global Accuracy
Clas. Tib. (318*k*; 15 tags)	96.3%
Old Tib. (3.5*k*; 15 tags)	92.8%
Old & Clas. (321.5*k*; 15 tags)	96.1%
Wylie translit. (318*k*; 15 tags)	96.5%
Unicode Tib. (318*k*; 79 tags)	95.0%
Wylie translit. (318*k*; 79 tags)	94.7%
NN-tagger (318*k*; 79 tags)	95.8%

5 Chunk-Parsing

To facilitate further future research, we also developed a 'hierarchical chunk-parse' of our Old Tibetan corpus. This is a detailed, but rather shallow parse that aims to be as theory-neutral as possible. Constituents are combined into phrases where necessary and uncontroversial, in a hierarchical fashion, e.g. nouns can combine with adjectives and determiners into a Determiner Phrase (DP), which can then combine with a post-positional case marker into a Pre/Postpositional Phrase (PP).

With the small tag set, all case markers are automatically converted into adpositions. This includes the 'Agentive Case' (`case.agn`) that is used to indicate the subject of transitive verbs. If instead we keep this agentive case marker, our small tag set will be extended, but since this marker is highly consistent in spelling, its Precision, Recall and f-score are extremely high (98%, 100% and 99% respectively for n=5627 in the Wylie transliteration evaluation of Classical Tibetan discussed above). The advantage of keeping the agentive case marker tag is that for many transitive sentences at least, we will be able to automatically detect the subject of the clause. Since Old Tibetan was a pro-drop language (i.e. pronouns need not necessarily be overtly expressed, see Tournadre 2010, 101), it is not always possible to detect non-marked subjects of verbs automatically, so a certain amount of manual correction is still always necessary. Similarly, keeping the genitive case markers (གྱི་ ཀྱི་ འི་/`case.gen`, see Tournadre and Dorje 2003, 102) has the advantage of getting much better automatically chunk-parsed results for complex nominals.

We used the NLTK chunk-parser[10] to combine tagged tokens into phrases. Semi-hierarchical structures were created by carefully formulating all phrase formation rules in the correct order, e.g. adjectival phrases (`ADJP`) before noun phrases (`NP`) and determiner phrases (`DP`) before pre/postpositional phrases (`PP`). A set of sample rules developed to generate a RegEx grammar for Old Tibetan looks like this:

```
ADJP: {<ADJ><ADJ>?}
NP: {<NOUN|PROPN>}
NUMP: {<NUM><NUM>?}
DP: {<DET>?<NP>?<ADJP|NUMP>?<DET>}
DP: {<NP><ADJP|NUMP><ADJP|NUMP>?}
DP: {<NP|DP><case.gen><NP|DP>}
SbjNP: {<NP|DP><case.agn>}
PP: {<DP|NP><ADP>}
VP: {<VERB|AUX>?<VERB|AUX>}
ADVP: {<ADV><ADV>?}
```

Some sample results are shown in (3) and (4):

(3) (S(SbjNP(NP ད་རྒྱལ་མང་པོ་རྗེ/PROPN) ས་/case.agn)
(PP (NP ཞིང་/NOUN) གྱི་/ADP)
(NP ཕྱིང་རིལ་/NOUN) (VP བགྱིས/VERB))
da rgyal mang po rje-s zhing gyi phying ril bgyis
'Dargyal Mangporje carried out a 'felt roll tax'.'

(4) (S(PP(DP(NP ཞང་ཞུང་ཡུལ་/PROPN) གྱི་/case.gen)
(NP མངན་/NOUN)) དུ་/ADP)
(NP སྤུག་གྱིམ་རྩན་རྨ་ཆུང་/PROPN) (VP བཅུག/V)

zhang zhung yul gyi mngan du spug gyim rtsan rma chung bcug

[10] http://www.nltk.org

'[He] installed Spug Gyimrtsan Rmachung as the fiscal governor of the land of Zhang-zhung.'

By exploiting the language's standard head-final word order, we can create subordinate clauses for phrases with nominalised verbs ending in subordinate conjunctions. Similarly, we can create relative clauses for nominalised verbs followed by the genitive, which functions as a relative marker linking the following word to the preceding relative clause.[11] The results require only minimal manual correction and are sufficiently theory-neutral to facilitate morpho-syntactic research within a variety of frameworks. The bracket notation is formatted according to the standard `.psd` guidelines and converted to `.psdx` (a TEI XML version of `.psd`) so that they can be queried by CorpusSearch,[12] CorpusStudio[13] or any other plain text or XML-based way of querying syntactic data. These semi-hierarchical structures are not only useful for historical syntacticians interested in comparing basic phrasal structure in different languages, but they are also invaluable for students and scholars of Tibetan to get a good insight into how the grammar of the language has changed over time. Finally, this semi-hierarchical phrasal structure serves as a great starting point for further Old Tibetan NLP challenges, such as creating more meaningful word embeddings and developing tools for keyphrase extraction, document clustering and topic modelling.[14]

6 Post-Processing

Throughout this paper we have shown how automatic NLP tools for Classical Tibetan can be optimised and extended in order to get as much use out of them for Old Tibetan. In this final section we present the results of a thorough error analysis. Suggestions for semi-automatic and rule-based corrections center around Old Tibetan, though some could be extended to the Classical Tibetan data as well.

6.1 Correction & Error Analysis

For the segmentation stage clear errors are instances of case markers and converbs that are still attached to the tokens they modify, but these markers should each receive their own tag. Because of their consistent orthography, they can often easily be split from their preceding token to facilitate POS tagging and parsing. In addition, these homophonous forms could be checked after POS tagging: their tag should be a converb following a verb, but a case marker following a noun. Similarly, a simple dictionary look-up script could 'check' whether the forms proposed by the segmenter actually exist. In order to make this latter loop-up task work well, however, we first need to collate and convert Old and/or Classical dictionaries into a reliable and searchable format.

6.1.1 Specific Old Tibetan Errors

We have detected a number of specific Old Tibetan errors as well. In example (5), for instance, we can identify some regular mistakes. Adverbial expressions like དགུན *dgun* 'in winter', དབྱརད *dbyard* 'in summer', have been tagged as nouns in many instances, so we can search for these and other recurring adverbial expressions and replace their incorrect nominal tags.

(5) བཙནཔོ་ དབྱརད་ སྤེལ་ ན་ བཞུགས ཤིང་
`NOUN ADV PROPN ADP VERB SCONJ`
"In summer, the emperor stayed in Spel."

Furthermore, converbs (functioning like subordinate conjunctions, `SCONJ`) like the ཤིང *shing* 'and, while' have often been tagged as particles instead of subordinate conjunctions, which again, can be automatically replaced.

The large amount of proper nouns in historical texts such as the Old Tibetan *Annals*, however, create a real challenge for our tools. For now, most of the time these tags (and segmentation) had to be corrected manually. For example, the following sentence was originally segmented and tagged as follows:

(6) བྲག་མ ར་ ན་ བཞུགས །
`NOUN ADP ADP VERB PUNCT`
Lit: 'cliff into/for in stayed'

The correct analysis here instead should combine the ར *-r*, which was originally tagged as an

[11] See Meelen and Roux (fc) for further examples of semi-automatic syntactic annotation.

[12] `corpussearch.sourceforge.net/index.html`

[13] `https://dev.clarin.nl/node/4239`

[14] After finishing all manual corrections at the end of the year, the entire annotated corpus and tools will be made available through Github.

adposition (ADP) with the preceding noun བྲག་མ *brag ma* ‘cliff’, resulting in the proper noun of the place called ‘Dagmar’:[15]

(7) བྲག་མར་ ན་ བཞུགས །
PROPN ADP VERB PUNCT
“[he] resided in Dagmar”

This correction, as many others occurring with proper nouns, cannot be done automatically since the error patterns are not regular. Sometimes *Dagmar*, a toponym, is tagged correctly as a proper noun, however, *dagma* + *r* ‘into a cliff’ is also a possible segmentation, in which case the correct POS tags would be NOUN + ADP. Since the Tibetan script does not identify capital letters, it is difficult for any NLP tool to make the right decision in these cases. It would also be difficult to look up ambiguous forms like these in a comprehensive, searchable Old Tibetan proper noun lexicon (which we are currently developing), as the alternative reading is still possible. This issue is exacerbated by the fact that Tibetan proper nouns are almost exclusively also normal nouns, mainly referring to natural phenomena, e.g. *Nyima* ‘sun, Nyima’.

6.2 De-Normalisation

Since in the pre-processing stage we converted and normalised our Old Tibetan to ‘Classical Tibetan’ orthography, in the post-processing stage we need to reverse the Cg3 normalization rules and apply them to the normalised text. This task is straightforward since the Cg3 normalisation grammar has been created with this de-normalisation process in mind. Through selecting and deselecting the OT and σ tags respectively, we converted our Old Tibetan corpus back to its original form after annotation.

7 Conclusion

Developing this Old Tibetan Treebank is a challenging case study of applying NLP tools to extremely low-resourced languages. We overcame many obstacles by first converting/normalising the Old Tibetan to a form of Tibetan that is orthographically much more similar to Classical Tibetan, so that the few extant tools for Classical Tibetan could be tested. We then optimised and extended these tools in various ways and finally developed a chunk-parser to create the first Old Tibetan Treebank as an indispensable tool for philologists, linguist, but also for scholars in Tibetan studies and the Tibetan communities, as it facilitates the development of good Tibetan dictionaries and other Tibetan NLP tools.

References

Chernodub, A., Oliynyk, O., Heidenreich, P., Bondarenko, A., Hagen, M., Biemann, C., and Panchenko, A. (2019). Targer: Neural argument mining at your fingertips. In *Proceedings of the 57th Annual Meeting of the Association of Computational Linguistics (ACL’2019)*, Florence, Italy.

Daelemans, W., Zavrel, J., van den Bosch, A., and Van der Sloot, K. (2003). Mbt: Memory-based tagger. *Reference Guide: ILK Technical Report-ILK*, pages 03–13.

De Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592.

Eckhoff, H., Bech, K., Bouma, G., Eide, K., Haug, D., Haugen, O. E., and Jøhndal, M. (2018). The PROIEL treebank family: a standard for early attestations of Indo-European languages. *Language Resources and Evaluation*, 52(1):29–65.

El-Haj, M., Kruschwitz, U., and Fox, C. (2015). Creating language resources for under-resourced languages: methodologies, and experiments with Arabic. *Language Resources and Evaluation*, 49(3):549–580.

Galves, C. (2018). The Tycho Brahe Corpus of Historical Portuguese. *Linguistic Variation*, 18(1):49–73.

Garrett, E., Hill, N. W., and Zadoks, A. (2014). A rule-based part-of-speech tagger for Classical Tibetan. *Himalayan Linguistics*, 13(2):9–57.

Green, N., Larasati, S. D., and Žabokrtskỳ, Z. (2012). Indonesian dependency treebank: Annotation and parsing. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 137–145.

Hill, N., Meelen, M., and Roux, E. (forthcoming). Improving the Annotated Corpus of Classical Tibetan (ACTib). *TALLIP, special issue on Asian and Low-Resource Language Information Processing.*

[15] The initial consonant cluster *br-* is pronounced as a retroflex /*ḍ*/ in Tibetan, hence the initial *D-* in the place name.

Hill, N. W. (2007). Aspirated and unaspirated voiceless consonants in Old Tibetan. *Languages and Linguistics*, 8(2):471–493.

Hill, N. W. (2010). An overview of Old Tibetan synchronic phonology. *Transactions of the philological society*, 108(2):110–125.

Hill, N. W. (2012). A note on the history and future of the 'Wylie' system. *Revue d'Etudes Tibétaines*, 23:103–105.

Meelen, M. and Hill, N. (2017). Segmenting and POS tagging Classical Tibetan using a memory-based tagger. *Himalayan Linguistics*, 16(2):64–89.

Meelen, M., Hill, N. W., and Handy, C. (2017). The Annotated Corpus of Classical Tibetan (ACTib), Part I - Segmented version, based on the BDRC digitised text collection, tagged with the Memory- Based Tagger from TiMBL.

Meelen, M. and Roux, E. (fc). Meta-dating the ACTib.

Rögnvaldsson, E., Ingason, A. K., Sigurðsson, E. F., and Wallenberg, J. (2012). The Icelandic Parsed Historical Corpus (IcePaHC). In *LREC*, pages 1977–1984.

Taylor, A. and Kroch, A. S. (1994). The Penn-Helsinki Parsed Corpus of Middle English. *University of Pennsylvania.*

Tournadre, N. (2010). The Classical Tibetan cases and their transcategoriality: From sacred grammar to modern linguistics. *Himalayan Linguistics*, 9(2):87–125.

Tournadre, N. and Dorje, S. (2003). *Manual of standard Tibetan: Language and civilization: Introduction to standard Tibetan (spoken and written) followed by an appendix on classical literary Tibetan.* Snow Lion Publications.

Summarizing Legal Rulings: Comparative Experiments

Diego de Vargas Feijo
Institute of Informatics
Federal University of Rio Grande do Sul
http://www.inf.ufrgs.br/
dvfeijo@inf.ufrgs.br

Viviane Pereira Moreira
Institute of Informatics
Federal University of Rio Grande do Sul
http://www.inf.ufrgs.br/
viviane@inf.ufrgs.br

Abstract

In the context of text summarization, texts in the legal domain have peculiarities related to their length and to their specialized vocabulary. Recent neural network-based approaches can achieve high-quality scores for text summarization. However, these approaches have been used mostly for generating very short abstracts for news articles. Thus, their applicability to the legal domain remains an open issue. In this work, we experimented with ten extractive and four abstractive models in a real dataset of legal rulings. These models were compared with an extractive baseline based on heuristics to select the most relevant parts of the text. Our results show that abstractive approaches significantly outperform extractive methods in terms of ROUGE scores.

1 Introduction

Text summarization is the task of producing a condensed representation of an input text, keeping the most relevant information. A summary should be concise, fluent, and contains paraphrased versions of the input text with a reduced length.

There are two approaches to text summarization. The first, known as *extractive*, works by selecting entire sentences directly from the source text. This has been the most widely used solution for a number of years (Luhn, 1958; Edmundson, 1969; Erkan and Radev, 2004a; Mihalcea and Tarau, 2004). Extractive methods typically work by simply (i) scoring phrases or sentences to determine the most relevant; and (ii) selecting the top scoring sentences to compose the summary. Often, the sentences are arranged in the same order of occurrence in the original text in an attempt to preserve the ideas and meanings of the sentences. The scoring function should capture how well the selected sentences represent the text and cover the topics present in the text. The lack of connectives may cause the impression that the generated summary does not have a logical flow.

The second approach, known as *abstractive*, aims to extract the main concepts or ideas from the text and generate a new condensed version different from the original. In this case, the model must learn how to write sentences in a logical flow. Using this approach, it is possible to paraphrase the original and use words that did not occur in the source. This is much more similar to the way a human would create a summary. Most recent research has focused on this approach.

With the increasing availability of data, the need for summarization is felt in many areas. This is especially true in the legal area as the texts are usually lengthy. Law operators are expected to keep updated with important information ranging from news, jurisprudence changes, and rulings from many courts.

This important information amounts to huge volumes of data which cannot be processed by humans. Each ruling from the Brazilian Supreme Court typically has more than 2,000 tokens on average. Hence, it is necessary to focus on the essential portions of each topic and extract just the information that is necessary, leaving the details aside. With that in mind, Courts usually provide extracts, with about 200 tokens on average, of their most important decisions summarizing the main topics discussed and the final outcomes. This provides an easier way to find relevant information without needing to read the whole texts.

Currently, these legal summaries are generated by humans, in a process that is time-consuming and expensive. Human summarizers need to have a good knowledge of the subject to extract the

Proceedings of Recent Advances in Natural Language Processing, pages 313–322,
Varna, Bulgaria, Sep 2–4 2019.

main topics that should appear in the summary. Another important issue with manually created summaries is the lack of standardization. Each specialist from each Court has their own writing style. A standardized way of writing is desirable as it would provide more homogeneous summaries. For these reasons, the use of abstractive approaches is especially appealing in this area.

The summarization of legal texts differs remarkably from mainstream work in text summarization, which is mostly devoted to summarizing news articles, headlines or tweets. Legal texts are generally lengthier and typically contain complex vocabulary and expressions. Also, the order of the words in some expressions can make a big difference in their meaning, *e.g.,"denied an appeal that had accepted"* is very different from *"accepted an appeal that had denied"*.

In this paper, we investigate the suitability of extractive and abstractive approaches in summarizing legal rulings. Given that the vast majority of works in this area focused solely on news datasets, we believe that testing summarization on a new domain is important given the different nature of the input documents. Fourteen approaches were tested over a real dataset containing 10K rulings from the Brazilian Supreme Court (Feijó and Moreira, 2018). Thus, another contribution is using a language that is not typically included in summarization experiments.

The remainder of this paper is organized as follows. Section 2 discusses text representations and introduces the problem of out of vocabulary tokens. Section 3 revises recent works on abstractive summarization. Section 4 describes the dataset and how it was prepared to be used in the experiments. Section 5 presents a simple heuristic-based extractive summarizer that we proposed to serve as a baseline. Sections 6 and 7 describes the algorithms and models that were investigated here. Section 8 discusses our results and findings. Finally, Section 9 concludes this paper and points out possibilities for future work.

2 Text Representation

Representing texts using a sequence of words is useful for exchanging information between humans. However, when using neural models, we need to convert the text into numbers that could be used as input into the neural network.

The usual way of generating a text representation is to split the text and generate a vocabulary from the training data keeping the most frequent tokens. Even considering that the vocabulary would be extracted from a single language, if we consider that texts have upper and lower case characters, numbers, dates, *etc.* the vocabulary usually becomes quite large, frequently with hundreds of thousands of different tokens.

A large vocabulary is a problem because the output layer of the neural network must have its size. This means that the probability of choosing the correct output diminishes as the vocabulary increases. Also, even if infrequent tokens are represented in the vocabulary, its infrequent use would not be sufficient for the model to learn when to use it correctly.

One approach to deal with the size of the vocabulary is to convert all characters to lowercase and replace numbers and dates with zero representations. With these modifications, the vocabulary becomes smaller, but the model will become less capable of generating outputs in the same way as humans would.

Even with the text simplifications, the problem with out of vocabulary words (OOV) remains because not all possible words will be represented. So, it is usual to represent any token that is not present in the vocabulary by a reserved OOV token. This token would be used for uncommon names, dates, and numbers.

Another approach is to use one token per character. This greatly reduces the vocabulary size only using lower and upper case characters, numbers, and symbols. Although the vocabulary is smaller, probably not more than few hundred tokens, it would require each word to be represented by several tokens, leading to very long document. This is problematic because the model will require a large memory to be able to generate the summary without repeating already generated tokens.

The alternative to mitigate the disadvantages of these two approaches is to use *sub-word units as the token representation* (Schuster and Nakajima, 2012; Chitnis and DeNero, 2015; Sennrich et al., 2015; Kudo, 2018). The idea is that a token would represent a common pattern seen in the training data rather than words or characters. This operates with a fixed vocabulary size and assigns a token for the most common patterns found. With this method, the OOV problem is reduced as one word now can be represented by a combination of sub-words. The problem of longer sequences is also addressed because a token now can represent several characters.

3 Related Work

Neural Networks are models capable of learning very complex functions. In the last few years, there has been significant interest in applying them for natural language tasks such as automatic translation and summarization.

One of the first issues that needs to be addressed is that Neural Networks require that both inputs and outputs have a fixed a length, and that is not the case when dealing with text, because each document (or sentence) can have a different length. In order to overcome this limitation, Sutskever et al. (2014) introduced a general end-to-end approach capable of representing sequence-to-sequence models using LSTM (Long Short-Term Memory) cells. Nevertheless, both input and output must still have fixed lengths large enough to fit. The network is trained to output the end-of-sentence (EOS) token when the output is large enough. With this approach, both source input and generated sequence may logically have different lengths and do not require any type of alignment between input and output.

Bahdanau et al. (2014) proposed a model for learning to align the input with the generated output. They realized that the approach for encoding the whole text before starting to decode requires that the whole idea is represented by just one vector. So, they proposed to use auxiliary vectors to represent the alignment of the input in relation to the generated output. Their approach significantly improved the quality of the generated outputs and became known as Attention vectors.

Following the same ideas, Luong et al. (2015) explored different versions for the attention mechanism. They also noticed that as the length of the input increases, the attention vector has a lot more difficulty in learning the weights. So they evaluated the impact of using a local attention mechanism to look just for a smaller portion of the source at each time step.

The attention mechanism works well for short texts, but struggles to focus on relevant information when applied to long documents, common in the legal domain.

Neural Machine Translation (NMT) is a sequence-to-sequence task that is similar to summarization in the sense that both require some text comprehension before an output can be generated. When translating, a model does not have an exact alignment for each word read in the source to the word generated in the output. This happens because a source word may be represented by more (or fewer) words in the translation. Also, the order of the tokens can be different.

Wu et al. (2016) describe the architecture used for the Google Neural Machine Translation (GNMT) system. In that work, they used a LSTM network with 8 encoders and 8 decoders using attention and residual connections. In the beam search, they used length-normalization and applied a coverage penalty to favor an output that is able to cover more words from the source sequence.

Vaswani et al. (2017) improved the GNMT system replacing entirely the recurrence and convolutions by an attention-based model known as Transformer. The model is quite complex and relies on many training variables, but it has the advantage of allowing more parallel computation. With this modification, the model is able to use many GPUs and train a lot faster.

See et al. (2017) addressed two common problems found in the application of RNNs in the context of summarization. First, the problem of rare words that were not present in the vocabulary was solved by having hybrid pointer networks that are capable of using the source word as output when the attention weight is high enough. The second problem is using a coverage vector that represents the weighted sum of the attention vectors. Then, to increase coverage, their model is trained to penalize every time the attention vectors are high in the same regions, encouraging the model to better distribute the attention over the source input.

In our work, the problem of rare words has been diminished using sub-word encoding, as discussed in Section 2. The coverage mechanism, proposed to deal with repetition problem, is complex to train. It is used after the training to condition the decoder to avoid generating attention vector using the same positions that were already used. Paulus et al. (2017) proposed trigram avoidance during the beam search. This approach is much simpler and easy to apply but does not really fix the problem, as it still will happen, it only masks its effects for the evaluation.

There is a growing interest in using reinforcement learning approaches (Paulus et al., 2017; Li et al., 2018; Celikyilmaz et al., 2018) to improve summarization performance. In general, reinforcement learning is employed when an non-differentiable operation is being used. These methods have the disadvantage of being hard to tune and generally slow to converge.

Chen and Bansal (2018) proposed a method for using both extracting and abstracting approaches. Their idea was to select salient sentences and rewrite them abstractively. Reinforcement learning was used to combine these two neural networks. Their idea seems to be a good approach for working with long documents. Although, they did not report results for long documents datasets. We intend to further explore the application of this technique to long documents in the legal domain.

4 Materials and Methods

Our summarization experiment in the legal area uses rulings written in Portuguese. There are some peculiarities when using a language different from English, *e.g.,* we need to check if the standard summarization evaluation (designed for English) can be directly applied.

4.1 Dataset

The dataset used in our experiments is RulingBR (Feijó and Moreira, 2018). It contains about 10K rulings from the Brazilian Supreme Court. These rulings were taken by a group of judges – the individual decision from each judge is known as their "vote", the final decision is made by the majority of the votes. Each ruling has four sections: (*i*) the *ementa*, which represents the summary; (*ii*) the *acórdão*, which has the final decision taken by the majority; (*iii*) the *relatório*, which is an extensive description reporting the request and the actions taken so far; and (*iv*) the *voto*, which contains the individual votes from all judges.

The dataset was randomly split into training, validation, and test sets. Training takes up 60% of the instances (6,373), whereas validation and test have 20% of the instances each (2,125). On average, each document has about 2,500 tokens.

We use the *ementa* as the human-generated ground truth summary. The other three sections together compose the input text, which is submitted to the summarization systems. On average, the *ementa* has a little less than 10% of the size of the source input.

4.2 Official Rouge Script

The standard evaluation metric for text summarization is called Recall-Oriented Understudy for Gisting Evaluation (ROUGE). The general idea of this metric is to count the number overlapping units between one or more reference summaries and the machine-generated summary. It is expected, that a high-quality summary should use the same words found in the reference summaries and preferably in the same order.

The results reported here include Precision, Recall, and F-measure for ROUGE-1, ROUGE-2, and ROUGE-L metrics. A confidence of 95% was adopted. We used the official ROUGE (Lin, 2004) 1.5.5 script in our experiments. Most parameters were set to their default values. The `pyrouge` package, which is a wrapper to the official script, was used to provide the required output text (in HTML) and the configuration file. The only change was to remove the call to the English Porter Stemmer, since our texts are in Portuguese.

4.3 Text Preprocessing

The official Rouge script treats any non-ASCII characters as word separators. Thus, we transformed all accented characters to their base form, *i.e.,* diacritics were removed. In addition, we changed all text to lowercase and isolated the standard punctuation symbols from the alphabetic characters to avoid them being interpreted as part of some word.

4.4 Vocabulary

Portuguese has a rich vocabulary, with words having lots of variant forms. Verbs, specially, can have dozens of different suffixes corresponding to the diverse conjugations. In the legal domain, it is common to reference existing laws, specific dates, and names. Thus, it is very unlikely that the vocabulary generated during training will contain all possible words that are present in the test set. To deal with problem of OOV words, we used the SentencePiece package (Google/SentencePiece, 2019), which implements the sub-word units with unigram representation. The combination of pieces is used to generate words even when they are not present in the training set.

5 A Simple Extractive Summarization Baseline

When generating summaries, the source input length and the desired summary length are required. Ideally, these two lengths should be automatically determined by the algorithm, but that is not how these standard extractive approaches work. In order to establish these lengths, we defined a heuristic-based baseline extractive method.

5.1 Heuristic for Sentence Selection

Through empirical observation, we found that the *relatório* (report) section from the source text usually contains most of the information that is typically present in the reference summary. So, after removing some boilerplate text that is usually present at the beginning of the documents, we extract a sequence of words until the desired summary length is reached.

5.2 Target Length

In the RulingBR dataset, the mean length for the test set is 190 tokens, with a minimum of 20 and maximum of 1,909 tokens. This represents a wide range of summary lengths. Since every summary generated by our baseline needed to have the same length, we experimented truncating the reference summaries at different points to observe the effect over the mean length of the test set.

Table 1 shows the effect of imposing length limits to reduce the standard deviation in favor of a more predictable summary length. The dilemma here is to balance between a lower limit and lower standard deviation (but risking losing important information) with a higher limit and higher deviation (but with a large error associated).

Another concern that arises when deciding about the target length is the fact that the summaries will be evaluated using the ROUGE metric. ROUGE relies both on the Precision and on the Recall. Shorter summaries are expected to have higher precision, while longer summaries tend to have higher recall. Our F-score results assign the same weight to precision and recall. Table 2 shows that truncating the source length has little effect over the F-measure of the ROUGE scores. In order to make a fair comparison between the algorithms, we aimed at generating summaries of similar lengths.

Limit	Mean	Min	Max	Std Dev
No	190	20	1,909	179.46
600	180	20	600	131.89
450	173	20	450	112.02
300	158	20	300	82.37
150	120	20	120	36.85

Table 1: Lengths of the reference summaries that compose our Test Set.

5.3 Source Length

Abstractive summarization models require a fixed size input. Different lengths will require padding, which in turn will have a negative impact on training. We used truncated parts of sections *relatório* and *voto* as they concentrate most of the important information. Table 2 shows the results of our experiment using *relatório* and *voto* with lengths of 150, 300, 450, or 600 tokens and trying to find summaries with this same length.

Because lengthier summaries would require more memory and would lead to a broader range of lengths, we adopted the 300+300 tokens as the input length limit in our experiments (*i.e.,* the input text is a concatenation of the 300 first tokens from *relatório* and the 300 first tokens from *voto*).

As shown in Table 1, truncating the target length to 300 tokens leads to summaries of 158 tokens on average. This will be used as the desired summary length.

6 Extractive Approaches

The ten extractive approaches used in our experiments are described in this section. We used the implementations provided by Sumy (Belica, 2018) and an improved version of TextRank provided by Gensim (Řehůřek and Sojka, 2010).

6.1 Luhn

Proposed by Luhn (1958), the seminal method for determining the importance of a sentence is calculated using Term Frequency (TF) and Inverse Document Frequency (IDF). Significant words are selected among the most frequent words found in the document. Then, the highest scoring sentences are selected to be part of the summary.

When calculating word frequencies, Luhn algorithm proposes a simple stemmer by matching words using their prefixes. A match happens when the number of non-matching letters is less than six.

Length	R1-F	R1-P	R1-R	R2-F	R2-P	R2-R	RL-F	RL-P	RL-R
600+600	33.99	33.46	**43.16**	**12.20**	12.34	**15.59**	19.44	19.10	**25.36**
450+450	34.07	34.40	41.32	12.06	**12.50**	14.72	19.48	19.64	24.24
300+300	**34.47**	**34.84**	40.25	12.08	12.43	14.19	19.74	19.88	23.58
150+150	**34.47**	34.32	37.46	11.88	11.84	13.01	**20.49**	**20.29**	22.61

Table 2: ROUGE scores for different source lengths. The results show that the F-measure is reasonably stable across different lengths.

This technique may be language specific, thus it is possible that a stemmer specificaly designed for the target language would have a different behavior. Here, we use the standard implementation as provided by the Sumy without stemming or stopword removal.

6.2 LexRank

This is a stochastic graph-based method for computing the relative importance of sentences (Erkan and Radev, 2004b). It assumes that the main idea of a text is often paraphrased. As a consequence, finding similar sentences would be the same as finding the important sentences. Also, the central sentence of a cluster would indicate that this sentence is the most similar among them and would probably capture more information.

6.3 TextRank

This is a graph-based ranking model of deciding the importance of a vertex within a graph (Mihalcea and Tarau, 2004). The basic idea is to have some form of votes every time one vertex is similar to another. The highest voted vertex would be the most important. Gensim uses a modified version (Barrios et al., 2016) in which the BM25 similarity function is used in place of just the number of common tokens as adopted by the original TextRank. Thus, TextRank appears twice in our results as we used both the Gensim and the Sumy implementations.

6.4 SumBasic

This algorithm is based on the fact that words present in the summary tend to be the most frequent in the text (Nenkova and Vanderwende, 2005). It computes the probability distribution over the words appearing in the input. Then the sentences containing the highest probability words are selected and for each word in these sentences, update their probabilities until the desired length is reached.

6.5 KLSum

Kullback-Leibler (KL) is a way to compare two probability distributions. It also computes the probability distribution of words in the text. Then, the problem of finding the summary can be stated as finding a set of summary sentences which the probability distribution closely matches the document distribution (Haghighi and Vanderwende, 2009).

6.6 LSA

The summarization using Latent Semantic Analysis (LSA) (Steinberger and Jezek, 2004; Gong and Liu, 2001) is done by constructing a sparse *token x sentences* matrix, applying Singular Value Decomposition (SVD), selecting the singular vector will retrieve the scores for each token, then selecting sentences with highest normalized scores.

6.7 Random

This is another baseline summarizer in which random scores are assigned to the sentences. The highest scoring sentences are selected to the summary.

7 Abstractive Approaches

We experimented with Neural Network models using the `OpenNMT-tf` package (Klein et al., 2017). The models evaluated here were NMTSmall, NMTMedium, Transformer, and TransformerAAN.

NMTSmall and NMTMedium are standard Recurrent Neural Network models. They use an encoder-decoder architecture. The decoder employs Luong et al. (2015) style attention model over the input. The network is trained to learn when to stop generating the summary. This is done appending an End-of-Document token to the instances during training. When the network generates this token, the output is truncated at this point. The model uses a beam search of size four when decoding, and it is configured to ignore outputs that were shorter than the minimum length.

Both NMT and Transformer models use word embedder of size 512. Each model was evaluated until its training loss was no longer diminishing. We report ROUGE results with minimum decoding lengths of 100 and 120 tokens. Recall that we are using SentencePiece and each decoded word may be represented by more than one token. So, the generated output may contain fewer words than this minimum length. In all reported results, we show the mean length of the output considering generated tokens separated by spaces.

Two NMT configurations were used. NMT-Small uses 2-layers, unidirectional LSTM with 512 units, and it has converged in 15,000 steps. NMTMedium uses 4-layers, bidirectional LSTM, with 512 units and it has converged in 26,000 steps. Transformer model uses the configuration as originally proposed by Vaswani et al. (2017). TransformerAAN uses cumulative average attention network in the decoder as described in (Zhang et al., 2018). The objective is to reduce the required training and inference time.

8 Results and Discussion

The performance of the extractive algorithms shown in Table 3 was disappointing. With the exception of SumBasic, all other algorithms have performed worse than our simple Baseline by at least 0.6 points in ROUGE-L. In some cases, the performances were not far from the random baseline. A possible explanation for such poor results is the limitation of this approach of generating summaries using only complete sentences that were present in the source text. Looking at the generated summaries, most of them have selected just one very long sentence while others used a few random disconnected sentences.

Our experiments varying the lengths of the input method (Table 2) have shown that even with larger source inputs, which could contain more tokens that should be present in the output, the performance was decreasing.

The baseline results provided by Feijó and Moreira (2018) for the extractive methods ranged between 11 and 16 points in terms of ROUGE-L. Those results cannot be directly compared to the results in our experiments because they had removed stopwords and they reported results for the entire dataset. Since in this paper, we require a training phase for the neural network models, ROUGE results are reported only for the test set.

One advantage of extractive algorithms is that they do not require prior training, and they can be applied directly over the test data. On the other hand, after the time-consuming training, the abstractive approaches can create the summary a lot faster.

Table 4 shows reasonably good results for both NMT and Transformer models. There was a small advantage for the standard Transformer model when compared to its modified version with the Average Attention Network. They both have reached very similar results and have converged in about 40K steps.

Since the Transformer model has many variables, it requires a lot of memory to run. So, the batches need to be smaller. As a consequence, it needed more steps to converge. Despite that, we observed that it trains faster than standard RNNs. As we are using a concurrent environment, our measures of the time taken for training were not accurate, so we could not report them.

Summarization results in other datasets are not directly comparable to our results. Still, they may serve as reference. Zhang et al. (2019) reports that the current state-of-the-art for the CNN/Daily Mail dataset (Nallapati et al., 2016) reaches scores of ROUGE-1 41.71, ROUGE-2 19.49 and ROUGE-L 38.79.

The summaries generated by the abstractive approaches were promising. They look similar to those produced by humans. In most generated summaries, the main topic was correctly captured by the summarizer. Nevertheless, as shown in Figure 1 there are still some cases in which the summarizer barely captured any meaning of the text, generating summaries that had almost no relation with the expected output. In these cases, the extractive approach would probably have done better. In other cases, the general meaning was correctly captured, but the output had repeating expressions. We believe this may have been caused by the minimum length restriction.

Legal operators rely on summaries to their jobs, since it is impossible to read the full contents of each decision to find precedents for their cases. Missing or referring to an incorrect precedent may cause the petition to be denied and the case would be lost. Thus, considering the results seen so far, neither approach delivers results that could safely

Algorithm	R1-F	R1-P	R1-R	R2-F	R2-P	R2-R	RL-F	RL-P	RL-R
Random	31.52	34.42	34.81	10.55	11.81	11.49	17.88	19.67	19.99
Baseline	34.47	**34.84**	40.25	12.08	12.43	14.19	**19.74**	**19.88**	**23.58**
Luhn	33.16	33.17	39.08	11.06	11.25	13.09	18.77	18.67	22.65
LexRank	34.06	34.06	40.07	11.65	11.85	13.69	19.16	19.04	23.06
LSA	32.31	32.26	38.04	10.44	10.62	12.23	17.88	17.76	21.50
KLSum	31.96	32.42	37.14	11.45	11.74	13.30	18.24	18.38	21.66
SumBasic	**34.51**	34.41	**40.74**	**12.32**	**12.49**	**14.46**	18.76	18.69	22.43
TextRank[1]	33.09	33.07	38.99	10.85	11.07	12.73	18.78	18.67	22.60
TextRank[2]	33.66	34.14	39.10	12.00	12.31	13.97	19.16	19.24	22.82

Table 3: ROUGE scores using extractive algorithms

Model	Len	R1-F	R1-P	R1-R	R2-F	R2-P	R2-R	RL-F	RL-P	RL-R
NMTSmall	130	38.86	44.75	40.42	21.28	23.14	22.89	30.22	33.99	32.02
NMTMedium	130	43.25	49.25	44.80	25.41	27.60	27.05	33.91	37.78	35.69
Transformer	134	**44.27**	**49.38**	46.24	**26.50**	**28.36**	28.26	**35.27**	**38.52**	37.36
TransformerAAN	137	43.67	48.38	45.90	25.60	27.15	27.43	34.47	37.38	36.74
NMTSmall	141	38.37	42.48	41.54	20.77	21.77	23.29	29.55	31.92	32.68
NMTMedium	140	41.56	46.44	44.00	23.43	24.95	25.56	32.01	34.87	34.58
Transformer	145	43.91	47.34	**47.76**	25.95	26.93	**28.84**	34.55	36.48	**38.16**
TransformerAAN	147	43.39	46.46	47.37	25.23	25.93	28.13	33.90	35.51	37.60

Table 4: ROUGE scores using abstractive models. The mean length is shown because it affects the scores.

replace humans in this task. The current state is promising, but automatic systems are not always capable of generating good summaries. Hence, they could be used to prepare drafts which then need to be revised by humans.

Ground Truth: direito administrativo . lei n^{o} 11.064/2002 . servico auxiliar voluntario . policial militar temporario . acrescimo de 1/3 , 13^{o} salario , adicional de insalubridade e de local de exercicio . eventual violacao reflexa da constituicao da republica nao viabiliza o recurso extraordinario . recurso extraordinario interposto sob a egide do cpc/1973 . alegacao de ofensa aos arts . 2^{o}, 5^{o}, ii , e 37 , caput , ii e ix , da constituicao da republica . agravo manejado sob a vigencia do cpc/2015 . . .

Generated: direito administrativo . militar . promocao . ato de bravura . recurso extraordinario interposto sob a egide do cpc/2015 . eventual ofensa reflexa nao enseja recurso extraordinario . necessidade de interpretacao de legislacao local . aplicacao da sumula no 280/stf . agravo manejado sob a vigencia do cpc/2015 . . .

Figure 1: Summary generated by the Transformer model. The ground truth refers to a petition for compensation when made by a policeman. The generated summary was about a petition for benefits due to an act of bravery by military personnel.

9 Conclusion

This work presented a comparative investigation of ten extractive and four abstractive methods for text summarization using Portuguese language applied to the legal area. The data used here consist of a real-world legal domain dataset containing 10K rulings from the Brazilian Supreme Court. The results show that extractive methods provided weak performance being unable to generate useful summaries. On the other hand, abstractive models provided much better results, with summaries that were very similar to those produced by humans. However, they also presented severe problems with repeating expressions and the introduction of subjects that were not present in the source documents. We intend to further investigate the causes of the factual errors and address this problem in future work.

Acknowledgment: This work was partially supported by CNPq/Brazil and by CAPES Finance Code 001.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *arXiv preprint arXiv:1602.03606* .

Mišo Belica. 2018. sumy: Module for automatic summarization of text documents and HTML pages. https://github.com/miso-belica/sumy.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357* .

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *CoRR* abs/1805.11080. http://arxiv.org/abs/1805.11080.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2088–2093.

H. P. Edmundson. 1969. New methods in automatic extracting. *J. ACM* 16(2):264–285. https://doi.org/10.1145/321510.321519.

Günes Erkan and Dragomir R. Radev. 2004a. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.* 22(1):457–479. http://dl.acm.org/citation.cfm?id=1622487.1622501.

Günes Erkan and Dragomir R Radev. 2004b. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Diego de Vargas Feijó and Viviane Pereira Moreira. 2018. Rulingbr: A summarization dataset for legal texts. In Aline Villavicencio, Viviane Moreira, Alberto Abad, Helena Caseli, Pablo Gamallo, Carlos Ramisch, Hugo Gonçalo Oliveira, and Gustavo Henrique Paetzold, editors, *Computational Processing of the Portuguese Language*. Springer International Publishing, Cham, pages 255–264.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 19–25.

Google/SentencePiece. 2019. Unsupervised text tokenizer for neural network-based text generation. `https://github.com/google/sentencepiece`. Accessed: 2019-05-25.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 362–370.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*. Association for Computational Linguistics, Vancouver, Canada, pages 67–72. https://www.aclweb.org/anthology/P17-4012.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *CoRR* abs/1804.10959. http://arxiv.org/abs/1804.10959.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* .

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.* 2(2):159–165. https://doi.org/10.1147/rd.22.0159.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025. http://arxiv.org/abs/1508.04025.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR* abs/1602.06023. http://arxiv.org/abs/1602.06023.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005* 101.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pages 5149–5152.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM* 4:93–100.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR* abs/1706.03762. http://arxiv.org/abs/1706.03762.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. *CoRR* abs/1805.00631. http://arxiv.org/abs/1805.00631.

Haoyu Zhang, Yeyun Gong, Yu Yan, Nan Duan, Jianjun Xu, Ji Wang, Ming Gong, and Ming Zhou. 2019. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243* .

Entropy as a Proxy for Gap Complexity in Open Cloze Tests

Mariano Felice **Paula Buttery**
ALTA Institute
Computer Laboratory
University of Cambridge
Cambridge, UK
{mf501,pjb48}@cam.ac.uk

Abstract

This paper presents a pilot study of entropy as a measure of gap complexity in open cloze tests aimed at learners of English. Entropy is used to quantify the information content in each gap, which can be used to estimate complexity. Our study shows that average gap entropy correlates positively with proficiency levels while individual gap entropy can capture contextual complexity. To the best of our knowledge, this is the first unsupervised information-theoretical approach to evaluating the quality of cloze tests.

1 Introduction

Fill-in-the-gap or *cloze* test exercises are common means of assessing grammar and vocabulary in the realm of English as a Foreign Language (EFL). The most common example is the *multiple choice question*, which presents the student with a gapped sentence and a set of possible answers from which the right one is to be selected. These are referred to as *closed* cloze questions, since the answer is limited to the alternatives given. On the contrary, *open* cloze questions do not provide predefined options, so the student must produce an answer from scratch.

Generating these exercises is a laborious process, since they must be carefully designed to ensure they test the desired learning objective and do not confuse or present trivial questions to the student. For this reason, choosing the optimal locations in a sentence to insert the gaps and defining a suitable set of answer options becomes crucial, especially when exercises are generated automatically.

In this paper, we focus on open cloze tests and show how entropy can be used to assess the complexity of each gap in the text. Entropy is shown to provide insights into the expected difficulty of the question and correlate directly with the target proficiency level of the exercises. Exploiting this information should thus facilitate the automatic generation of more reliable open cloze exercises.

2 Related Work

Work on automated cloze test generation has mostly focused on multiple choice questions and distractor selection (Mitkov and Ha, 2003; Sumita et al., 2005; Brown et al., 2005; Lee and Seneff, 2007; Lin et al., 2007; Smith et al., 2010; Sakaguchi et al., 2013). Conversely, there has been little work on open cloze tests. Pino et al. (2008) describes a strategy to generate open cloze questions using example sentences from a learners' dictionary. Sentences are chosen based on four linguistic criteria: (grammatical) complexity, well-defined context (collocations), grammaticality and length. Further work improved on this method by providing hints for the gapped words (Pino and Eskenazi, 2009).

Malafeev (2014) developed an open source system to emulate open cloze tests in Cambridge English exams based on the most frequent gapped words. Expert EFL instructors found the generated gaps to be useful in most cases and had difficulty differentiating automated exercises from authentic exams. More recently, Marrese-Taylor et al. (2018) trained sequence labelling and classification models to decide where to insert gaps in open cloze exercises. The models achieved around 90% accuracy/F_1 when evaluated on manually created exercises.

While the quality of the generated gaps has traditionally been judged by human experts (Pino et al., 2008; Malafeev, 2014) or estimated from student responses (Sumita et al., 2005; Brown et al., 2005; Skory and Eskenazi, 2010; Beinborn et al., 2014; Susanti et al., 2016), systems should ideally predict the quality of the gaps during the generation process. In this regard, Skory and Eskenazi (2010) observe that Shannon's information theory (Shannon, 1948) could be used to estimate the reading difficulty of answers to a gap based on their probability of occurrence. Thus, for the sentence *"She drives a nice ______"*, the word "car" would be the most likely answer (lowest readability level) while words such as "taxi", "tank" and "ambulance" would be at increasingly higher levels.

Research on predicting the difficulty of cloze tests is also directly relevant to this work. Beinborn et al. (2014) built models to predict the difficulty of C-tests (i.e. gaps with half of the required word removed) at the gap and test level and later extended their approach to cover closed cloze tests (Beinborn et al., 2015; Beinborn, 2016). More recently, Pandarova et al. (2019) presented a difficulty prediction model for cued gap-fill exercises aimed at practising English verb tenses while Lee et al. (2019) investigated how difficulty predictions could be manipulated to adapt tests to a target proficiency level. Unlike our work, however, all these approaches are supervised and not applied to open cloze tests.

3 Entropy

In this paper, we build on the assumption that the complexity of a gap is correlated to the number of possible answers determined by the surrounding context and the likelihood of each answer. As noted by Pino et al. (2008), high-quality open cloze questions should sufficiently narrow the context of each gap in order to avoid multiple valid answers, which would make the exercise too broad in scope and therefore ineffective. We thus assume that gaps with more restricted context eliciting very specific answers should be more useful than broad gaps with very general answers, so the less "branching" that a gap allows, the better.

This property can be modelled by *entropy*, which quantifies the amount of information conveyed by an event. Intuitively, entropy can be considered a measure of disorder, uncertainty or surprise. If the probability of an event is very high, entropy will be low (i.e. there is less surprise about what will happen) while events with low probabilities will lead to higher entropy. Shannon's entropy, a common formulation to measure the number of bytes needed to encode information, is shown in Equation 1, where $P(x_i)$ stands for the probability of event x_i, i.e. the probability that each word in the vocabulary occurs in the evaluated context.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \qquad (1)$$

In this work, we use entropy to assign a score to each gap based on the number of valid words that could fill in the slot given the surrounding context. As a result, gaps with many possible answers will yield higher entropy than those with fewer answers.

4 Experiments

We followed Malafeev's (2014) approach and used open cloze tests from Cambridge English examinations as our gold standard data, since they are manually created by experts in the field of EFL testing. We collected the sample open cloze tests for KET, FCE, CAE and CPE exams that are featured in their respective online handbooks[1] (one per exam together with their answers). These exams correspond respectively to levels A2, B2, C1 and C2 in the Common European Framework of Reference for Languages (CEFR). An open cloze test is not included in the PET (B1) exam, which is why it has not been included in our experiments.

For each exam, we restored the original text by using the answers provided (using the first alternative if there were many) and created 10 different variations of the open cloze tests by inserting gaps randomly throughout the text. We created the same number of gaps as in the original tests.

For each original and automatically generated test, we compute entropy per gap using a 5-gram language model trained on the 1 Billion Word WMT 2011 News Crawl corpus[2] using KenLM (Heafield, 2011). We use the language model bidirectionally, taking 3 words to the left and right of each gap to predict the probability of the next and previous words respectively. Since we obtain a probability for all the words in our vocabulary ($>$ 82,200 words) given the left and right context individually, we multiply the probabilities for each word to get a unified "bidirectional" probability (see Figure 1). Given that this can lead to infinitesimal probabilities that can affect computa-

[1] https://www.cambridgeenglish.org/exams-and-tests/

[2] https://www.statmt.org/lm-benchmark/

Electronics firms, for example, expect to have only six months after they have introduced a new product before a rival company produces a _____ efficient or cheaper alternative.

→		←
wide	more	more
variety	very	energy
lot	less	is
quarter	is	as
line	and	less
...	...	...

Figure 1: An example calculation of candidate answers for a gap using the left and right context (in red). Candidate words are ranked from the most to the least probable.

tion, we use only the top 100 most probable words when computing entropy for each gap.

4.1 Results

Table 1 shows information about our gold standard tests, including CEFR levels, number of gaps and average gap entropy. The average gap entropy correlates positively with CEFR levels, suggesting that entropy increases with proficiency levels.

We then computed the average gap entropy for each of the 10 automatically generated tests per exam and compared them to the gold standard. Results are shown in Table 2.

Unlike the handcrafted gold standard, the automatically generated tests were produced randomly by a machine with no knowledge of test design so we would expect automatic gaps to be often inserted in inconvenient locations within the text, yielding lower quality tests. This hypothesis is verified by looking at the average gap entropy for the automatic tests, which is much higher than for the gold standard in the majority of cases (77.5%). This supports our intuition that entropy can be used to discriminate between good and bad gaps and, consequently, between good and bad tests.

We noticed that automatically generated tests for CPE tend to have lower entropy than the gold standard, contradicting our assumption in principle. However, we do not believe that these lower values indicate better tests but rather that they deviate from the expected difficulty for this proficiency level. In fact, we would expect high-quality tests to have average gap entropy around that of the gold standard tests, not too far below or over this reference value. Based on this premise, better automated tests can be constructed by controlling the entropy of gaps in the text, in line with previous work by Lee et al. (2019).

Exam	**CEFR level**	**Number of gaps**	**Avg. gap entropy**
KET	A2	11	1.29 ± 0.69
FCE	B2	9	2.33 ± 1.28
CAE	C1	9	2.69 ± 1.22
CPE	C2	9	5.16 ± 3.38

Table 1: Characterisation of our gold standard data.

4.2 Analysis

We looked at the gaps with the lowest and highest entropy to analyse how these values relate to the surrounding contexts. Table 3 shows the gaps in our gold standard tests with the lowest and highest entropy.

First, we found that gaps with the lowest entropy correspond mostly to exams at low CEFR levels while those with the highest entropy correspond to the highest CEFR level. This confirms our initial finding that entropy correlates directly with proficiency levels.

Second, we observed that gaps with low entropy are very restricted in context and built around very simple grammatical structures or vocabulary, making it easy to figure out the answers. On the other hand, gaps with high entropy are part of more complex grammatical structures and require longer context or understanding in order to be solved. This explains why our language model is unable to estimate the right answers for complex gaps, leading to higher entropy.

Finally, we investigated the correlation between entropy and the number of valid answers per gap. Pearson correlation for gaps in our gold standard tests is reported in Table 4. Contrary to our intuition, there is no consistent relationship between entropy and the number of valid answers per gap in our gold standard: KET shows negative correlation while CPE shows moderate positive correlation. We hypothesise that this is due to a limitation of the language model used in this preliminary study, which is unable to estimate the right word probabilities for gaps in complex contexts for the reasons described above. Using a more sophisticated language model should ameliorate this problem.

In any case, the values of entropy computed with our current model seem to capture the complexity of the gaps in context, which serves as a measure of difficulty. This, combined with the positive correlation with CEFR levels, makes en-

Exam	Average gap entropy per test									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
KET	5.40	3.47	3.63	3.73	4.22	4.18	4.60	4.33	4.74	4.34
FCE	4.53	7.13	6.12	4.30	**2.23**	4.01	2.45	3.93	4.18	5.67
CAE	4.70	4.66	3.57	**2.68**	3.26	4.38	2.79	5.08	5.07	4.82
CPE	6.58	**3.91**	**2.72**	**4.43**	**5.02**	**4.18**	5.83	5.46	**4.02**	**3.26**

Table 2: Average gap entropy for the automatically generated tests. Values lower than the gold standard are marked in bold.

Exam	Gap in context	Entropy	Answers
FCE	*..., apart _____ some minor mechanical problem...*	0.01	*from*
KET	*But _____ is some good news!*	0.23	*there / here*
KET	*_____ this okay?*	0.43	*is*
CPE	*... modern robots are dumb automatons, _____ of striking up relationships with their human operators.*	8.40	*incapable*
CPE	*Phones and computers have already shown the _____ to which people can develop relationships with...*	8.65	*extent / degree*
CPE	*Although sophisticated _____ to assemble cars...*	9.66	*enough*

Table 3: Example gaps with the lowest and highest entropy.

Exam	Pearson's ρ
KET	-0.1518
FCE	0.2333
CAE	0.0908
CPE	0.5149

Table 4: Correlation between entropy and the number of valid answers per gap.

tropy a suitable unsupervised evaluation measure for gaps in open cloze tests and encourages future work beyond this pilot study.

5 Conclusion and Future Work

This work investigated the use of entropy as an evaluation measure for gaps in open cloze EFL tests. Our study revealed that the average gap entropy of a test correlates positively with proficiency levels, so easier tests will contain gaps with lower entropy. A comparison between randomly generated tests and the handcrafted gold standard tests showed that the former had much higher entropy in general, confirming our intuition that generating random gaps is not optimal and that entropy can be used to discriminate between good and bad tests.

We also investigated the correlation between entropy and the number of valid answers per gap but results showed no consistent relationship, most likely due to the limitations of the n-gram language model used in this preliminary work. However, entropy was found to be a suitable proxy for gap complexity, which can be used to control the automatic generation of open cloze tests. Future work will address the limitations in this pilot study and investigate entropy on a larger sample.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the difficulty of language proficiency tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2015. Candidate evaluation strategies for improved difficulty prediction of language tests. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–11, Denver, Colorado. Association for Computational Linguistics.

Lisa Marina Beinborn. 2016. *Predicting and Manipulating the Difficulty of Text-Completion Exercises for Language Learning*. Ph.D. thesis, Fachbereich Informatik, Technische Universitt Darmstadt, Darmstadt, Germany. PhD thesis.

Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 819–826, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.

Ji-Ung Lee, Erik Schwan, and Christian M. Meyer. 2019. Manipulating the difficulty of c-tests. *CoRR*, abs/1906.06905.

John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions. In *Eighth Annual Conference of the International Speech Communication Association*, pages 2173–2176, Antwerp, Belgium.

Y. Lin, L. Sung, and M. Chen. 2007. An automatic multiple-choice question generation scheme for english adjective understanding. In *Workshop on Modeling, Management and Generation of Problems/Questions in eLearning*, pages 137–142. 15th International Conference on Computers in Education (ICCE 2007).

Alexey Malafeev. 2014. Language exercise generation: Emulating cambridge open cloze. *Int. J. Concept. Struct. Smart Appl.*, 2(2):20–35.

Edison Marrese-Taylor, Ai Nakajima, Yutaka Matsuo, and Ono Yuichi. 2018. Learning to automatically generate fill-in-the-blank quizzes. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 152–156, Melbourne, Australia. Association for Computational Linguistics.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Irina Pandarova, Torben Schmidt, Johannes Hartig, Ahcène Boubekki, Roger Dale Jones, and Ulf Brefeld. 2019. Predicting the difficulty of exercise items for dynamic difficulty adaptation in adaptive language tutoring. *International Journal of Artificial Intelligence in Education*.

Juan Pino and Maxine Eskenazi. 2009. Measuring hint level in open cloze questions. In *Twenty-Second International FLAIRS Conference*, pages 460–465.

Juan Pino, Michael Heilman, and Maxine Eskenazi. 2008. A selection strategy to improve cloze question quality. *Intelligent Tutoring Systems for Ill-Defined Domains: Assessment and Feedback in Ill-Defined Domains.*, page 22.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–242, Sofia, Bulgaria. Association for Computational Linguistics.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Adam Skory and Maxine Eskenazi. 2010. Predicting cloze task quality for vocabulary training. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 49–56, Los Angeles, California. Association for Computational Linguistics.

Simon Smith, P. V. S. Avinesh, and Adam Kilgarriff. 2010. Gap-fill tests for language learners: Corpus-driven item generation. In *Proceedings of ICON-2010: 8th International Conference on Natural Language Processing*, pages 1–6. Macmillan Publishers.

Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, EdAppsNLP 05, pages 61–68, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuni Susanti, Hitoshi Nishikawa, Takenobu Tokunaga, and Hiroyuki Obari. 2016. Item difficulty analysis of english vocabulary questions. In *CSEDU 2016 - Proceedings of the 8th International Conference on Computer Supported Education*, volume 1, pages 267–274.

Song Lyrics Summarization Inspired by Audio Thumbnailing

Michael Fell, Elena Cabrio, Fabien Gandon, Alain Giboin
Université Côte d'Azur, CNRS, Inria, I3S, France
{firstname.lastname}@inria.fr

Abstract

Given the peculiar structure of songs, applying generic text summarization methods to lyrics can lead to the generation of highly redundant and incoherent text. In this paper, we propose to enhance state-of-the-art text summarization approaches with a method inspired by audio thumbnailing. Instead of searching for the thumbnail clues in the audio of the song, we identify equivalent clues in the lyrics. We then show how these summaries that take into account the audio nature of the lyrics outperform the generic methods according to both an automatic evaluation and human judgments.

1 Introduction

Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning of a text (Allahyari et al., 2017). Numerous approaches have been developed to address this task and applied widely in various domains including news articles (Cheng and Lapata, 2016), scientific papers (Mei and Zhai, 2008), web content as blogs (Hu et al., 2007), customer reviews (Pecar, 2018) and social media messages (He and Duan, 2018). Just as we may need to summarize a story, we may also need to summarize song lyrics, for instance to produce adequate snippets for a search engine dedicated to an online song collection or for music digital libraries. From a linguistic point of view however, lyrics are a very peculiar genre of document and generic summarization methods may not be appropriate when the input for summarization comes from a specific domain or type of genre as songs are (Nenkova et al., 2011). Compared to news documents, for instance, lyrics have a very different structure. Given the repeating forms, peculiar structure (e.g. the segmentation into verse, chorus, etc.) and other unique characteristics of song lyrics, we need the summarization algorithms to take advantage of these additional elements to more accurately identify relevant information in song lyrics. But just as such characteristics enable the exploration of new approaches, other characteristics make the application of summarization algorithms very challenging, as the presence of repeated lines, the discourse structure that strongly depends on the interrelation of music and words in the melody composition, the heterogeneity of musical genres each featuring peculiar styles and wording (Brackett, 1995), and simply the fact that not all songs tell a story.

In this direction, this paper focuses on the following research questions: *What is the impact of the context in summarizing song lyrics?*. This question is broken down into two sub questions: 1) *How do generic text summarization methods perform over lyrics?* and 2) *Can such peculiar context be leveraged to identify relevant sentences to improve song text summarization?* To answer our research questions, we experiment with generic unsupervised state-of-the-art text summarization methods (i.e. TextRank, and a topic distribution based method) to perform lyrics summarization, and show that adding contextual information helps such models to produce better summaries. Specifically, we enhance text summarization approaches with a method inspired by audio thumbnailing techniques, that leverages the repetitive structure of song texts to improve summaries. We show how summaries that take into account the audio nature of the lyrics outperform the generic methods according to both an automatic evaluation over 50k lyrics, and judgments of 26 human subjects.

In the following, Section 2 reports on related work. Section 3 presents the lyrics summarization task and the proposed methods. Sections 4 and 5 report on the experiments and on the evaluation, respectively. Section 6 concludes the paper.

Proceedings of Recent Advances in Natural Language Processing, pages 328–337,
Varna, Bulgaria, Sep 2–4 2019.

2 Summarization Methods

This section reports on the related work on both text and audio summarization methods.

2.1 Text Summarization

In the literature, there are two different families of approaches for automatic text summarization: extraction and abstraction (Allahyari et al., 2017). *Extractive summarization methods* identify important elements of the text and generate them verbatim (they depend only on extraction of sentences or words from the original text). In contrast, *abstractive summarization methods* interpret and examine the text to generate a new shorter text that conveys the most critical information from the original text. Even though summaries created by humans are usually not extractive, most of the summarization research has focused on extractive methods. Purely extractive summaries often give better results (Nallapati et al., 2016), due to the fact that latter methods cope with more complex problems such as semantic representation, inference and natural language generation. Existing abstractive summarizers often rely on an extractive pre-processing component to produce the abstract of the text (Berg-Kirkpatrick et al., 2011; Knight and Marcu, 2000). Consequently, in this paper we focus on extractive summarization methods, also given the fact that lyrics *i)* strongly use figurative language which makes abstractive summarization even more challenging; and *ii)* the choice of the words by the composer may also have an importance for capturing the style of the song.

In the following, we focus on *unsupervised* methods for text summarization, the ones targeted in our study (no available gold-standard of human-produced summaries of song texts exists). Most methods have in common the process for summary generation: given a text, the importance of each sentence of that text is determined. Then, the sentences with highest importance are selected to form a summary. The ways different summarizers determine the importance of each sentence may differ: *Statistics-based summarizers* extract indicator features from each sentence, e.g. (Fattah and Ren, 2009) use among others the sentence position and length and named entities as features. *Topic-based summarizers* aim to represent each sentence by its underlying topics. For instance, (Hennig, 2009) apply Probabilistic Latent Semantic Analysis, while Latent Dirichlet Allocation is used in (Arora and Ravindran, 2008) to model each sentence's distribution over latent topics. Another type of summarization methods is *graph-based summarizers*. Three of the most popular graph-based summarizers are TextRank (Mihalcea and Tarau, 2004), LexRank (Erkan and Radev, 2004), and (Parveen et al., 2015). These methods work by constructing a graph whose nodes are sentences and whose graph edge weights are sentence similarities. Then, the sentences that are central to the graph are found by computing the PageRank (Page et al., 1999). Contrarily to all previously described methods, systems using *supervised machine learning* form another type of summarizers. For instance, (Fattah, 2014) treats extractive summarization as a binary classification task, where they extract indicator features from sentences of gold summaries and learn to detect the sentences that should be included in a summary.

Context-Specific Summarization. If specific knowledge about the application scenario or the domain of the summarized text is available, generic summarization methods can be adapted to take into account the prior information. In query-based summarization (Otterbacher et al., 2005; Wang et al., 2016), the user's query is taken into account when generating a summary. Summarization of a scientific paper can be improved by considering the citations of it, as in (Delort et al., 2003). However, to the best of our knowledge no summarization methods have been proposed for the domain of song texts. In this paper we present a summarization method that uses prior knowledge about the text it summarizes to help generic summarizers generate better summaries.

Evaluation Criteria and Methods. Summaries should *i)* contain the most important information from input documents, *ii)* not contain redundant information, *iii)* be readable, hence they should be grammatical and coherent (Parveen and Strube, 2015). While a multitude of methods to identify important sentences has been described above, several approaches aim to make summaries less redundant and more coherent. The simplest way to evaluate summaries is to let humans assess the quality, but this is extremely expensive. The factors that humans must consider when giving scores to each candidate summary are grammaticality, non redundancy, integration of most important pieces of information, structure and coherence

(Saggion and Poibeau, 2013). The more common way is to let humans generate possibly multiple summaries for a text and then automatically assess how close a machine-made summary is to the human gold summaries computing ROUGE scores (Lin, 2004), which boils down to measuring n-gram overlaps between gold summaries and automatic summary. More recently there have been attempts to rate summaries automatically without the need for gold summaries (Nenkova et al., 2011). The key idea is that a summary should be similar to the original text in regard to characteristic criteria as the word distribution. (Mackie et al., 2014) find that topic words are a suitable metric to automatically evaluate micro blog summaries.

2.2 Audio Summarization

Lyrics are texts that accompany music. Therefore, it is worthwhile to see if methods in audio summarization can be transferred to lyrics summarization. In audio summarization the goal is to find the most representative parts in a song, in Pop songs those are usually the chorus and the bridge, in instrumental music the main theme. The task of creating short audio summaries is also known as audio thumbnailing (Bartsch and Wakefield, 2005; Chai and Vercoe, 2003; Levy et al., 2006), as the goal is to produce a short representation of the music that fits onto a thumbnail, but still covers the most representative parts of it. In a recent approach of audio thumbnailing (Jiang and Müller, 2015), the authors generate a *Double Thumbnail* from a musical piece by finding the two most representative parts in it. For this, they search for candidate musical segments in an a priori unsegmented song. Candidate musical segments are defined as sequences of music that more or less exactly repeat themselves. The representativeness of each candidate segment to the whole piece is then estimated by their fitness metric. They define the fitness of a segment as a trade-off between how exactly a part is repeated and how much of the whole piece is covered by all repetitions of that segment. Then, the audio segments along with their fitness allow them to create an audio double thumbnail consisting of the two fittest audio segments.

3 Lyrics Summarization

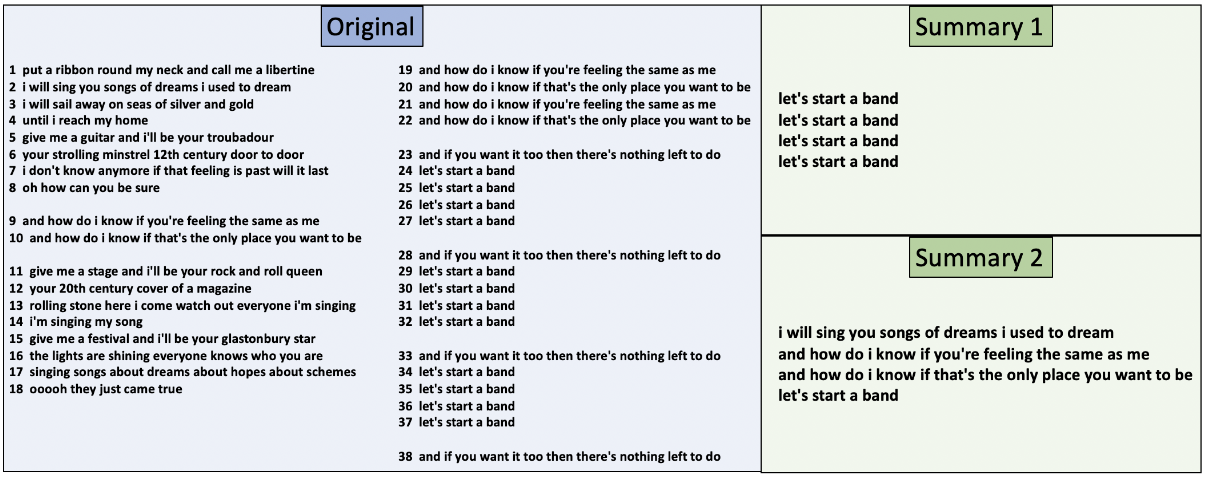
Song texts are arranged in segments and lines. For instance the song text depicted in Figure 1 consists of 8 segments and 38 lines. Given a song text S consisting of n lines of text, $S = (x_1, ..., x_n)$, we define the task of *extractive lyrics summarization* as the task of producing a concise summary sum of the song text, consisting of a subset of the original text lines: $sum(S) \subseteq S$, where usually $|sum(S)| << |S|$. We define the goal of a summary as to preserve key information and the overall meaning of a song text. To address this task, we apply the following methods from the literature: the popular graph-based summarizer TextRank; an adaptation of a topic-based method (TopSum). Moreover, we introduce a method inspired by audio thumbnailing (which we dub Lyrics Thumbnail) which aims at creating a summary from the most representative parts of the original song text. While for TextRank we rely on the off-the-shelf implementation of (Barrios et al., 2016), in the following we describe the other two methods.

3.1 TopSum

We implement a simple topic-based summarization model that aims to construct a summary whose topic distribution is as similar as possible to that of the original text. Following (Kleedorfer et al., 2008), we train a topic model by factorizing a tf-idf-weighted term-document matrix of a song text corpus (see Section 4.2) using non-negative matrix factorization into a term-topic and a topic-document matrix. Given the learnt term-topic matrix, we compute a topic vector t for each new document (song text). In order to treat t as a (pseudo-) probability distribution over latent topics t_i, we normalize t by applying $\lambda t.t/\sum_{t_i \in t} t_i$ to it. Given the distributions over latent topics for each song text, we then incrementally construct a summary by greedily adding one line from the original text at a time (same mechanism as in KLSum algorithm in (Haghighi and Vanderwende, 2009)); that line x^* of the original text that minimizes the distance between the topic distribution t_S of the original text S and the topic distribution of the incremental summary $sum(S)$:

$$x^* = \underset{x \in (S \setminus sum(S))}{\text{argmin}} \{W(t_S,\ t_{sum(S)+x})\}$$

W is the Wasserstein distance (Villani, 2008) and is used to measure the distance between two probability distributions (an alternative to Jensen-Shannon divergence (Louis and Nenkova, 2013)).



Original

1 put a ribbon round my neck and call me a libertine
2 i will sing you songs of dreams i used to dream
3 i will sail away on seas of silver and gold
4 until i reach my home
5 give me a guitar and i'll be your troubadour
6 your strolling minstrel 12th century door to door
7 i don't know anymore if that feeling is past will it last
8 oh how can you be sure

9 and how do i know if you're feeling the same as me
10 and how do i know if that's the only place you want to be

11 give me a stage and i'll be your rock and roll queen
12 your 20th century cover of a magazine
13 rolling stone here i come watch out everyone i'm singing
14 i'm singing my song
15 give me a festival and i'll be your glastonbury star
16 the lights are shining everyone knows who you are
17 singing songs about dreams about hopes about schemes
18 ooooh they just came true

19 and how do i know if you're feeling the same as me
20 and how do i know if that's the only place you want to be
21 and how do i know if you're feeling the same as me
22 and how do i know if that's the only place you want to be

23 and if you want it too then there's nothing left to do
24 let's start a band
25 let's start a band
26 let's start a band
27 let's start a band

28 and if you want it too then there's nothing left to do
29 let's start a band
30 let's start a band
31 let's start a band
32 let's start a band

33 and if you want it too then there's nothing left to do
34 let's start a band
35 let's start a band
36 let's start a band
37 let's start a band

38 and if you want it too then there's nothing left to do

Summary 1

let's start a band
let's start a band
let's start a band
let's start a band

Summary 2

i will sing you songs of dreams i used to dream
and how do i know if you're feeling the same as me
and how do i know if that's the only place you want to be
let's start a band

Figure 1: Song text of "Let's start a band" by Amy MacDonald along with two example summaries.

3.2 Lyrics Thumbnail

Inspired by (Jiang and Müller, 2015), we transfer their fitness measure for audio segments to compute the fitness of lyrics segments. Analog to an audio thumbnail, we define a Lyrics Thumbnail as the most representative and repetitive part of the song text. Consequently, it usually consists of (a part of) the chorus. In our corpus the segments are annotated (as double line breaks in the lyrics), so unlike in audio thumbnailing, we do not have to induce segments, but rather measure their fitness. In the following, we describe the fitness measure for lyrics segments and how we use this to produce a summary of the lyrics.

Lyrics Fitness Given a segmented song text $S = (S_1, ..., S_m)$ consisting of text segments S_i, where each S_i consists of $|S_i|$ text lines, we cluster the S_i into partitions of similar segments. For instance, the lyrics in Figure 1 consists of 8 segments and 38 lines and the cluster of chorus consists of $\{S_5, S_6, S_7\}$. The fitness Fit of the segment cluster $C \subseteq S$ is defined through the precision pr of the cluster and the coverage co of the cluster. pr describes how similar the segments in C are to each other while co is the relative amount of lyrics lines covered by C:

$$pr(C) = (\sum_{\substack{S_i,S_j \in C \\ i<j}} 1)^{-1} \cdot \sum_{\substack{S_i,S_j \in C \\ i<j}} sim(S_i, S_j)$$

$$co(C) = (\sum_{S_i \in S} |S_i|)^{-1} \cdot \sum_{S_i \in C} |S_i|$$

where sim is a normalized similarity measure between text segments. Fit is the harmonic mean between pr and co. The fitness of a segment S_i is defined as the fitness of the cluster to which S_i belongs:

$$\forall S_i \in C : Fit(S_i) = Fit(C) = 2\frac{pr(C) \cdot co(C)}{pr(C) + co(C)}$$

For lyrics segments without repetition the fitness is defined as zero. Based on the fitness Fit for segments, we define a fitness measure for a text line x. This allows us to compute the fitness of arbitrary summaries (with no or unknown segmentation). If the text line x occurs $f_i(x)$ times in text segment S_i, then its line fitness fit is defined as:

$$fit(x) = (\sum_{S_i \in S} f_i(x))^{-1} \cdot \sum_{S_i \in S} f_i(x) \cdot Fit(S_i)$$

Fitness-Based Summary Analog to (Jiang and Müller, 2015)'s audio thumbnails, we create fitness-based summaries for a song text. A *Lyrics Double Thumbnail* consists of two segments: one from the fittest segment cluster (usually the chorus), and one from the second fittest segment cluster (usually the bridge).[1] If the second fittest cluster has a fitness of 0, we generate a *Lyrics Single Thumbnail* solely from the fittest cluster (usually the chorus). If the thumbnail generated has a length of k lines and we want to produce a summary of $p < k$ lines, we select the p lines in the middle of the thumbnail following (Chai and Vercoe, 2003)'s "Section-transition Strategy" that

[1] We pick the first occurring representative of the segment cluster. Which segment to pick from the cluster is a potential question for future work.

they find to capture the “hook” of the music more likely.[2]

4 Experimental Setting

We now describe the WASABI dataset of song lyrics (Section 4.1), and the tested configurations of the summarization methods (Section 4.2).

4.1 Dataset

From the WASABI corpus (Meseguer-Brocal et al., 2017) we select a subset of 190k unique song texts with available genre information. As the corpus has spurious genres (416 different ones), we focus on the 10 most frequent ones in order to evaluate our methods dependent on the genre. We add 2 additional genres from the underrepresented Rap field (Southern Hip Hop and Gangsta Rap). The dataset contains 95k song lyrics.

To define the length of $sum(S)$ (see Section 3), we rely on (Bartsch and Wakefield, 2005) that recommend to create audio thumbnails of the median length of the chorus on the whole corpus. We therefore estimate the median chorus length on our corpus by computing a Lyrics Single Thumbnail on each text, and we find the median chorus length to be 4 lines. Hence, we decide to generate summaries of such length for all lyrics and all summarization models to exclude the length bias in the methods comparison[3]. As the length of the lyrics thumbnail is lower-bounded by the length of the chorus in the song text, we keep only those lyrics with an estimated chorus length of at least 4. The final corpus of 12 genres consists of 50k lyrics with the following genre distribution: Rock: 8.4k, Country: 8.3k, Alternative Rock: 6.6k, Pop: 6.9k, R&B: 5.2k, Indie Rock: 4.4k, Hip Hop: 4.2k, Hard Rock: 2.4k, Punk Rock: 2k, Folk: 1.7k, Southern Hip Hop: 281, Gangsta Rap: 185.

4.2 Models and Configurations

We create summaries using the three summarization methods described in Section 3, i.e. a graph-based (TextRank), a topic-based (TopSum), and fitness-based (Lyrics Thumbnail) method, plus two additional combined models (described below). While the Lyrics Thumbnail is generated from the full segment structure of the lyrics including its duplicate lines, all other models are fed with unique text lines as input (i.e. rendundant lines are deleted). This is done to produce less redundant summaries, given that for instance, TextRank scores each duplicate line the same, hence it may create summaries with all identical lines. TopSum can suffer from a similar shortcoming: if there is a duplicate line close to the ideal topic distribution, adding that line again will let the incremental summary under construction stay close to the ideal topic distribution. All models were instructed to produce summaries of 4 lines, as this is the estimated median chorus length in our corpus (see Section 4.1). The summary lines were arranged in the same order they appear in the original text.[4] We use the TextRank implementation[5] of (Barrios et al., 2016) without removing stop words (lyrics lines in input can be quite short, therefore we avoid losing all content of the line if removing stop words). The topic model for TopSum is built using non-negative matrix factorization with scikit-learn[6] (Pedregosa et al., 2011) for 30 topics on the full corpus of 190k lyrics.[7] For the topical distance, we only consider the distance between the 3 most relevant topics in the original text, following the intuition that one song text usually covers only a small amount of topics. The Lyrics Thumbnail is computed using String-based distance between text segments to facilitate clustering. This similarity has been shown in (Watanabe et al., 2016) to indicate segment borders successfully. In our implementation, segments are clustered using the DBSCAN (Ester et al., 1996) algorithm.[8] We also produce two summaries by combining TextRank + TopSum and TextRank + TopSum + Lyrics Thumbnail, to test if summaries can benefit from the complementary perspectives the three different summarization methods take.

Model Combination For any lyrics line, we can obtain a score from each of the applied methods. TextRank provides a score for each line, TopSum provides a distance between the topic distributions of an incremental summary and the original text, and fit provides the fitness of each line. We treat our summarization methods as blackboxes and use a simple method to combine the scores the different methods provide for each line. Given the

[2] They also experiment with other methods to create a thumbnail, such as section initial or section ending.

[3] We leave the study of other measures to estimate the summary length to future work.

[4] In case of repeated parts, the first position of each line was used as original position.

[5] https://github.com/summanlp/textrank

[6] https://scikit-learn.org

[7] loss=’kullback-leibler’

[8] eps=0.3, min_samples=2

original text separated into lines $S = (x_1, ..., x_n)$, a summary is constructed by greedily adding one line x^* at a time to the incremental summary $sum(S) \subseteq S$ such that the sum of normalized ranks of all scores is minimal:

$$x^* = \text{argmin} \bigcup_x \{\sum_A \text{R}_\text{A}(x)\}$$

Here $x \in (S \setminus sum(S))$ and $A \in \{\text{TextRank}, \text{TopSum}, \text{fit}\}$. The normalized rank $\text{R}_\text{A}(x)$ of the score that method A assigns to line x is computed as follows: first, the highest scores[9] are assigned rank 0, the second highest scores get rank 1, and so forth. Then the ranks are linearly scaled to the [0,1] interval, so each sum of ranks $\sum_A \text{R}_\text{A}(x)$ is in [0,3].

Model Nomenclature For abbreviation, we call the TextRank model henceforth M_r, the TopSum model M_s, the fitness-based summarizer M_f, model combinations M_{rs} and M_{rsf}, respectively.

5 Evaluation

We evaluate the quality of the produced lyrics summary both soliciting human judgments on the goodness and utility of a given summary (Section 5.1), and through an automatic evaluation of the summarization methods (Section 5.2) to provide a comprehensive evaluation.

5.1 Human Evaluation

We performed human evaluation of the different summarization methods introduced before by asking participants to rate the different summaries presented to them by specifying their agreement / disagreement according to the following standard criteria (Parveen and Strube, 2015):

Informativeness: The summary contains the main points of the original song text.

Non-redundancy: The summary does not contain duplicate or redundant information.

Coherence: The summary is fluent to read and grammatically correct.

Plus one additional criterion coming from our definition of the lyrics summarization task:

Meaning: The summary preserves the meaning of the original song text.

An experimental psychologist expert in Human Computer Interaction advised us in defining the questionnaire and setting up the experiment. 26 participants - 12 nationalities, 18 men, 8 women, aged from 21 to 59 - were taking a questionnaire (Google Forms), consisting of rating 30 items with respect to the criteria defined before on a Likert scale from 1 (low) to 5 (high). Each participant was presented with 5 different summaries - each produced by one of the previously described summarization models - for 6 different song texts. Participants were given example ratings for the different criteria in order to familiarize them with the procedure. Then, for each song text, the original song text along with its 5 summaries were presented in random order and had to be rated according to the above criteria. For the criterion of Meaning, we asked participants to give a short explanation in free text for their score. The selected 6 song texts[10] have a minimum and a median chorus length of 4 lines and are from different genres, i.e. Pop/Rock (4), Folk (1) and Rap (1), similar to our corpus genre distribution. Song texts were selected from different lengths (18-63 lines), genders of singer (3 male, 3 female), topics (family, life, drugs, relationship, depression), and mood (depressive, angry, hopeful, optimistic, energetic). The artist name and song title were not shown to the participants.

Results Figure 2 shows the ratings obtained for each criterion. We examine the significant differences between the models performances by performing a paired two-tailed t-test. The significance levels are: $0.05^*, 0.01^{**}, 0.001^{***}$, and $n.s.$ First, Informativeness and Meaning are rated higher** for the combined model M_{rs} compared to the single models M_r and M_s. Combining all three models improves the summaries further: both for Informativeness and Meaning the model M_{rsf} is rated higher*** than M_{rs}. Further, summaries created by M_{rsf} are rated higher*** in Coherence than summaries from any other model - except from M_f ($n.s.$ difference). Summaries are rated on the same level ($n.s.$ differences) for Non-redundancy in all but the M_r and M_f summaries, which are perceived as lower*** in Non-redundancy than all others. Note, how the model M_{rsf} is more stable than all others by exhibiting lower standard deviations in all criteria except

[9] In the case of topical distance, a "higher score" means a lower value.

[10] "Pills N Potions" by Nicki Minaj, "Hurt" by Nine Inch Nails, "Real to me" by Brian McFadden, "Somebody That I Used To Know" by Gotye, "Receive" by Alanis Morissette, "Let's Start A Band" by Amy MacDonald

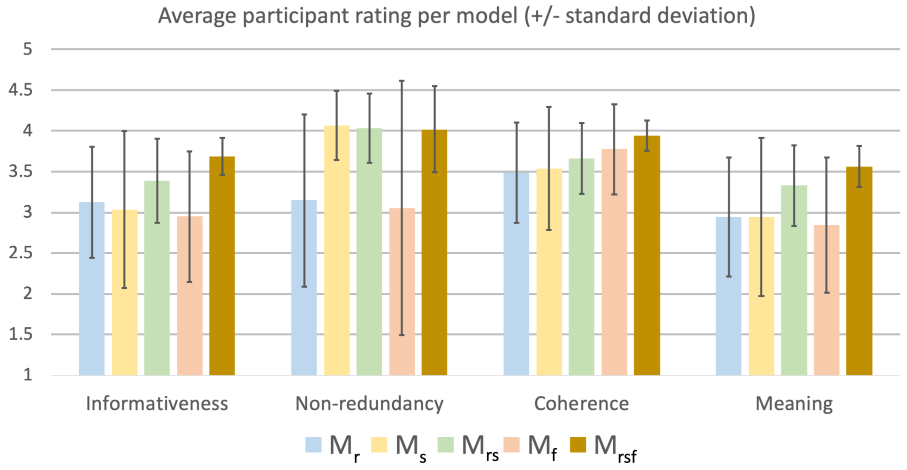


Figure 2: Human ratings per summarization model in terms of average and standard deviation.

Non-redundancy. The criteria Informativeness and Meaning are highly correlated (Pearson correlation coefficient 0.84). Correlations between other criteria range between 0.29 and 0.51.

Overall, leveraging the Lyrics Fitness in a song text summary improves summary quality. Especially with respect to the criteria that, we believe, indicate the summary quality the most - Informativeness and Meaning - the M_{rsf} method is significantly better performing and more consistent.

Figure 1 shows an example song text and example summaries from the experiment. Summary 1 is generated by M_f and consists of the chorus. Summary 2 is made by the method M_{rsf} and has relevant parts of the verses and the chorus, and was rated much higher in Informativeness and Meaning. We analyzed the free text written by the participants to comment on the Meaning criterion, but no relevant additional information was provided (the participants mainly summarized their ratings).

5.2 Automatic Evaluation

We computed four different indicators of summary quality on the dataset of 50k songs described in Section 4.1. Three of the criteria use the similarity between probability distributions P, Q, which means we compute the Wasserstein distance between P and Q (cf. Section 3.1) and apply $\lambda x.\ x^{-1}$ to it.[11] The criteria are:

Distributional Semantics: similarity between the word distributions of original and summary, cf. (Louis and Nenkova, 2013). We give results relative to the similarity of the best performing model (=100%).

Topical: similarity between the topic distributions of original and summary. Restricted to the 3 most relevant topics of the original song text. We give results relative to the similarity of the best performing model (=100%).

Coherence: average similarity between word distributions in consecutive sentences of the summary, cf. (ShafieiBavani et al., 2018). We give results relative to the coherence of the original song text (=100%).

Lyrics fitness: average line-based fitness fit (cf. Section 3) of the lines in the summary. We give results relative to the Lyrics fitness of the original song text (=100%).

Results When evaluating each of the 12 genres, we found two clusters of genres to behave very similarly. Therefore, we report the results for these two groups: the *Rap* genre cluster contains Hip Hop, Southern Hip Hop, and Gangsta Rap. The *Rock / Pop* cluster contains the 9 other genres. Results of the different automatic evaluation metrics are shown in Table 1. Distributional Semantics metrics have previously been shown (Louis and Nenkova, 2013; ShafieiBavani et al., 2018) to highly correlate with user responsiveness judgments. We would expect correlations of this metric with Informativeness or Meaning criteria therefore, as those criteria are closest to responsiveness, but we have found no large differences between the different models for this criterion. The summaries of the M_s model have the highest similarity to the original text and the M_f have the lowest similarity of 90%. The difference between the highest and lowest values are low.

For the Topical similarity, the results are mostly

[11] This works as we always deal with distances > 0.

Evaluation criterion	Genre	M_r	M_s	M_{rs}	M_f	M_{rsf}	original text
Distributional Semantics [%]	Rock / Pop	92	100	97	90	93	n/a
	Rap	94	100	99	86	92	
	$\sum$	92	100	98	90	93	
Topical [%]	Rock / Pop	44	100	76	41	64	n/a
	Rap	58	100	80	48	66	
	$\sum$	46	**100**	77	42	**64**	
Coherence [%]	Rock / Pop	110	95	99	99	100	100
	Rap	112	115	112	107	107	
	$\sum$	110	97	101	100	101	
Lyrics fitness [%]	Rock / Pop	71	53	63	201	183	100
	Rap	0	0	0	309	249	
	$\sum$	62	47	55	**214**	**191**	

Table 1: Automatic evaluation results for the 5 summarization models and 2 genre clusters. Distributional Semantics and Topical are relative to the best model (=100%), Coherence and Fitness to the original text (=100%).

in the same order as the Distributional Semantics ones, but with much larger differences. While the M_s model reaches the highest similarity, this is a self-fulfilling prophecy, as summaries of M_s were generated with the objective of maximizing topical similarity. The other two models that incorporate M_s (M_{rs} and M_{rsf}), show a much higher topical similarity to the original text than M_r and M_f.

Coherence is rated best in M_r with 110%. All other models show a coherence close to that of the original text - between 97% and 101%. We believe that the increased coherence of M_r is not linguistically founded, but merely algorithmic. M_r produces summaries of the most central sentences in a text. The centrality is using the concept of sentence similarity. Therefore, M_r implicitly optimizes for the automatic evaluation metric of coherence, based on similar consecutive sentences. Sentence similarity seems to be insufficient to predict human judgments of coherence in this case.

As might be expected, methods explicitly incorporating the Lyrics fitness produce summaries with a fitness much higher than the original text - 214% for the M_f and 191% for the M_{rsf} model. The methods not incorporating fitness produce summaries with much lower fitness than the original - M_r 62%, M_s 47%, and M_{rs} 55%. In the Rap genre this fitness is even zero, i.e. summaries (in median) contain no part of the chorus.

Overall, no single automatic evaluation criterion was able to explain the judgments of our human participants. However, considering Topical similarity and fitness together gives us a hint. The model M_f has high fitness (214%), but low Topical similarity (42%). The M_s model has the highest Topical similarity (100%), but low fitness (47%). M_{rsf} might be preferred by humans as it strikes a balance between Topical similarity (64%) and fitness (191%). Hence, M_{rsf} succeeds in capturing lines from the most relevant parts of the lyrics, such as the chorus, while jointly representing the important topics of the song text.

6 Conclusion

In this paper we have defined and addressed the task of lyrics summarization. We have applied both generic unsupervised text summarization methods (TextRank and a topic-based method we called TopSum), and a method inspired by audio thumbnailing on 50k lyrics from the WASABI corpus. We have carried out an automatic evaluation on the produced summaries computing standard metrics in text summarization, and a human evaluation with 26 participants, showing that using a fitness measure transferred from the musicology literature, we can amend generic text summarization algorithms and produce better summaries.

In future work, we will model the importance of a line given the segment to avoid cutting off important parts of the chorus, as we sometimes observed. Moreover, we plan to address the challenging task of abstractive summarization over song lyrics, with the goal of creating a summary of song texts in prose-style - more similar to what humans would do, using their own words.

Acknowledgement

This work is partly funded by the French Research National Agency (ANR) under the WASABI project (contract ANR-16-CE23-0017-01).

References

Mehdi Allahyari, Seyed Amin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. 2017. Text summarization techniques: A brief survey. *CoRR*, abs/1707.02268.

Rachit Arora and Balaraman Ravindran. 2008. Latent dirichlet allocation based multi-document summarization. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 91–97. ACM.

Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606.

Mark A. Bartsch and Gregory H. Wakefield. 2005. Audio thumbnailing of popular music using chroma-based representations. *Trans. Multi.*, 7(1):96–104.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 481–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Brackett. 1995. *Interpreting Popular Music*. Cambridge University Press.

Wei Chai and Barry Vercoe. 2003. Music thumbnailing via structural analysis. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 223–226.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Jean Yves Delort, Bernadette Bouchon-Meunier, and Maria Rifqi. 2003. Enhanced web document summarization using hyperlinks. In *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, HYPERTEXT '03, pages 208–215, New York, NY, USA. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Mohamed Abdel Fattah. 2014. A hybrid machine learning model for multi-document summarization. *Applied intelligence*, 40(4):592–600.

Mohamed Abdel Fattah and Fuji Ren. 2009. Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Comput. Speech Lang.*, 23(1):126–144.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370. Association for Computational Linguistics.

Ruifang He and Xingyi Duan. 2018. Twitter summarization based on social network and sparse reconstruction. In *AAAI*.

Leonhard Hennig. 2009. Topic-based multi-document summarization with probabilistic latent semantic analysis. In *Proceedings of the International Conference RANLP-2009*, pages 144–149.

Meishan Hu, Aixin Sun, Ee-Peng Lim, and Ee-Peng Lim. 2007. Comments-oriented blog summarization by sentence extraction. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 901–904, New York, NY, USA. ACM.

Nanzhu Jiang and Meinard Müller. 2015. Estimating double thumbnails for music recordings. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 146–150.

Florian Kleedorfer, Peter Knees, and Tim Pohle. 2008. Oh oh oh whoah! towards automatic topic detection in song lyrics. In *Ismir*, pages 287–292.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Mark Levy, Mark Sandler, and Michael Casey. 2006. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 5, pages V–V. IEEE.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2).

Stuart Mackie, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2014. On choosing an effective automatic evaluation metric for microblog summarisation. In *Proceedings of the 5th Information Interaction in Context Symposium*, pages 115–124. ACM.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *ACL*.

Gabriel Meseguer-Brocal, Geoffroy Peeters, Guillaume Pellerin, Michel Buffa, Elena Cabrio, Catherine Faron Zucker, Alain Giboin, Isabelle Mirbel, Romain Hennequin, Manuel Moussallam, Francesco Piccoli, and Thomas Fillon. 2017. WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conference 2017 – Collaborative Audio #WAC2017*, London, United Kingdom. Queen Mary University of London.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Jahna Otterbacher, Güneş Erkan, and Dragomir R Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 915–922. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954.

Daraksha Parveen and Michael Strube. 2015. Integrating importance, non-redundancy and coherence in graph-based extractive summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1298–1304. AAAI Press.

Samuel Pecar. 2018. Towards opinion summarization of customer reviews. In *Proceedings of ACL 2018, Student Research Workshop*, pages 1–8. Association for Computational Linguistics.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Mathieu Brucher, Mathieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Horacio Saggion and Thierry Poibeau. 2013. *Automatic Text Summarization: Past, Present and Future*, pages 3–21. Springer, Berlin, Heidelberg.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2018. Summarization evaluation in the absence of human model summaries using the compositionality of word embeddings. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 905–914. Association for Computational Linguistics.

Cédric Villani. 2008. *Optimal transport: old and new*, volume 338. Springer Science & Business Media.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2016. A sentence compression based framework to query-focused multi-document summarization. *arXiv preprint arXiv:1606.07548*.

Kento Watanabe, Yuichiroh Matsubayashi, Naho Orita, Naoaki Okazaki, Kentaro Inui, Satoru Fukayama, Tomoyasu Nakano, Jordan Smith, and Masataka Goto. 2016. Modeling discourse segments in lyrics using repeated patterns. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1959–1969.

Comparing Automated Methods to Detect Explicit Content in Song Lyrics

Michael Fell, Elena Cabrio, Michele Corazza, Fabien Gandon
Université Côte d'Azur, CNRS, Inria, I3S, France
{firstname.lastname}@inria.fr

Abstract

The Parental Advisory Label (PAL) is a warning label that is placed on audio recordings in recognition of profanity or inappropriate references, with the intention of alerting parents of material potentially unsuitable for children. Since 2015, digital providers – such as iTunes, Spotify, Amazon Music and Deezer – also follow PAL guidelines and tag such tracks as "explicit". Nowadays, such labelling is carried out mainly manually on voluntary basis, with the drawbacks of being time consuming and therefore costly, error prone and partly a subjective task. In this paper, we compare automated methods ranging from dictionary-based lookup to state-of-the-art deep neural networks to automatically detect explicit contents in English lyrics. We show that more complex models perform only slightly better on this task, and relying on a qualitative analysis of the data, we discuss the inherent hardness and subjectivity of the task.

1 Introduction

All content is not always appropriate for all ages and music is no exception. Content industries have been actively searching for means to help adults determine what is and is not appropriate for children. In USA, in 1985, the Recording Industry Association of America (RIAA) introduced the Parental Advisory label (PAL) in order to alert parents of content unsuitable for children because of profanity or inappropriate references[1]. PAL is "a notice to consumers that recordings identified by this mark may contain strong language or depictions of violence, sex or substance abuse"[2] and that parental discretion is advised. In UK, the British Phonographic Industry (BPI) adds to this list "racist, homophobic, misogynistic or other discriminatory language or behavior; or dangerous or criminal behavior"[3].

In the case of a song, the explicit logo is applied when the lyrics or content of a song matches one of these criteria, raising the problem of detecting and labelling explicit songs in a scalable way.

Within the Natural Language Processing (NLP) community, there have been several efforts to deal with the problem of online abusive language detection, since the computational analysis of language can be used to quickly identify offenses and ease the removal of abusive messages. Several workshops (Park and Fung, 2017; Fišer et al., 2018) and evaluation campaigns (Fersini et al., 2018; Bosco et al., 2018; Wiegand et al., 2018) have been recently organized to discuss existing approaches to abusive language detection, propose shared tasks and foster the development of benchmarks for system evaluation. These have led to the creation of a number of datasets for abusive language detection in different languages, that have been shared within the NLP research community. The SemEval 2019 tasks HatEval (Basile et al., 2019) and OffensEval (Zampieri et al., 2019) have aimed at the multilingual detection of hate speech against women or immigrants and the categorization of hate speech, respectively.

In this direction, and given the similarity with the abusive language detection task, this paper addresses the problem of explicit content detection in song lyrics as a binary classification task: a song can be labelled either as explicit or clean (=not explicit). To this end, we first compare a range of classification methods for the task of explicit lyrics detection, from dictionary lookup to deep neural networks. We then attempt the comparison to the

[1] Parental Advisory https://en.wikipedia.org/wiki/Parental_Advisory

[2] RIAA PAL https://www.riaa.com/resources-learning/pal-standards/

[3] BPI Parent Advisory https://www.bpi.co.uk/media/1047/parental-advisory-guidelines.pdf

available related works and shed light on the inherent hardness and subjectivity of the task at hand.

The paper is organized as follows: in Section 2 we survey the state of the art in explicit lyrics detection. In Sections 3 and 4 we introduce the classification methods we apply, and the comparative experimentation. Conclusions end the paper.

NOTE: This paper contains examples of language which may be offensive to some readers. They do not represent the views of the authors.

2 Related Work

Only a few works on the problem of explicit lyrics detection exist. (Bergelid, 2018) consider a dataset of English lyrics (see Table 1, B18) to which they apply classical machine learning algorithms such as Support Vector Machine (SVM) and Random Forest (RF). As features they extract either (i) tf-idf weighted bag-of-word (BOW) representations of each song text or (ii) represent the lyrics with paragraph vectors (Le and Mikolov, 2014). The explicit labels are obtained from Soundtrack Your Brand [4]. They find the RF with tf-idf BOW to perform best, especially in combination with a random undersampling strategy to the highly imbalanced dataset. They also experiment with adding lyrics metadata to the feature set, such as the artist name, the release year, the music energy level, and the valence/positiveness of a song. This results in marginal improvements for some of their models.

(Chin et al., 2018) apply explicit lyrics detection to Korean song texts. They also use tf-idf weighted BOW as lyrics representation and aggregate multiple decision trees via boosting and bagging to classify the lyrics for explicit content. On their corpus (see Figure 1, C18) they report 78% F_1 using the bagging method. Note, that bagging with decision trees is similar to the Random Forest method used by (Bergelid, 2018). Interestingly, they also report a baseline for dictionary lookup, i.e. given a profanity dictionary the song text is classified as explicit if and only if one of its words occurs in the profanity dictionary. With such a baseline they obtain 61% F_1.

More recently, (Kim and Mun, 2019) proposed a method to create explicit words dictionaries automatically by weighting a vocabulary according to all words' frequencies in the explicit class vs. the clean class, accordingly. For instance the word "fuck" is typical for explicit lyrics and atypical for clean lyrics. They compare different methods to generate such a lexicon. The achieved performances using solely dictionary lookup range from 49% F_1 for a man-made dictionary to 75.6% F_1 when using relative class frequencies. Note, that the latter performance is achieved with a dictionary of only 25 words. They work with a corpus of Korean lyrics (see Figure 1, K19). Unlike previous work, they apply a recursive neural network, resulting in 76.6% F_1, slightly higher than the simple dictionary lookup. They find performance to increase to 78.1% when combining the vector representation of the RNN with a one-hot vector indicating for each profane word from the dictionary if the lyric contains it. They argue to use the RNN to find such cases where the expliciteness arises from the context and not from a dictionary check. However, no examples of finding this phenomenon are presented.

3 Methods for Explicit Lyrics Detection

In this work, we compare a range of classification methods for the task of explicit lyrics detection. Common to all methods is that they classify a full song into one of two mutually exclusive classes - explicit or clean (=not explicit). This means, the decision if a song text is explicit is taken globally. We assess the performance of different classification methods ranging from simple dictionary lookup / lexicon checking to general purpose deep learning language understanding models. We try to identify contextual effects by applying a method that outputs the "importance" for each word (see Section 3.4).

3.1 Dictionary-Based Methods

The most straightforward way to implement an automated explicit content detection method, is checking against a dictionary of explicit words. The dictionary can be man-made or automatically created from example explicit and clean lyrics. Then, a classifier uses this dictionary to predict the class of an unseen song text.

3.1.1 Dictionary Creation

It is possible to use handcrafted dictionaries such as Noswearing [5]. Performance using an automatically created lexicon has previously been shown (Kim and Mun, 2019) to improve over the manually created dictionary. We therefore consider only

[4] https://www.soundtrackyourbrand.com

[5] https://www.noswearing.com/

the case of the machine-made dictionary in this work. We generate a dictionary of words that are indicative of explicit lyrics. We define the importance I of a word w for explicit lyrics by the frequency $f(w, ex)$ of w in explicit lyrics compared to its frequency $f(w, cl)$ in clean lyrics:

$$I(w) = f(w, ex)/f(w, cl)$$

We filter out unique and too common words and restrict the number of terms to 1,000 to avoid overreliance on terms that are very corpus specific. The dictionary D_n of the n words most important for explicit lyrics, is now straightforwardly defined as containing the n words with the highest I score.

3.1.2 Dictionary Lookup

Given a dictionary D_n, this method simply checks if a song text S contains any of the explicit terms defined in D_n. Then, S is classified as explicit iff it contains at least one explicit term from D_n.

3.1.3 Dictionary Regression

This method uses BOW made from D_n as the feature set of a classifier. We used a logistic regression, but RF or SVM have been used alike in (Bergelid, 2018).

3.2 Tf-idf BOW Regression

Similar to the Dictionary Regression, but the BOW contains the whole vocabulary of a training sample instead of only the explicit terms. The word features are weighted with the well-known tf-idf weighting scheme.

3.3 Transformer Language Model

Recently, approaches based on self-attention (Vaswani et al., 2017) have been proposed and have proven effective for natural language understanding tasks. These models are structured as an encoder-decoder, and they are trained on unsupervised tasks (such as masked language modelling) in order to learn dense representations of sentences or documents. These models differ from more traditional recurrent neural networks in different aspects. In particular, while recurrent models can process sequences (in NLP, typically word embeddings) in order, transformers use a joint model of the right and left context of each word in order to encode an entire sequence or document. Additionally, transformers are typically less computationally expensive than recurrent models, especially when trained on a GPU accelerator.

One of the most successful transformer-based models proposed in the last few years is BERT (Devlin et al., 2018). This model is composed of multiple transformers connected by residual connections. Pre-trained models are provided by the authors, and they are used in our work to perform explicit language detection in lyrics, without retraining the full model.

3.4 Textual Deconvolution Saliency

We use the Textual Deconvolution Saliency (TDS) model of (Vanni et al., 2018), which is a Convolutional Neural Network (CNN) for text classification. It is a simple model containing an embedding layer for word representations, a convolutional layer with max pooling and two fully connected layers. The interesting part about this model is that they manage to reverse the convolution. Given the learned feature map (the output of the convolution before max pooling) of the CNN, they upsample it to obtain a 3-dimensional sample with dimensions (#words, embedding size, #filters). The TDS for each word is now defined as the sum along the embedding axes of the output of the deconvolution. The TDS represents the importance of each word of the input with respect to the learned feature maps. We use this model with the goal to find local explanations for the global decision of the classification as explicit or clean. Such explanations can arise from contexts or phrases that the model assigns a high importance.

4 Experimental Setting and Evaluation

We compare the different methods as introduced in the previous section to the task of explicit lyrics detection. We attempt a comparison to the related work as well, although due to different datasets comparing the reported scores directly is problematic. We finally analyze the classification qualitatively with examples, and demonstrate the intrinsic hardness and subjectivity of the explicit lyrics detection task.

Abbreviations used: to refer to related works in Table 1 and 3, we use the following abbreviations. B18 stands for (Bergelid, 2018), C18 is (Chin et al., 2018), K19 means (Kim and Mun, 2019), while Ours is this work.

4.1 Dataset

The WASABI database (Meseguer-Brocal et al., 2017) contains song-wise labels for explicit lyrics,

Work	*total*	*explicit*	*ratio*	*language*
B18	25,441	3,310	13.0%	English
C18	27,695	1,024	3.7%	Korean
K19	70,077	7,468	10.7%	Korean
WAS	179,391	17,808	9.9%	English

Table 1: Overview of our dataset WAS (# songs) and comparison to the related works.

such as explicit, unknown, no advice available, or clean (=not explicit). These labels are provided by the music streaming service Deezer [6]. We selected a subset of English song texts from the corpus which are tagged as either explicit or clean. We filtered out duplicate lyrics and such that contain less than 10 tokens. Finally, our dataset (WAS) comprises of 179k lyrics, with a ratio of explicit lyrics of 9.9%. The details and comparison with related works datasets are depicted in Table 1.

For training any of the models described in the previous section, we once randomly split the data into training-development-test sets with the common 60%-20%-20% ratio. We tuned the hyperparameters of the different classification algorithms on the development set to then test with the best performing parameters on the test set. As evaluation metrics we use precision (*P*), recall (*R*), and f-score (F_1). Unless stated otherwise, the scores are macro-averaged over the two possible classes.

4.2 Hyperparameters

For the dictionary-based methods, we found the ideal dictionary size to be 32 words for the lookup and 128 words for the regression. The Tf-idf BOW regression performed best when the full vocabulary of unigrams and bigrams was used. We used the sklearn implementation of logistic regression with the class weighting scheme 'balanced' to account for the class imbalance in the dataset. We used TDS with max sequence length 512 and dropout probability 50%. As is the default with TDS, corpus-specific word vectors were trained using Word2Vec (Mikolov et al., 2013) with dimensionality 128. The BERT model comes pretrained and no further pre-training was performed. We used the smaller of the two published models. BERT then was finetuned to our task using max sequence length 256 and batch size 16, otherwise default parameters for text classification task learning.

4.3 Results

Overall, the results of the different classification methods we tried are all close to each other. The simple dictionary lookup with 32 words performs comparably to the deep neural network with 110M parameters (BERT base model). As baseline, we include the majority class classifier that always predicts the clean class. Furthermore, all related works show similar tendencies of performance on their respective datasets. The results of all the different methods we applied are depicted in Table 2 and described in the following.

The majority class classifier delivers a performance of 47.4% F_1, which is the only outlier in the sense that this is far below any other model. The dictionary lookup with a vocabulary of the 32 most indicative explicit words obtains a balanced performance as precision and recall are close to each other, the overall performance is 77.3% F_1. The dictionary regression performs somewhat better in terms of f-score (78.5% F_1), achieving this with the highest overall recall of 81.5%, but it has lower precision. The tf-idf BOW regression performs very similarly to the dictionary regression. This proves that a limited number of words influences the overall performance of the models, and that they do not need to consider the whole vocabulary, just the most offensive words. The increased vocabulary of 929k unigrams and bigrams is gigantic compared to the explicit words dictionary (32 words). As most of these n-grams may be noise to the classifier, this could explain the slight decrease in performance over the dictionary regression. Finally, the neural-network-based methods behave a bit differently: the BERT language model is clearly better in precision (84.4%) over all other models - the second best is TDS with 81.2%. However, BERT performs the worst in recall with only 73.7%. The overall performance of BERT is average with 77.7% F_1. Finally, TDS performs best in terms of 79.6% F_1. We tested if TDS outperforming BERT was due to TDS using domain-specific word vectors trained on our corpus (BERT is trained on books and Wikipedia). This was not the case as TDS performed almost identically, when using generic word vectors (GloVe, 200d): 80.4% *P*, 78.7% *R*, 79.5% F_1.

A closer look at the classification performance shows that the F_1 scores for the minority class (explicit lyrics) is highest with TDS (63%) and lowest with the dictionary lookup (58.9%). The majority

[6] https://www.deezer.com

Model	*P*	*R*	F_1
Majority Class	45.0	50.0	47.4
Dictionary Lookup	78.3	76.4	77.3
Dictionary Regression	76.2	81.5	78.5
Tf-idf BOW Regression	75.6	81.2	78.0
TDS Deconvolution	81.2	78.2	79.6
BERT Language Model	84.4	73.7	77.7

Table 2: Performances of our different models on the WAS dataset. Values in percent.

Work	*Model*	F_1
Ours	Dictionary Lookup	77.3
Ours	Dictionary Regression	78.5
C18	Man-made Dictionary	61.0
K19	Man-made Dictionary	49.0
K19	Dictionary Lookup	75.6
Ours	Tf-idf BOW Regression	78.0
C18	Tf-idf BOW	78.0
C18	Tf-idf BOW+	80.0
B18	Tf-idf BOW	67.5
B18	Tf-idf BOW+	82.6
Ours	TDS Deconvolution	79.6
Ours	BERT Language Model	77.7
K19	HAN	76.7
K19	HAN + Dictionary	78.1

Table 3: Performances of dictionary-based methods (top), tf-idf BOW models (middle) and deep models (below). Note that different works use different datasets. Ours always uses the WAS dataset. Values in percent.

class (clean lyrics) on the other hand is best detected by BERT (96.3% F_1) and worst with the tf-idf BOW (95.1% F_1).

We attempt a comparison of the different approaches used in the different related works as well as ours. While the scores achieved (see Table 3) are not strictly comparable, we can see clear tendencies. According to K19, a man-made dictionary is inferior to an automatically generated one. This is supported by the man-made lexicon in C18 performing subpar to their tf-idf BOW. An appropriate lexicon of explicit terms, on the other hand, can compete with a tf-idf BOW model, as we showed with both the dictionary lookup and the regression performance. This is further supported by the generated dictionary of K19 which competes with the deep HAN model. Optimizations to the standard tf-idf BOW models are marked with the + sign. Restricting the POS tags to more likely ones found in explicit terms (C18) improves performance slighly. Using random undersampling to fight the imbalanced class problem (B18) increases performance drastically, however makes the problem somewhat different from the imbalanced problem. The final takeaway is that deep models do not necessarily outperform shallow models. Neither HAN, TDS, nor BERT deliver much higher scores than the dictionary-based or the BOW method.

4.4 Qualitative Analysis

In this section we analyze examples of explicit content lyrics and point to the inherent hardness and subjectivity in classifying and even labelling such data.

4.4.1 Explicitness in Context?

The highest difference in model performance we measured between the deep TDS model (79.6% F_1) and the dictionary lookup (77.3% F_1). We analyzed why the TDS method performed better than the dictionary lookup by inspecting those examples that (i) were explicit, (ii) were tagged as clean by the dictionary lookup, and (iii) were detected as explicit by TDS with high confidence. [7]

From the 13 examples analyzed, we found three main phenomena: (1) Four texts contained explicit terms that were not contained in the dictionary of explicit terms. Words such as *f**kin', motherf**kers* were too rare to be included in the generated lexicon and other words like *fucking, cunt, cum, shit* were not uniquely contained in explicit lyrics. The reason why this is the case can be traced back to problems in the annotations or the fact that these words are relatively frequently used in lyrics. (2) Five texts whose explicitness arises in context rather than on a word level. Examples with violent context found were "organization with horns of satan performs the ancient rituals" or "bombin on mc's, crushin crews with ease". There were also instances of sexual content such as "give it to him down in the parking lot in the backseat, in the backseat of the car". Note that the words {give, it, to, him} in isolation do not belong to an explicit terms list and the sexuality arises from the context. Similarly in "(turn the lights on) so i can see that ass work". Also here, putting "ass" in an explicit terms dictionary is tempting but may not be ideal,

[7] The last layer of TDS outputs probabilities for the input text being explicit or clean. We looked at examples where the explicit class was predicted with at least 80% probability.

as its meaning is not necessarily explicit. (3) Four texts appeared to have been mislabelled since no explicitness could be found. We found for three of them that the album the song is contained in is tagged as explicit. In cases as these, inheriting the label from the album is wrong, but it seems this is exactly what had happened here. In one Raggae lyric, in particular, we found no explicit content, so we suspect the song was mislabelled.

Since we found some annotation to be problematic, we will discuss difficulties that arise from annotating explicitness in lyrics.

4.4.2 How Hard is this Task?

As stated in the introduction, the explicit label is voluntary and we will argue that it is also somewhat subjective in its nature. There are lyrics which are not tagged as explicit although they have profanity in them. Consider for example the song *Bitch* by Meredith Brooks. While it already contains profanity in the title, it does not carry the explicit label and one can argue that in the context of the song, the term "bitch" is used as a contrastive term and to raise attention to the struggle the songwriter sees in her life, torn between potentially conflicting expectations of society ("I'm a little bit of everything - All rolled into one - I'm a bitch, I'm a lover - I'm a child, I'm a mother - I'm a sinner, I'm a saint - I do not feel ashamed").

Another example is *Check Your Head* by Buckcherry where it says "Ooh and you still bitch about your payments" where "bitch" is used as a verb and one can argue that the acceptance in this verb form is higher than in the noun form. A similar case where the part of speech influences the perceived level of profanity is *Hail Hail Rock 'n' Roll* by Discipline. It contains the line "the band starts to play loud as fuck".

We encounter a different kind of problem when dealing with substance abuse or other drug-related content. It is evident that the legal status of the substances mentioned plays a major role in how such content is labelled. This is further complicated by the fact that legislation about substances can vary wildly between different countries. The labels applied to this content are not culture-invariant, and furthermore changes in the societal view can lead to labels that are not relevant anymore. This, like other examples, shows why the labels applied to lyrics are subject to change in different cultures and time periods.

Another aspect that is very sensitive to time periods and cultures comes from words themselves: an inoffensive word can become offensive in slang or common language. One such example can be found in Johnny Cash's *The Christmas Guest*: "When the cock was crowing the night away - The Lord appeared in a dream to me". Here, cock means male chicken, as opposed to the offensive meaning that is now arguably more common.

We finally want to raise attention to the problem of genre confounding. We found that the genre *Hip Hop* contributed by far the most to all explicit lyrics - 33% of all *Hip Hop* lyrics. Since only about 5% of the whole corpus are tagged as *Hip Hop*, this genre is highly overrepresented. This raises the question in how far our task is confounded with genre classification. When inspecting the explicit terms dictionaries we have created, we clearly see that genre bias reflected. The dictionary of 32 terms that we used for the dictionary lookup method consists approximately half of terms that are quite specific to the Rap genre, such as glock, gat, clip (gun-related), thug, beef, gangsta, pimp, blunt (crime and drugs). Finally, the terms holla, homie, and rapper are arguably no causes for explicit lyrics, but highly correlated with explicit content lyrics. Biasing an explicit lyrics detection model away from genres is an interesting future direction of work.

5 Conclusion

Classifying song lyrics as explicit or clean is an inherently hard task to accomplish since what is considered offensive strongly depends on cultural aspects that can change over time. We showed that shallow models solely based on a dictionary of profane words achieve a performance comparable to deep neural networks. We argued that even the hand-labelling is highly subjective, making it problematic to automatically detect if a song text should be tagged as explicit or clean.

We propose as a possible simplification and objectification to study the local detection of explicit content. If we present an authority a report on found trigger words, found contextual sexual content, and alike, they can come to their own subjective conclusion about the final label of the text.

Acknowledgement

This work is partly funded by the French Research National Agency (ANR) under the WASABI project (contract ANR-16-CE23-0017-01).

References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63.

Linn Bergelid. 2018. Classification of explicit music content using lyrics and music metadata.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 hate speech detection task. In *Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy.*

Hyojin Chin, Jayong Kim, Yoonjong Kim, Jinseop Shin, and Mun Y Yi. 2018. Explicit content detection in music lyrics using machine learning. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 517–521. IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018. Overview of the task on automatic misogyny identification at ibereval 2018. In *IberEval@SEPLN*, volume 2150 of *CEUR Workshop Proceedings*, pages 214–228. CEUR-WS.org.

Darja Fišer, Ruihong Huang, Vinodkumar Prabhakaran, Rob Voigt, Zeerak Waseem, and Jacqueline Wernimont. 2018. Proceedings of the 2nd workshop on abusive language online (alw2). In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Association for Computational Linguistics.

Jayong Kim and Y Yi Mun. 2019. A hybrid modeling approach for an automated lyrics-rating system for adolescents. In *European Conference on Information Retrieval*, pages 779–786. Springer.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Gabriel Meseguer-Brocal, Geoffroy Peeters, Guillaume Pellerin, Michel Buffa, Elena Cabrio, Catherine Faron Zucker, Alain Giboin, Isabelle Mirbel, Romain Hennequin, Manuel Moussallam, Francesco Piccoli, and Thomas Fillon. 2017. WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conference 2017 – Collaborative Audio #WAC2017*, London, United Kingdom. Queen Mary University of London.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45. Association for Computational Linguistics.

Laurent Vanni, Mélanie Ducoffe, Carlos Aguilar, Frederic Precioso, and Damon Mayaffre. 2018. Textual deconvolution saliency (tds): a deep tool box for linguistic analysis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 548–557.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018).*

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *CoRR*, abs/1903.08983.

Linguistic Classification:
Dealing Jointly with Irrelevance and Inconsistency

Laura Franzoi
Faculty of Mathematics
and Computer Science
University of Bucharest
laura.franzoi@gmail.com

Andrea Sgarro
DMG
University of Trieste
sgarro@units.it

Anca Dinu
Faculty of
Foreign Languages
and Literatures
University of Bucharest
ancaddinu@gmail.com

Liviu P. Dinu
Faculty of Mathematics
and Computer Science
University of Bucharest
liviu.p.dinu@gmail.com

Abstract

In this paper we present new methods for language classification which put to good use both syntax and fuzzy tools, and are capable of dealing with irrelevant linguistic features (i.e. features which should not contribute to the classification) and even inconsistent features (which do not make sense for specific languages). We introduce a metric distance, based on the generalized Steinhaus transform, which allows one to deal jointly with irrelevance and inconsistency. To evaluate our methods, we test them on a syntactic data set, due to the linguist G. Longobardi and his school. We obtain phylogenetic trees which sometimes outperform the ones obtained by Atkinson and Gray (Gray and Atkinson, 2003; Bouckaert et al., 2012).

1 Introduction

According to Ethnologue (Eth, 2018), there are around 7000 living natural languages in the world, and one of the most interesting topics (not only in the academic field, but also in the general public) is their classification. While the comparative method was the main method of classifying natural languages until the 90s, the last decades brought an increasing number of computational approaches for estimating the historical evolution of languages and their relationships. Most of the computational historical linguistics approaches rely on the use of lexical items. In contrast, very few of them take into account syntactic aspects. Moreover, fuzzy tools and information theory were employed quite sparsely in language classification tasks (Ciobanu et al., 2018), in spite the inherent fuzzy nature of the natural language data.

This paper is based on previous work on fuzzy string distances and linguistic classification started in (Franzoi and Sgarro, 2017a,b; Franzoi, 2017), and inspired by the path-breaking ideas put forward back in 1967 (Muljačić, 1967) by the Croat linguist Ž., Muljačić. The technical tool which will be used in this paper is the *general Steinhaus transform*, or *biotope transform*, applied to crisp strings which are however affected by irrelevance and inconsistency, as happens with data due to the linguist G. Longobardi and his school. Fuzziness in linguistics has been seldomly treated (Franzoi and Sgarro, 2017a,b; Dinu et al., 2018), as compared to crisp approaches.

In his 1967 paper Muljačić, even if only rather implicitly, had introduced what appears to us as a natural *fuzzy* generalization of crisp Hamming distances between binary strings of fixed length n, and this only two years after Zadeh's seminal work (Zadeh, 1965): the aim was showing that Dalmatic, now an extinct language, is a bridge between the Western group of Romance languages and the Eastern group, mainly Romanian. The situation is the following: Romance languages $L, \Lambda, \ldots$ are each described by means of n features, which can be present or absent, and so are encoded by string $s(L) = \underline{x} = x_i \ldots x_n$, where x_i is the truth value of the proposition *feature i is present in language L*; however, presence/absence is sometimes only vaguely defined and so each $x = x_i$ is rather a truth value $x \in [0,1]$ in a multi-valued logic as is fuzzy logic; $x = x_i$ is *crisp* only when either $x = 0 =$ false $=$ absent or $x = 1 =$ true $=$ present, else x is *strictly fuzzy*. So, the mathematical objects one deals with are *strings* $\underline{x}, \underline{y}, \ldots$ of length n, each of the n components being a real number in the interval $[0,1]$, and moreover *distances* between such objects, since the classifications are all distance-based. In what follows, rather than Muljačić distance, we need string distances obtained by use of the *Steinhaus transform*, cf. (Dinu et al., 2018), and the *generalized Steinhaus transform*; they are all *metric* distances, in particular they verify the triangle equality. Unlike the case of Muljačić distances, which span the interval $[0, n]$, these distances are *normalized* to the interval $[0, 1]$. Steinhaus transforms allow one to deal with *irrelevance* and *inconsistency* in linguistics, as we already argued in (Dinu et al., 2018), and not only with vagueness, or fuzziness, as in Muljačić case, cf. (Muljačić, 1967; Franzoi and Sgarro, 2017a); the reason to use the *generalized* Steinhaus transform, as we do here, is that it allows one to deal *jointly* with both irrelevance and inconsistency.

Based on arguments defended by the linguist G. Longobardi and his school, cf. (Bortolussi et al., 2011; Longobardi et al., 2016, 2013, 2015), if a feature i has a low truth value in two languages L and Λ, then that feature is scarcely relevant: in fact, in the practice of linguistics the values 0 and 1 have a very *asymmetric* use, and the fact that languages L and Λ both have zero in a position i means that such an irrelevant feature i should *not* really contribute to the distance between the two languages. Technically, one should move from Hamming distances to (normalized) Jaccard distances. To achieve the goal, the convenient tool we have used was the *Steinhaus transform*, cf. (Dinu et al., 2018), which is known to preserve metricity and which is general enough so as to amply cover also the fuzzy situation: one starts from a distance like Muljačić distance $d_M(\underline{x}, \underline{y})$, and obtains its Steinhaus transform, in this case a *fuzzy Jaccard distance* $d_J(\underline{x}, \underline{y})$ for fuzzy strings $\underline{x}$ and $\underline{y}$; starting from the usual *crisp* Hamming distance the transform gives the usual *crisp* Jaccard distance.

In general, to apply a Steinhaus transformation one needs a *pivot string*, which in the Jaccard case is the all-0 string $\underline{z} = \underline{0} = (0, \ldots, 0)$. In the transform, actually, any other string $\underline{z}$ might be used, cf. (Dinu et al., 2018), as we do here so as to cover the case of *logical inconsistency*, as appears in the data due to G. Longobardi: his school is involved in an ambitious and innovative project on language classification based on *syntax*, cf. (Bortolussi et al., 2011; Longobardi et al., 2016); languages are represented through yes-no strings of length 53, each string position corresponding to a syntactic feature which can be present or absent. In his notation Longobardi uses + if a feature is present, - if it is absent, 0 if it is undefined; in our case, cf. Tables 1, 2, we write 1 if a feature is present, 0 if it is absent, * if it is undefined. Actually, due to a complex network of logical implications which constrain features, some positions might be undefined (logically inconsistent). For example, in Longobardi's classification, feature 34 is defined if and only if feature 8 is set to + and either feature 9 is set to + or feature 18 is not set to + (or both); otherwise it will be "neutralized" (*inconsistent*)[1]. This property does not hold true for Ptg (Portuguese), OE (Old English) and Ice (Icelandic).
All this establishes an extremely complex network of logical dependencies in Longobardi's data, and makes it necessary, if one wants to cover also this new intriguing facet, to suitably generalize crisp Hamming distances, or crisp Jaccard distances, respectively: in Longobardi's approach, cf. (Bortolussi et al., 2011; Longobardi et al., 2016, 2013, 2015), the two distances for ternary strings one defines and uses are quite useful, but unfortunately they violate the triangle property, and so are not metric. In this paper we propose one *metric* alternative based on the generalized Steinhaus transform (or generalized biotope transform): the star $*$ will be replaced by the totally ambiguous truth value $\frac{1}{2}$, and the pivot strings in the transform will be given by the set compound by the all-$\frac{1}{2}$ string, i.e. the totally ambiguous string $\underline{z} = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$ (which stands for inconsistency) and all-0 string $\underline{z} = (0, \ldots, 0)$, i.e. the totally false string, which

[1] Feature 34 stands for *checking possessives*: it opposes languages like French, wherein possessives occur without any visible article (*mon livre* vs. *le mon livre*), to those like Italian, in which a visible determiner is possible and normally required instead (*il mio libro* vs. *mio libro*). This feature seems to conceptually and typologically depend on full grammaticalization of definiteness (feature 8). Also, it is relevant only in languages with strong Person in D (feature 9) or without strong article (feature 18), because otherwise the language would have GenS with determiner-like function, cf. (Longobardi et al., 2013). Feature 8 asks if a language generalizes the overt marking of definiteness to all relevant cases. Feature 9 (*Strong Person*) defines whether attraction to the D area of referential nominal material (e.g. proper names) is overt (e.g. Romance) or not (e.g. English). Feature 18 (*Strong Article*) is presence of an indefinite article, i.e. of an obligatory marker on singular indefinite count argument nominals, distinct from those used for definite and mass indefinite, cf. (Longobardi et al., 2013).

stands for irrelevance. The idea is to play down not only the contribution of 0's and $\frac{1}{2}$'s separately, as we have done in (Dinu et al., 2018), but rather the contribution of both 0's and $\frac{1}{2}$'s *jointly*. It will turn out that in this case, which is not genuinely fuzzy, rather than to Muljačić distances, the generalized Steinhaus transform had been better applied to the usual *taxicab distance* (Manhattan distance, Minkowski distance), re-found when the standard fuzzy logical operators of *min* and *max* for conjunctions and disjunctions are replaced by Łukasiewicz T-norms and T-conorms, cf. (Franzoi and Sgarro, 2017b; Dinu et al., 2018).

The paper is divided as follow: in Section 2 we shortly re-take both fuzzy Hamming distances, or Muljačić distances, and taxicab distances stressing how the latter relate to Łukasiewicz T-norms; in Section 3 we introduce Steinhaus transform and we apply it to taxi-cab or Łukasiewicz distances; in Section 4 we introduce the general Steinhaus transform to deal with irrelevance and inconsistency *jointly* and we comment on our linguistic results; in Section 5 we sum up our results.

2 Fuzzy Hamming Distances vs. Łukasiewicz or Taxicab Distances

We need some notations and definitions: we set $x \wedge y \doteq \min[x, y]$, $x \vee y \doteq \max[x, y]$ and $\overline{x} \doteq 1 - x$; these are the truth values of conjunction AND, disjunction OR and negation NOT, w.r. to propositions with truth values x and y in *standard fuzzy logic*, a relevant form of multi-valued logic; $x \in [0, 1]$. Define the *fuzziness* of the truth value x to be $f(x) \doteq x \wedge (1 - x)$. For the truth values x and y in $[0, 1]$ we say that x and y are *consonant* if either $x \vee y \leq \frac{1}{2}$ or $x \wedge y \geq \frac{1}{2}$, else they are *dissonant*; let $\mathcal{D}$ and $\mathcal{C}$ denote the set of dissonant and consonant positions i, respectively. We define the following distance for strings $\underline{x}, \underline{y} \in [0, 1]^n$:

$$d_M(\underline{x}, \underline{y}) \doteq \sum_{i \in \mathcal{D}} [1 - [f(x_i) \vee f(y_i)]] + \sum_{i \in \mathcal{C}} [f(x_i) \vee f(y_i)] \quad (1)$$

This expression stresses the link with *crisp* Hamming distances for binary strings $\in \{0, 1\}^n$, but its meaning is better understood due to the following fact: each of the n *additive* terms summed is the truth value of the statement:

[(feature f_i is present in L and absent in Λ) or (feature f_i is absent in L and present in Λ)]

since, as soon proved, cf. e.g. (Franzoi and Sgarro, 2017a), for two truth values x and y one has $(x \wedge \overline{y}) \vee (\overline{x} \wedge y)$ equal to $f(x_i) \vee f(y_i)$ or to $1 - [f(x_i) \vee f(y_i)]$ according whether there is consonance or dissonance. This distance, called henceforth Muljačić distance (and called *Sgarro distance* in (Deza and Deza, 2009), cf. also (Sgarro, 1977)) is simply a natural generalization of crisp Hamming distances to a fuzzy setting. As for alternative logical operators for conjunctions and disjunctions (different T-norms and T-conorms, for which cf. e.g. (Dubois et al., 2000)), they have been discussed in (Franzoi and Sgarro, 2017b). From a metric point of view, the only attractive choice, beside fuzzy Hamming distances, turned out to be Łukasiewicz T-norms for conjunctions and the corresponding T-conorms for disjunctions:

$$x \top y \doteq (x + y - 1) \vee 0, \quad x \bot y \doteq (x + y) \wedge 1$$

One soon checks that in this case, rather curiously, $(x \top \overline{y}) \bot (\overline{x} \top y)$ turns out to be simply $|x - y|$, and so the string distance one obtains is nothing else but the very well-known taxicab distance $d_T(\underline{x}, \underline{y}) = \sum_i |x_i - y_i|$, which in our context, when it is applied to fuzzy strings of length n, might be also legitimately called *Łukasiewicz distance*.

If we consider the fuzziness $f(x) \doteq d(x, x)$ of a logical value x and if we use the Muljačić distance, then we get $f_M(x) = x \wedge (1 - x)$; if we use instead the Łukasiewicz distance, then the fuzziness is always 0.

However, if we consider another equally legitimate definition of fuzziness, namely "ambiguity - crispness", which can be formalized as $\frac{1}{2} - d\left(x, \frac{1}{2}\right)$, then if we use the Muljačić distance the new fuzziness is 0, but if we use the Łukasiewicz distance it is $f_T(x) = \frac{1}{2} - d_T\left(x, \frac{1}{2}\right) = x \wedge (1 - x)$: the result of the competition Muljačić distance vs. Łukasiewicz distance turns out to be a tie. In the next Section we explain why, with Longobardi's data, we decided to resort to taxicab distances.

The distance in (1) is a *fuzzy metric distance*, cf. (Sgarro, 1977; Franzoi and Sgarro, 2017a), from which a standard metric distance is soon obtained by imposing that self-distances $d_M(\underline{x}, \underline{y})$ should be 0, while, unless $\underline{x}$ is crisp (i.e. belong to $\{0, 1\}^n$, the set of the 2^n binary strings of length n), the value given by (1) would be strictly positive.

As for taxicab or Łukasiewicz distances, the self-distance $d_T(\underline{x},\underline{y})$ is always zero even when the argument $\underline{x}$ is not crisp, a possibly unpleasant fact in a fuzzy context (but not in ours), as argued in (Franzoi and Sgarro, 2017b).

3 Steinhaus Transforms

In the general situation, one has objects $x, y, \ldots$, not necessarily strings, a *metric* distance $d(x,y)$ between objects, and a special object z called the "*pivot*-object". The Steinhaus transform, cf. (Deza and Deza, 2009), itself proven to be a metric distance, is:

$$\mathcal{S}_d(x,y) \doteq \frac{2d(x,y)}{d(x,y)+d(x,z)+d(y,z)}$$

set equal to zero when $x = y = z$.

In our case the objects are strings and *pivots* $\underline{z}$ will always be *constant* strings $\underline{z} = (z, \ldots, z)$, $z_i = z$, $\forall i, z \in [0,1]$.

If one starts with the crisp Hamming distance, one obtains the usual crisp Jaccard distance (distances from the pivot are then Hamming *weights*); starting with the more general fuzzy Hamming distance, or Muljačić distance, one has an appropriate Jaccard-like generalization, which weighs only "little" a position where both x and y are "almost 0", and which accounts for irrelevance in itself, but not for inconsistency, as instead we need.

If the term $d_M(\underline{x},\underline{z})$ is equal to the *fuzzy Hamming weight* $w(\underline{x}) \doteq \sum_i x_i$ for $\underline{z} = \underline{0}$, it is equal to $\frac{n}{2}$ independent of $\underline{x}$ when $\underline{z} = \frac{1}{2}$, a constant pivot string which we shall need to deal with inconsistency. The fact that $d_M(\underline{x},\underline{z})$ with $\underline{z} = \frac{1}{2}$ is independent of $\underline{x}$ is a serious drawback, indeed. This is why in the case of Longobardi's data, we have applied the Steinhaus transform, rather than to the fuzzy Hamming distance or Muljačić distance, directly to the taxicab distance or Łukasiewicz distance $d_T(\underline{x},\underline{y})$. In this case, in the denominator of the corresponding Steinhaus transform, the fuzzy Hamming weight $w(\underline{x})$ is replaced by $d_T(\underline{x},\underline{z}) = \sum_i \left| x_i - \frac{1}{2} \right|$. In the next Section, more ambitiously, we shall deal *jointly* with both irrelevance and inconsistency.

4 Dealing with Irrelevance and Inconsistency

In (Franzoi and Sgarro, 2017a,b; Franzoi, 2017; Dinu et al., 2018) one has presented new methods for language classification, testing them on data sets due to Muljačić and Longobardi. So far we have dealt separately with irrelevance and inconsistency, but a question arises spontaneously: can we consider jointly both irrelevance and inconsistency? Does a mathematical tool which takes into account both of them exist? The answer is yes and the tool we are looking for is the *generalized Steinhaus transform* or *generalized biotope transform*, cf. (Deza and Deza, 2009).

Prompted by arguments defended by G. Longobardi and his school, cf. (Bortolussi et al., 2011; Longobardi et al., 2016, 2013, 2015), the novelty of this section is that, since in the language classifications features can be irrelevant or inconsistent, we want to consider both aspects together.

As we said above the idea is to play down not only the contribution of 0's, as in the case of irrelevance, but also the contribution of the $\frac{1}{2}$-positions. Unlike ours, Longobardi's non-metric distance gets rid of irrelevant and inconsistent positions in quite a drastic way, possibly a serious draw-back, as we comment in our Conclusions.

The generalized Steinhaus transform, or generalized biotope transform, is:

$$\mathcal{S}_d(x,y) = \frac{2d(x,y)}{d(x,y) + \inf_{z \in M}(d(x,z)+d(y,z))} \tag{2}$$

where M is the set of pivots we are considering, cf. (Deza and Deza, 2009).

We tackle Longobardi's data (or rather to a sample of his languages, since the data he and his school are providing are steadily improving and extending), data which are not really fuzzy, even if we have decided to "simulate" logical inconsistency by *total fuzziness*. In this case the number of features is 53, and the languages are: Sic = Sicilian, Cal = Calabrese as spoken in South Italy, It = Italian, Sal = Salentin as spoken in Salento, South Italy, Sp = Spanish, Fr = French, Ptg = Portuguese, Rm = Romanian, Lat = Latin, ClG = Classical Attic Greek, NTG = New Testament Greek, BoG = Bova Greek as spoken in the village of Bova, Italy, Gri = Grico, a variant of Greek spoken in South Italy, Grk = Greek, Got = Gothic, OE = Old English, E = English, D = German, Da = Danish, Ice = Icelandic, Nor = Norwegian, Blg = Bulgarian, SC = Serbo Croatian, Slo = Slovenian, Po = Polish, Rus = Russian, Ir = Gaelic, Wel = Welsh, Far = Farsi, Ma = Marathi, Hi = Hindi, Ar = Arabic, Heb = Hebrew or 'ivrit, Hu = Hungarian, Finn = Finnish, StB = Standard Basque, WB = Western

Basque, Wo = Wolof as spoken mainly in Senegal. For comparison reasons, we have selected a part of Longobardi's data set compound by 38 languages; taking $M = \left\{\underline{0}, \underline{\frac{1}{2}}\right\}$ in (2), the UPGMA tree we obtain is given in the following figure:

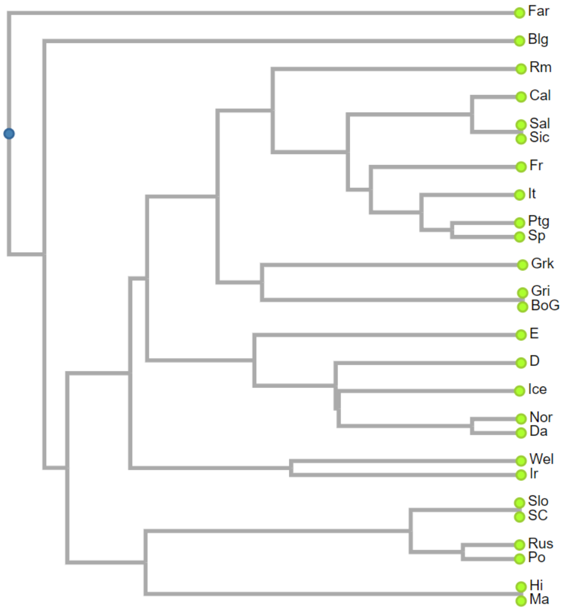


Figure 1: Generalized Steinhaus transform with taxi-cab distance and Longobardi's data

while the Longobardi's original tree is the following one:

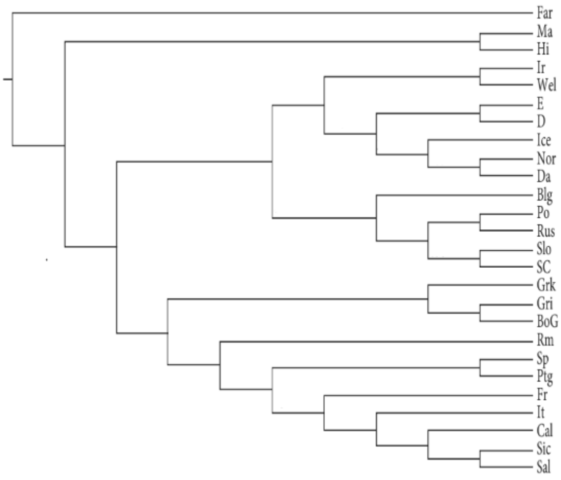


Figure 2: Longobardi's classification tree

We can observe that the Romance languages are grouped together. However there are some differences between the two trees: in our tree (Fig. 1) the big Romance languages (i.e. Italian, Spanish, Portuguese and French) are grouped together and Italian is more integrated with the Ibero-Romance languages (i.e. Portuguese ans Spanish), which are clustered together like in the standard language classifications. The three Italian dialects (i.e. Salentine, Sicilian and Calabrese) are external to this cluster in our case in Fig. 1, while in the original Longobardi's tree (Fig. 2) they are integrated with Italian and then the entire group is linked with French and after with the Ibero-Romance group. In both trees the Romanian is grouped with Romance languages, but is the most exterior with the languages from this group. In both trees the Celtic languages Gaelic (Ir) and Welsh (Wel) and Germanic languages are grouped together, but in the Longobardi's tree in Fig. 2 the Celtic group is more integrated with the Germanic group. There are two main differences between the two trees: the first one is that in Longobardi's tree in Fig. 2 Bulgarian is grouped with Slavic languages; the second one is the moving of the entire Slavic group from a closet proximity with the Germanic group (in the Longobardi's tree) to a more distance linkage with them in our case.

Our classification compares with the one obtained by Longobardi's school with these data, cf. comments in the Conclusion, where we argue why our distance is quite promising for the new and ambitious data Longobardi's school are now providing. Actually, our distance compares rather well also with the classification obtained by Q. D. Atkinson and R. D. Gray, cf. (Gray and Atkinson, 2003; Bouckaert et al., 2012).

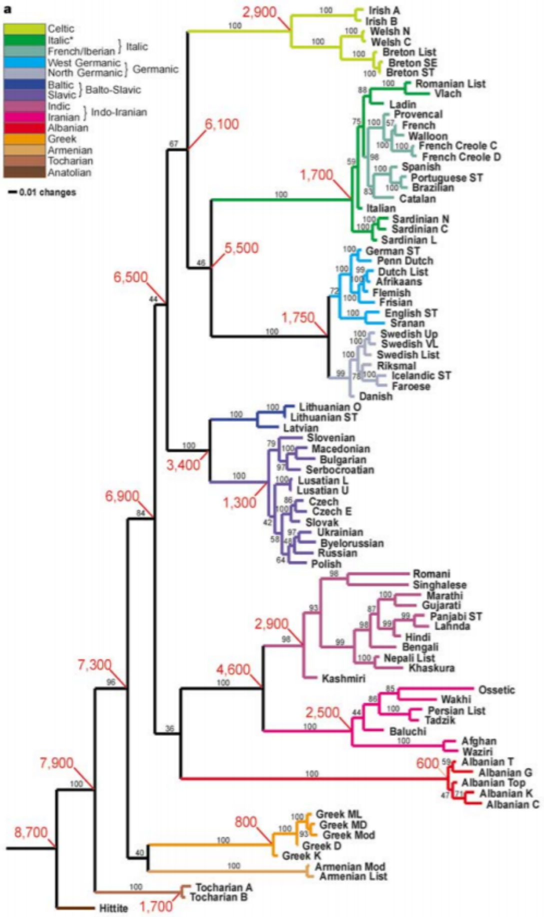


Figure 3: Q. D. Atkinson and R. D. Gray classification tree, cf. (Gray and Atkinson, 2003; Bouckaert et al., 2012)

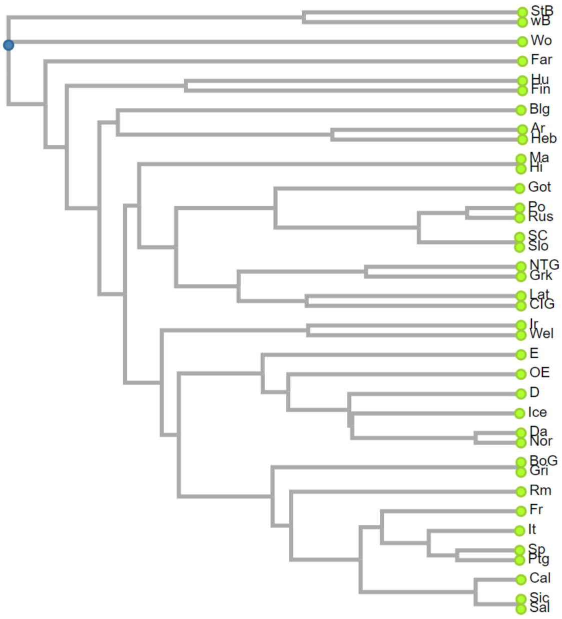


Figure 4: Classification obtained with the generalized Steinhaus transform applied to the taxi-cab distance with Longobardi's data

First of all for the classification we have used Longobardi's dataset, while Atkinson and Gray have used their own dataset. If we look to Marathi and Hindi we can notice that they are grouped together in both trees; also Polish, Russian, Serbo Croatian and Slovenian are grouped together in both trees; the same is for New Testament Greek, Greek and Classical Attic Greek. Also the Celtic languages (i.e. Gaelic and Welsh) and Germanic languages are grouped together. Our misclassification of Bulgarian is not that worring, since Longobardi covers only the syntax of the noun, and the Bulgarian noun is well-known to behave in quite a non-Slavic way, due possibly to its Balcanian substratum.

5 Conclusions

In this paper we have investigated the language classification problem by using original tools inspired by fuzzy logic. In the literature fuzzy tools and information theory have been used only quite sparsely. We have exhibited a metric distance which allows one to deal jointly with both irrelevance and inconsistency, and which is based on the generalized Steinhaus transform. Our classification compares quite well both with the one obtained by Longobardi and the one obtained by Atkinson and Gray. The merits of our metric proposal should not be underestimated, as we now comment. In more recent datasets, Longobardi and his school introduce families and macrofamilies which are quite apart. Now, think of two languages L and Λ such that the following occurs (and this does occur with "remote" languages): in most position i at least one of the two languages has a star signalling non-definition of the corresponding features. Since such positions are totally ignored by Longobardi's non-metric distance, the value obtained for the distance relies on a handful of positions only, and it is no surprise that the two languages end up being poorly classified, a sourse of worry, indeed. Now, our metric distances are not that drastic, and so might be used as a sort of companion to Longobardi's non-metric distances, useful when the latter have a low significance due to the fact that only few features "survive".
We are confident that the fuzzy ideas and methods discussed in this paper and in (Franzoi and Sgarro, 2017a; Franzoi, 2017; Dinu et al., 2018) will prove to be useful not only in linguistic classification and linguistic phylogeny, but also outside

linguistic, first of all in coding theory cf. (Franzoi and Sgarro, 2017a), or even in bioinformatics. Irrelevance and inconsistency appear to be features which are dealt with quite sparsely, if ever, outside Longobardi's school; actually, these flexible features might prove to be quite useful not only in linguistic classification phylogeny, cf. (Franzoi and Sgarro, 2017a,b), but also in the investigation of the history of texts. So far, we are just providing technical tools to be used in Longobardi's research, which, in its turn, is methodically matched with the *current state of the art*, cf. (Bortolussi et al., 2011; Longobardi et al., 2016, 2013, 2015; Longobardi, 2017; Kazakov et al., 2017).

Table 1: Longobardi original data

ft.	Sic	Cal	It	Sal	Sp	Fr	Ptg	Rm	Lat	ClG	NtG	BoG	Gri	Grk	Got	OE	E	D	Da	Ice	Nor
1.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6.	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7.	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
8.	1	1	1	1	1	1	1	1	*	1	1	1	1	1	0	1	1	1	1	1	1
9.	1	1	1	1	1	1	1	1	*	1	1	1	1	1	*	0	0	0	0	0	0
10.	0	0	0	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11.	0	0	0	0	0	0	0	0	*	0	0	0	0	0	*	0	0	0	0	0	0
12.	0	0	0	0	0	0	0	1	*	0	0	0	0	0	*	0	0	0	1	1	1
13.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	1	1
14.	0	0	0	0	0	0	0	0	*	0	0	0	0	0	*	0	0	0	0	0	0
15.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18.	1	1	1	1	1	*	1	1	*	0	0	1	1	1	*	0	1	1	1	0	1
19.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
20.	0	0	0	0	1	0	1	*	*	1	1	0	0	0	*	*	*	*	*	*	*
21.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
23.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
24.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1	.*	1	1	1	1
25.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
27.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
28.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29.	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	1	1	1	1	1
30.	0	0	0	0	0	0	0	*	1	1	*	*	*	*	*	*	0	0	0	0	0
31.	1	1	1	1	1	1	1	*	1	1	*	*	*	*	*	*	1	1	1	1	1
32.	0	1	0	0	0	0	0	0	*	*	1	1	1	1	1	1	0	1	0	1	0
33.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34.	0	0	0	0	1	1	*	0	*	0	0	0	0	0	*	*	*	*	*	*	*
35.	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	*	1	1	1	1
36.	0	0	0	0	0	0	0	*	0	0	*	*	*	*	*	*	*	0	0	0	0
37.	0	0	0	0	0	0	0	0	0	0	0	*	*	*	0	0	0	0	0	0	0
38.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39.	*	*	*	*	*	*	*	*	1	1	1	*	*	*	1	1	*	*	*	1	*
40.	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	0
41.	1	1	1	1	1	1	1	1	*	0	0	1	1	0	*	1	1	1	1	*	1
42.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
43.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44.	1	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
45.	*	*	1	*	1	1	1	1	0	0	0	*	*	0	0	0	0	0	0	0	0
46.	*	*	*	*	*	*	*	*	0	0	0	*	*	0	0	0	0	0	0	0	0
47.	*	*	*	*	*	*	*	*	0	0	1	*	*	1	1	1	*	1	*	1	*
48.	*	*	*	*	*	*	*	*	1	1	*	*	*	*	*	*	0	*	0	*	0
49.	*	*	*	*	*	*	*	*	1	1	1	*	*	1	1	0	0	0	0	0	0
50.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0
51.	0	0	0	0	0	0	0	0	*	1	1	0	0	1	1	*	0	*	*	*	*
52.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*
53.	*	*	1	*	1	1	1	1	0	0	0	*	*	0	*	1	1	1	1	1	1

Table 2: Longobardi original data

ft.	Blg	SC	Slo	Po	Rus	Ir	Wel	Far	Ma	Hi	Ar	Heb	Hu	Fin	StB	wB	Wo
1.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3.	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	1	1
4.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
5.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
6.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	*	*	*
7.	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	1
8.	1	*	*	*	*	1	1	*	*	*	1	1	1	*	*	*	1
9.	1	*	*	*	*	0	0	*	*	*	1	1	1	*	*	*	*
10.	0	0	0	0	0	0	0	0	0	0	0	0	0	1	*	*	*
11.	0	*	*	*	*	0	0	*	*	*	0	0	0	*	0	1	1
12.	1	*	*	*	*	0	0	*	*	*	0	0	0	*	*	*	0
13.	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
14.	0	*	*	*	*	0	0	*	*	*	1	0	0	*	*	*	1
15.	0	0	0	0	0	0	0	0	0	0	1	0	0	0	*	*	*
16.	1	1	1	1	1	1	0	0	1	1	1	1	0	0	*	*	*
17.	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	1
18.	0	*	*	*	*	0	0	*	*	*	0	0	*	*	*	*	*
19.	*	*	*	*	*	*	*	1	0	0	*	*	0	*	*	*	0
20.	*	*	*	*	*	*	*	*	*	*	0	0	0	*	1	1	*
21.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
22.	1	1	1	1	1	1	0	0	1	1	1	1	0	1	*	*	*
23.	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
24.	*	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*	*
25.	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1
26.	*	*	*	*	*	*	*	*	1	1	*	*	*	*	0	0	*
27.	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	*
28.	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	*
29.	1	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1	0
30.	0	*	*	*	*	0	0	0	0	0	0	0	*	*	0	0	*
31.	1	*	*	*	*	1	1	1	*	*	1	1	*	*	*	*	*
32.	0	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	1
33.	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
34.	0	*	*	*	*	1	1	*	*	*	0	0	0	*	*	*	0
35.	1	1	1	1	1	0	*	*	1	1	0	0	*	0	*	*	0
36.	0	*	*	*	*	0	*	*	1	1	*	*	*	*	*	*	*
37.	1	1	1	0	1	*	0	0	0	0	*	*	*	*	0	0	*
38.	0	0	0	0	0	0	0	0	0	0	1	1	*	*	0	0	*
39.	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	0	1
40.	0	0	0	0	0	*	*	0	0	0	1	*	0	0	0	*	0
41.	1	*	*	*	*	*	*	*	*	*	0	*	0	*	1	*	1
42.	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
43.	0	0	0	0	0	0	0	1	0	0	*	*	0	0	0	0	0
44.	0	0	0	0	0	1	1	*	0	0	*	*	0	0	*	*	*
45.	0	0	0	0	0	*	*	*	0	0	*	*	0	0	*	*	*
46.	0	0	0	0	0	*	*	*	0	0	*	*	0	0	*	*	*
47.	*	1	1	1	1	*	*	*	*	*	*	*	*	*	*	*	1
48.	1	*	*	*	*	*	*	*	1	1	*	*	1	1	*	*	*
49.	0	0	0	0	0	*	*	*	0	0	*	*	0	0	*	*	*
50.	0	0	0	1	1	*	*	*	0	0	*	*	0	0	*	*	*
51.	*	*	*	*	*	0	0	*	*	*	1	1	*	*	*	*	0
52.	*	*	*	0	0	0	0	1	*	*	0	0	*	*	0	0	1
53.	0	0	0	0	0	*	*	0	*	*	*	*	1	1	1	1	*

References

2018. Ethnologue. https://www.ethnologue.com/.

L. Bortolussi, A. Sgarro, G. Longobardi, and C. Guardiano. 2011. How many possible languages are there? *Biology, Computation and Linguistics* pages 168–179.

R. Bouckaert, P. Lemey, M. Dunn, S. J. Greenhill, A. V. Alekseyenko, A. J. Drummond, R. D. Gray, M. A. Suchard, and Q. D. Atkinson. 2012. Mapping the origins and expansion of the indo-european language family. *Science* 337(6097):957–960.

A. M. Ciobanu, L. P. Dinu, and A.Sgarro. 2018. Towards a map of the syntactic similarity of languages. In *CICLing 2017*. volume LNCS 10761, pages 1–15.

M. M. Deza and E. Deza. 2009. *Encyclopedia of Distances*. Springer Dordrecht Heidelberg, London New York.

A. Dinu, L. P. Dinu, L. Franzoi, and A. Sgarro. 2018. Steinhaus transforms of fuzzy string distances in computational linguistics. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations - 17th International Conference, IPMU 2018, Cádiz, Spain, June 11-15, 2018*. volume Proceedings, Part I, pages 171–182.

D. Dubois, H. T. Nguyen, and H. Prade. 2000. Possibility theory, probability and fuzzy sets: Misunderstanding, bridges and gaps. In *Fundamentals of Fuzzy Sets*. Kluwer Academic Publishers, pages 343–438.

L. Franzoi. 2017. Jaccard-like fuzzy distances for computational linguistics. In *19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timişoara, Romania, September 21-24, 2017*. pages 196–202.

L. Franzoi and A. Sgarro. 2017a. Fuzzy hamming distinguishability. In *2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2017, Naples, Italy, July 9-12, 2017*. pages 1–6.

L. Franzoi and A. Sgarro. 2017b. Linguistic classification: T-norms, fuzzy distances and fuzzy distinguishabilities. In *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference KES-2017, Marseille, France,
6-8 September 2017.*. pages 1168–1177.

R. D. Gray and Q. D. Atkinson. 2003. Language-tree divergence times support the anatolian theory of indo-european origin. *Nature* 426:435–439.

D. Kazakov, G. Cordoni, A. Ceolin, M. A. Irimia, S. Kim, D. Michelioudakis, N. Radkevich, C. Guardiano, and G. Longobardi. 2017. Machine learning models of universal grammar parameter dependencies. *Proceedings of Knowledge Resources for the Socio-Economic Sciences and Humanities associated with RANLP-17* pages 31–37.

G. Longobardi. 2017. Principles, parameters, and schemata: A radically underspecified ug. *Linguistic Analysis* 41(3–4):517–557.

G. Longobardi, A. Ceolin, L. Bortolussi, C. Guardiano, M. A. Irimia, D. Michelioudakis, N. Radkevich, and A. Sgarro. 2016. Mathematical modeling of grammatical diversity supports the historical reality of formal syntax. *University of Tübingen, online publication system Tübingen DEU* pages 1–4.

G. Longobardi, S. Ghirotto, C. Guardiano, F. Tassi, A. Benazzo, A. Ceolin, and G. Barbujan. 2015. Across language families: Genome diversity mirrors language variation within europe. *American Journal of Phisical Anthropology* 157:630–640.

G. Longobardi, C. Guardiano, G. Silvestri, A. Boattini, and A. Ceolin. 2013. Toward a syntactic phylogeny of modern indo-european languages. *Journal of Historical Linguistics* 3:11:122–152.

Ž. Muljačić. 1967. Die Klassifikation der romanischen Sprachen. *Rom. Jahrbuch 18* pages 23–37.

A. Sgarro. 1977. A fuzzy hamming distance. *Bullettin Math. de la Soc. Sci. Math. de la R. S. de Roumanie* 69(1-2):137–144.

L. A. Zadeh. 1965. Fuzzy sets. *Information and Control* 8(3):338–353.

Corpus Lexicography in a Wider Context

Chen Gafni
Bar-Ilan University
chen.gafni@gmail.com

Abstract

This paper describes a set of tools that offers comprehensive solutions for corpus lexicography. The tools perform a range of tasks, including construction of corpus lexicon, integrating information from external dictionaries, internal analysis of the lexicon, and lexical analysis of the corpus. The set of tools is particularly useful for creating dictionaries for under-resourced languages. The tools are integrated in a general-purpose software that includes additional tools for various research tasks, such as linguistic development analysis. Equipped with a user-friendly interface, the described system can be easily incorporated in research in a variety of fields.

1 Introduction

Corpus lexicography, a key component in modern dictionary compilation, has become increasingly powerful and efficient due to the development of various tools such as Word Sketch (Kilgarriff and Tugwell, 2002) and TickBox Lexicography (Kilgarriff et al., 2010). While corpus lexicography deserves further development in its own right, it is worthwhile considering it as an integral part of wider scientific missions. This paper describes several tools for corpus lexicography, whose design takes into consideration their contribution to linguistic research. The tools are integrated in a general-purpose linguistic software, where they can be readily applied in language acquisition studies, psycholinguistics, and other fields of research.

2 Preliminaries

The described system is implemented in the *Child Phonology Analyzer* software (CPA; Gafni, 2015)[1], which was built in MS Excel due to its popularity and user-friendly interface.[2] Nevertheless, the concepts behind the system are general and can be implemented in various environments. The software can analyze corpora stored in various file formats, including Excel and plain-text files, as well as several special formats used in linguistic research: Praat's TextGrids, CHAT transcription files, EAF annotation files, and XML schema for TalkBank data. The software converts analyzed corpora into Excel format and adds all analysis products to the Excel file.

2.1 Organizing the Data

The described tools ("macros") require that the corpus text be stored in a vector format. The text can be converted to a vector format using CPA's "Data preparation" macro with the "Corpus tokenization" option, which segments the text into words.

Segmentation is performed on the basis of blank spaces and additional word-dividing characters, which can be defined in CPA's "Word dividers" table (Figure 1). There are two types of word dividers, which can be used for separating words even at the absence of a blank space: *punctuation* marks (e.g., comma) are deleted during segmentation, while *final letters* are not (final letters are special letter forms appearing only at word endings. See some examples from Hebrew in Figure 1).

[1] Website: https://chengafni.wordpress.com/cpa/

[2] The code is written in Visual Basic for Applications for MS Excel and is available under the GNU General Public License. A version of this system for LibreOffice Calc is planned to appear in the future in order to free it from dependency on proprietary software.

2.2 Longitudinal and Multi-Level Corpora

The system was designed especially for analyzing longitudinal data from language acquisition studies. Such corpora typically list utterances made by a child alongside the hypothesized intended target utterances (a corpus containing such paired utterances is a *multi-level* corpus). The age of the child is also recorded for each utterance for the purpose of developmental analysis (Figure 2). Corpus tokenization (see above) processes all the above-mentioned information. This further allows analyzing both the input (target) and output (production) lexicons of the child from a developmental perspective.

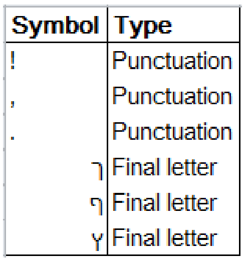
2.3 Multi-Speaker Corpora

CPA can also handle corpora that contain data from multiple sources ("speakers"), stored in a single or multiple spreadsheets. The sources can be different texts, cross-sectional data from several children (Figure 3), parallel data from elicitation experiments, etc. During corpus tokenization, the data from each speaker is labelled accordingly. This allows building a separate lexicon for each speaker for the purpose of comparative analysis.

3 Constructing Corpus Lexicon

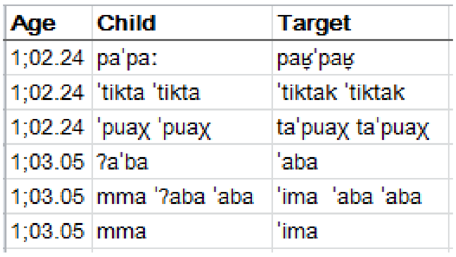
The "Construct lexicon" macro takes as input a corpus in vector format and lists the different item types in the vector including the number of occurrences of each item ("Count"). For multi-level corpora, this analysis can be done separately for target and output levels. For multi-speaker corpora, the macro constructs separate lexicons for each speaker. In addition, the macro constructs a general lexicon based on the lexicons of individual speakers (this is done separately for target and output levels). For developmental data, the macro also records the age in which the item is first attempted (for target lexicon) or produced (for output lexicon) and the spreadsheet row index containing the first attempt (Figure 4).

For items containing explicit morphological boundaries (e.g., # in *ha#kelev* 'the.dog' (Hebrew)), the macro creates a list of potential affixes based on word fragments separated by morpheme boundary markers (naturally, the initial list contains both true grammatical affixes and lexical stems). The list of affixes can be used later for analyzing the properties of polymorphemic words in the lexicon.



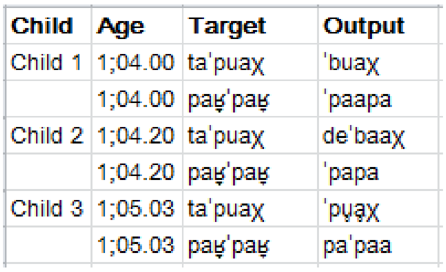
Symbol	Type
!	Punctuation
,	Punctuation
.	Punctuation
ך	Final letter
ף	Final letter
ץ	Final letter

Figure 1: Word dividing symbols



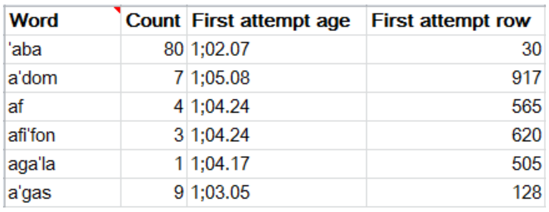
Age	Child	Target
1;02.24	paˈpaː	paʁ̞ˈpaʁ̞
1;02.24	ˈtikta ˈtikta	ˈtiktak ˈtiktak
1;02.24	ˈpuaχ ˈpuaχ	taˈpuaχ taˈpuaχ
1;03.05	ʔaˈba	ˈaba
1;03.05	mma ˈʔaba ˈaba	ˈima ˈaba ˈaba
1;03.05	mma	ˈima

Figure 2: Longitudinal language acquisition data from a Hebrew-speaking child



Child	Age	Target	Output
Child 1	1;04.00	taˈpuaχ	ˈbuaχ
	1;04.00	paʁ̞ˈpaʁ̞	ˈpaapa
Child 2	1;04.20	taˈpuaχ	deˈbaaχ
	1;04.20	paʁ̞ˈpaʁ̞	ˈpapa
Child 3	1;05.03	taˈpuaχ	ˈpu̯a̯χ
	1;05.03	paʁ̞ˈpaʁ̞	paˈpaa

Figure 3: A corpus of language acquisition cross-sectional study. Data from three Hebrew-speaking children



Word	Count	First attempt age	First attempt row
'aba	80	1;02.07	30
a'dom	7	1;05.08	917
af	4	1;04.24	565
afi'fon	3	1;04.24	620
aga'la	1	1;04.17	505
a'gas	9	1;03.05	128

Figure 4: A lexicon for a developmental corpus

4 Importing Lexical Information

The corpus lexicon can be turned into a dictionary by adding information describing the various items. The descriptive information is stored in separate columns in the lexicon spreadsheet. Each such column represents some lexical property (e.g., part-of-speech, grammatical gender). In the absence of external dictionaries, the lexical information needs to be added manually. However, if there is an available resource for the particular language, the “Import dictionary” macro can import the lexical information from the external resource.

This macro receives as input the corpus lexicon and an external dictionary in a table format. The macro copies information from the external dictionary to the lexicon for every lexicon entry found in the dictionary. The macro can be used for importing information about lexical words (Figure 5), as well as grammatical affixes (Figure 6).

The macro was specially designed to handle lexical entries containing explicit morphological boundaries. For such lexical entries (e.g., *le#ˈkof* ‘to.monkey’ (Hebrew)), the macro first searches the full entry in the words dictionary. If not found, the macro then searches the individual morphemes. If a morpheme is found in the dictionary (e.g., *kof*), the macro imports the information for that entry to the corpus lexicon (Figure 7).

When importing information from an affix dictionary, the macro can use the external list of affixes to remove irrelevant entries from the corpus affixes lexicon (i.e., stems included in the affixes lexicon during its construction).

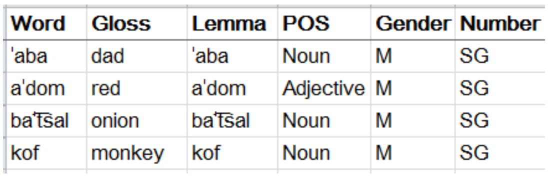
An affixes dictionary is constructed in a similar way to a words dictionary; it contains a list of affixes with additional columns providing information about these affixes. However, the additional fields have a functional role: they specify how the affix modifies the properties of affixed words. This information can be used for modifying polymorphemic entries in the lexicon (see 5.1).

5 Analyzing the Lexicon

5.1 Morphological Analysis

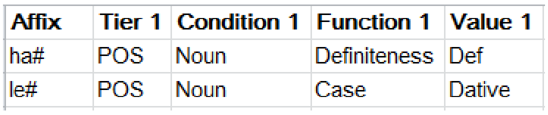
If the corpus lexicon contains polymorphemic words with explicit marking of morphological boundaries, and an affixes dictionary is available (see 4), the “Morphological analysis” macro can import information from the affixes dictionary to the corpus lexicon.

Each entry in the affixes dictionary should have the following fields (Figure 6): (a) Tier: a name of a field in the words lexicon. For example, a “POS” value in the tier field (stands for “Part-of-speech”) indicates that the affix applies to lexical items in a specific lexical category. (b) Condition: a possible value of the lexical field specified in the tier field. For example, a condition value “Noun” indicates that the affix applies to nouns. (c) Function: the name of the lexical field modified by the affix. For example, a value of “Definiteness” indicates that the affix specifies the definiteness value of the hosting word. (d) Value: the value assigned to the lexical field specified in the “Function” field. For example, a value of “Def” indicates that the affix marks the hosting word as being definite.



Word	Gloss	Lemma	POS	Gender	Number
ˈaba	dad	ˈaba	Noun	M	SG
aˈdom	red	aˈdom	Adjective	M	SG
baˈt͡sal	onion	baˈt͡sal	Noun	M	SG
kof	monkey	kof	Noun	M	SG

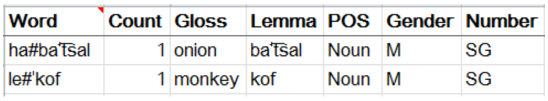
Figure 5: An external words dictionary



Affix	Tier 1	Condition 1	Function 1	Value 1
ha#	POS	Noun	Definiteness	Def
le#	POS	Noun	Case	Dative

Figure 6: An external affixes dictionary

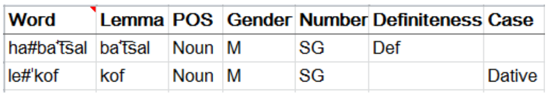
For each affix in the affixes dictionary, it is possible to define multiple feature quadruplets (e.g., “Tier 1”, “Condition 1”, …, “Tier 2”, “Condition 2”, etc.). This option is useful for handling affixes that can affect multiple word classes (e.g., nouns and adjectives) or have multiple functions (e.g., express possession and mark tense).



Word	Count	Gloss	Lemma	POS	Gender	Number
ha#baˈt͡sal	1	onion	baˈt͡sal	Noun	M	SG
le#ˈkof	1	monkey	kof	Noun	M	SG

Figure 7: Imported lexical information based on the stems of prefixed words

The “Morphological analysis” macro finds lexical entries containing affixes and modifies their properties according to the details of the affix. If an affix modifies a lexical field not defined in the lexicon, the macro adds that field to the lexicon (Figure 8).

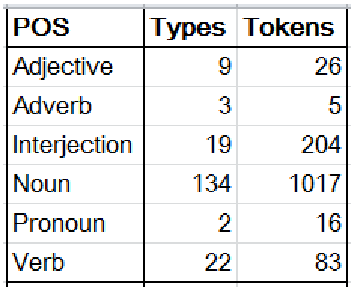


Word	Lemma	POS	Gender	Number	Definiteness	Case
ha#baˈt͡sal	baˈt͡sal	Noun	M	SG	Def	
le#ˈkof	kof	Noun	M	SG		Dative

Figure 8: Lexical entries of prefixed words after morphological analysis

5.2 Lexicon Summary

This macro generates a summary table of the lexicon. The summary table includes a list for each lexical field (e.g., “POS”) that specifies the various values of the field (e.g., “Noun”, “Verb”). For each value, the list indicates the number of corpus tokens and types. The number of types is the number of items in the lexicon with the relevant value (e.g., the number of noun types), and the number of corpus tokens is calculated from the “Count” field in the lexicon (Figure 9).



POS	Types	Tokens
Adjective	9	26
Adverb	3	5
Interjection	19	204
Noun	134	1017
Pronoun	2	16
Verb	22	83

Figure 9: Lexicon summary by part-of-speech

6 Integrating Lexicons

Efficient integration of information is essential for compiling a dictionary based on data from multiple resources. The “Merge worksheets” macro is a general utility macro that integrates the contents of multiple spreadsheets in a file. Thus, it requires lexicons generated from different corpora to be stored in one file (this can be done either manually or automatically with the “Merge workbooks” CPA macro).

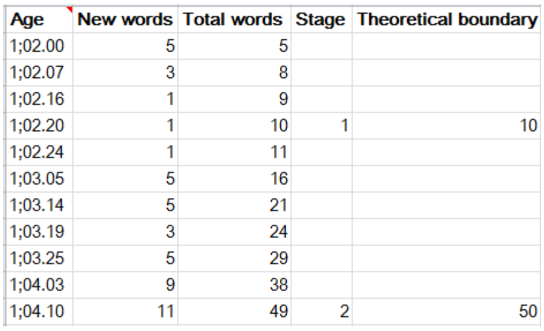
The “Merge worksheets” macro has several operation modes, one of which is designed specifically to integrate lexicon tables. The macro receives as input any number of spreadsheets. It creates a single lexicon[3] containing information from all input lexicons. The merged lexicon contains the union of lexical fields in all input lexicons (i.e., a lexical field will be included in the merged lexicon if it appears at least in one input lexicon).

The entries in the merged lexicon are sorted alphabetically. If a lexical entry appears in multiple input lexicons, the duplicate entries are merged. The merged entry summarizes token counts from the contributing corpora (e.g., if an item appears 10 times in one corpus and 20 times in another corpus, the merged lexicon will record 30 tokens for that item). In addition, the merged entry will contain the lexical properties collected from all contributing entries. In case of conflicting inputs (e.g., an item is classified as a noun in one lexicon and as a verb in another), the merged entry will indicate all possible values for that property (e.g., Noun / Verb). The merging macro can also add labels indicating the source(s) (i.e., the name of the input lexicon) of each entry.

7 Lexical Development

Assessing the size of the child’s lexicon is an important part of longitudinal language acquisition studies, from both theoretical and clinical perspectives. In particular, there is evidence that aspects of grammatical development are tightly correlated with vocabulary size (Bates and Goodman, 1997).

The “Lexical development” macro analyzes lexical growth in corpora that record the age of production of every utterance. Using the age of first attempt to produce target words (see 3), the macro divides the child’s lexicon into stages of lexical development (Figure 10). The first stage is marked by the acquisition of the first 10 words, the second by a total lexicon size of 50 words, and then an additional 50 words for every subsequent stage (Adam and Bat-El, 2009).

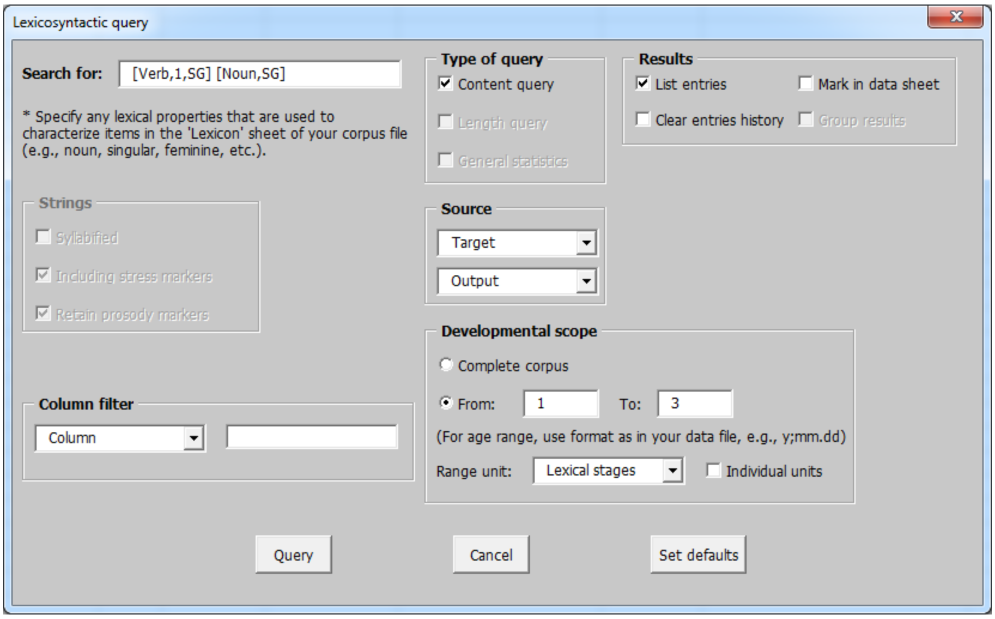


Age	New words	Total words	Stage	Theoretical boundary
1;02.00	5	5		
1;02.07	3	8		
1;02.16	1	9		
1;02.20	1	10	1	10
1;02.24	1	11		
1;03.05	5	16		
1;03.14	5	21		
1;03.19	3	24		
1;03.25	5	29		
1;04.03	9	38		
1;04.10	11	49	2	50

Figure 10: Lexical development

Stages of lexical development are aligned with recording sessions, such that if a theoretical stage boundary is reached in mid-session, the actual boundary will be assigned either to that session or to the preceding session (whichever is closer). For

[3] Due to software limitations, the maximal number of entries in a single lexicon (or a lexicon generated by merging a number of lexicons) is 1,048,575. When this limit is exceeded, CPA splits the lexicon over multiple spreadsheets.



Lexicosyntactic query

Search for: [Verb,1,SG] [Noun,SG]

* Specify any lexical properties that are used to characterize items in the 'Lexicon' sheet of your corpus file (e.g., noun, singular, feminine, etc.).

Type of query
- ☑ Content query
- □ Length query
- □ General statistics

Results
- ☑ List entries
- □ Mark in data sheet
- □ Clear entries history
- □ Group results

Strings
- □ Syllabified
- ☑ Including stress markers
- ☑ Retain prosody markers

Source
- Target
- Output

Developmental scope
- ○ Complete corpus
- ● From: 1 To: 3

(For age range, use format as in your data file, e.g., y;mm.dd)

Range unit: Lexical stages □ Individual units

Column filter
- Column

Figure 11: Lexicosyntactic query form

example, if the child has reached 49 cumulative target types at the end of session 1 and 58 cumulative target types at the end of session 2, the theoretical landmark of 50 words will be assigned to session 1. If the lexicon grows rapidly, such that more than one theoretical stage is passed in a single session, the macro will “skip” intermediate stages and assign only the last stage to that session. For example, if the size of the lexicon jumps from 100 words (stage 3) to 200 words (stage 5) in a single session, that session will be marked as the end point of stage 5, skipping stage 4. The macro also provides a more fine-grained account, indicating the number of new words added to the lexicon in every session and the total lexicon size after every session.

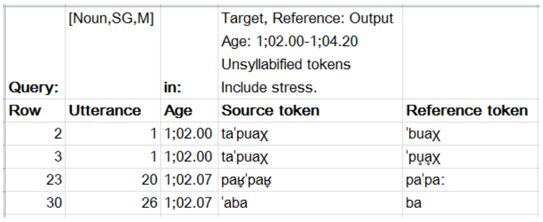
By default, lexical development is calculated based on the full list of lexical entries. However, this list is organized by word form (types), such that words that are interrelated via inflectional morphology (e.g., *cat–cats*) are listed as separate entries. Relying on plain surface forms can result in over-estimation of lexicon size. This can be avoided by analyzing lexical development by lemma/lexeme. When this option is chosen, the macro analyzes the lemma field of the lexicon rather than the word field. The lemma field indicates the lemma of each lexical entry (e.g., the lemma of *cat* and *cats* is *cat*). The lemma information can be supplied manually or imported from an external dictionary (see 4). When a lexical entry has no lemma specified, the surface form of the entry will be taken as the lemma.

8 Lexical Queries

Once lexical properties are specified in the lexicon, this information can be used to analyze the corpus. CPA has a set of macros that can extract linguistic information from the corpus on various levels of analysis, via a user-friendly query form (Figure 11). One of these macros, “Lexicosyntactic query”, queries the corpus at the word and utterance levels. Specifically, “Content Lexicosyntactic queries” can find occurrences of lexical properties and sequences of lexical properties in the corpus. For example, the query [Verb] [Noun] will find all instances of verbs followed by nouns in the corpus. Similarly, the query [Verb,1,SG] [Noun,SG] will find all instances of verbs conjugated in the first person singular followed by singular nouns.

The scope of queries can be constrained by age or stage of lexical development. Thus, for example, it is possible to get all verbs attempted by a child at a given age/lexical stage or range of ages/lexical stages. This option allows for investigation of lexical development at a more fine-grained level.

Queries over single-item sequences (e.g., [Verb,SG]) calculate the number of tokens and types and can also return a list of items that matched the query (Figure 12). Queries over multi-item sequences (e.g., [Verb] [Noun]) do not return



Query:	[Noun,SG,M]	in:	Target, Reference: Output Age: 1;02.00-1;04.20 Unsyllabified tokens Include stress.	
Row	**Utterance**	**Age**	**Source token**	**Reference token**
2	1	1;02.00	taˈpuaχ	ˈbuaχ
3	1	1;02.00	taˈpuaχ	ˈpu̥ḁχ
23	20	1;02.07	paʁ̥ˈpaʁ̥	paˈpaː
30	26	1;02.07	ˈaba	ba

Figure 12: Corpus instances of singular masculine nouns (source) paired with the corresponding forms produced by an infant (reference).

specific items, but rather a list of indices of rows in the corpus where such sequences are found.

In addition to lexicosyntactic queries, CPA has similar query macros for analyzing the phonological properties of corpora. These queries, too, can be correlated with lexical development. Additional, more advanced macros can be used to combine queries on different levels of analysis. This allows, for example, to study the interaction between phonological and lexical development.

9 Discussion

The described set of tools can help creating resources for under-resourced languages. For example, it was used for creating a lexicon with corpus frequency data for Hebrew (Gafni, 2019), and it is currently being used in an ongoing longitudinal study of phonological development in twins. In addition to building a lexicon for each participating child and assessing lexical development, the system can assist in improving the quality of the transcribed data.

Given that the transcribed data can contain many errors (typos, misperceptions), it is important to have it validated. Since the amount of transcribed data can be enormous (tens of thousands of tokens) and the transcription task is very time-consuming, it is impractical to have every token transcribed by multiple transcribers. One possibility to check data quality is to have a random subset of the corpus (e.g., 10% of the tokens) be transcribed by more than one transcriber, and calculate inter-transcriber reliability. However, such an approach can help detecting problems in a limited part of the corpus. The tools described in this paper offer a more systematic approach to quality check of transcribed data. In this lexicon-based approach, one goes over the entries in the automatically generated corpus lexicon and looks for suspicious entries. For the lexicon of target words, this mainly involves looking for non-existing words, which likely resulted from typos. For the lexicon of produced forms (output lexicon), quality check mainly involves examining tokens with unusual structure that deviates from the phonology of the ambient language. Thus, in the proposed approach, one estimates the potential of lexical entries to contain errors, and then focuses on suspicious forms. This is more effective than examining corpus subsets randomly.

The described tools can be integrated in any task involving corpus analysis. For example, the CPA software includes an n-gram frequency calculator, which can calculate corpus-weighted mean n-gram frequencies over a list of strings (in this context, n-gram refers to a sequence of letters or phones within words). This is useful for creating controlled sets of stimuli for psycholinguistic experiments.

Finally, it should be acknowledged that there is some overlap between the described system and other existing systems. Well-established systems such as Sketch Engine (Kilgarriff et al., 2014) provide powerful solutions for corpus lexicography, and the CLAN program (MacWhinney, 2000) can be used for studying lexical development.

Compared to these programs, CPA is currently limited in areas such as collocation analysis and POS tagging. On the other hand, CPA has some advantages over these programs. Its unique built-in lexical development tool allows for more comprehensive study of language development, and its querying system allows for combined lexical and phonological corpus analysis. The user-friendly interface enhances user experience and saves the need to learn complex query syntax, as used by the CLAN program. In addition, CPA is distributed as an Excel file. This means that Excel users can perform the various analysis tasks in the natural environment of the data, without the need to install (or purchase) additional software.

To conclude, this paper views corpus lexicography in a wide context of linguistic research. Accordingly, the described tools are integrated in a single, user-friendly system designed to support any task requiring corpus analysis. Future improvements to the current system will include the addition of standard lexicographic functions, such as collocation analysis and morphological analysis that does not require overt marking.

References

Galit Adam and Outi Bat-El. 2009. When Do Universal Preferences Emerge in Language Development? the Acquisition of Hebrew Stress. Brill's Journal of Afroasiatic Languages and Linguistics, 1(1):255–282.

Elizabeth Bates and Judith C. Goodman. 1997. On the Inseparability of Grammar and the Lexicon: Evidence from Acquisition, Aphasia and Real-time Processing. Language and Cognitive Processes, 12(5–6):507–584.

Chen Gafni. 2015. Child Phonology Analyzer: processing and analyzing transcribed speech. In The Scottish Consortium for ICPhS 2015, editor, Proceedings of the 18th International Congress of Phonetic Sciences., pages 1–5, paper number 531, Glasgow, UK: the University of Glasgow.

Chen Gafni. 2019. General Lexicons of Hebrew: Resources for Linguistic and Psycholinguistic Research (version 1.0).

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. The Sketch Engine: ten years on. Lexicography, 1(1):7–36.

Adam Kilgarriff, Vojtěch Kovář, and Pavel Rychlý. 2010. Tickbox lexicography. In Sylviane Granger and Magli Paquot, editors, eLexicography in the 21st century: new challenges, new applications. Proceedings of eLex 2009, volume 7, pages 411–418, Louvain-La-Neuve. UCL Presses Universitaires De Louvain.

Adam Kilgarriff and David Tugwell. 2002. Sketching words. In Marie-Hélene Corréard, editor, Lexicography and Natural Language Processing: A Festschrift in Honour of B. T. S. Atkins, pages 125–137. Euralex.

Brian MacWhinney. 2000. The CHILDES Project: Tools for Analyzing Talk. Lawrence Erlbaum Associates Inc, Mahwah, NJ, 3rd edition.

A Universal System for Automatic Text-to-Phonetics Conversion

Chen Gafni
Bar-Ilan University
chen.gafni@gmail.com

Abstract

This paper describes an automatic text-to-phonetics conversion system. The system was constructed to primarily serve as a research tool. It is implemented in a general-purpose linguistic software, which allows it to be incorporated in a multifaceted linguistic research in essentially any language. The system currently relies on two mechanisms to generate phonetic transcriptions from texts: (i) importing ready-made phonetic word forms from external dictionaries, and (ii) automatic generation of phonetic word forms based on a set of deterministic linguistic rules. The current paper describes the proposed system and its potential application to linguistic research.

1 Introduction

There are currently many commercial and academic works dealing with automatic conversion of text to phonetics (G2P). Existing solutions are based on some combination of language-specific phonetic dictionaries, corpus-based statistical models (e.g., Hidden Markov Models), deterministic models based on linguistic rules, and machine learning techniques (see review in Tomer, 2012). Importantly, most available tools are closed-source, support only a limited number of languages, and are often available only for commercial use (e.g., Baytukalov, 2019).

The current paper describes a new, open-source system that generates phonetic transcriptions from texts. Its design allows it to apply, in principle, to any language. At present, the system relies on two mechanisms to generate the transcriptions: (i) importing ready-made phonetic word forms from external dictionaries, and (ii) automatic generation of phonetic word forms based on a set of deterministic linguistic rules. The following sections describe the transcription mechanisms and additional relevant tools.

2 Preliminaries

The described system is implemented in the *Child Phonology Analyzer* software (CPA; Gafni, 2015)[1], which was built in MS Excel due to its popularity and user-friendly interface.[2] Nevertheless, the concepts behind the system are general and can be implemented in various environments. The following subsections describe the general organization of the system and guidelines for working with the data.

2.1 Tables

The system uses a set of tables of definitions and rules to guide its operation. The tables are stored in separate spreadsheets in the CPA file and can be edited according to the properties of the language in question. Moreover, variants of these tables can be stored in separate files and imported by the system when needed. This feature allows users to maintain sets of definitions and rules for multiple languages.[3]

2.2 Organizing the Data

The transcription procedures ("macros") operate on a vector of words. Thus, the input text should be converted into a vector format prior to running the transcription macros. This can be done using the "Corpus tokenization" option of CPA's "Data

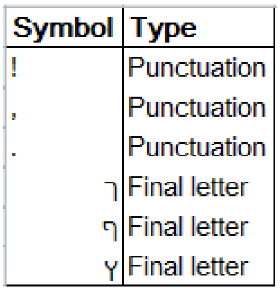
[1] Website: https://chengafni.wordpress.com/cpa/

[2] The code is written in Visual Basic for Applications for MS Excel and is available under the GNU General Public License. A version of this system for LibreOffice Calc is planned to appear in the future in order to free it from dependency on proprietary software.

[3] The system is accompanied by an external resource containing proposed sets of rules for several languages.

preparation” macro, which segments the text into words.

Segmentation is performed on the basis of blank spaces and additional word-dividing characters, which can be defined in CPA’s “Word dividers” table (Figure 1). There are two types of word dividers, which can be used for separating words even at the absence of a blank space: *punctuation* marks (e.g., comma) are deleted during segmentation, while *final letters* are not (final letters are special letter forms appearing only at word endings. See some examples from Hebrew in Figure 1). Once the text is transformed into a vector format, transcription can be performed. One of CPA tools (“Reconstruct corpus”) can then be used to recombine the phonetic word forms according to the structure of the original text.



Symbol	Type
!	Punctuation
,	Punctuation
.	Punctuation
ך	Final letter
ף	Final letter
ץ	Final letter

Figure 1: Word dividing symbols

3 Phonetic Dictionaries

For languages with irregular spelling or high proportion of homographs, such as English, Hebrew, and Arabic, automatic phonetic transcription requires a source of ready-made phonetic forms (i.e., a phonetic dictionary) for irregular and ambiguous words. CPA has a built-in macro that can import such ready-made forms. The macro receives as input a vector of written words to-be-transcribed and a phonetic dictionary – a table of written word forms and corresponding phonetic forms. The macro matches phonetic forms from the dictionary to written words in the vector. For words that are not found in the phonetic dictionary, transcription needs to be generated, either manually or with the automatic linguistic model (see next section). However, once the additional phonetic word forms are supplied, CPA can add them to the phonetic dictionary for future use.

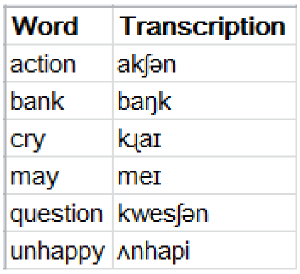
4 The Linguistic Model

For languages with some degree of regular mapping between spelling and sound, phonetic transcription of text can be generated automatically on the basis of deterministic rules. This section describes a multi-stage model of automatic phonetic transcription, guided by linguistic principles. Underlying this model is the assumption that, for any regular orthographic system, phonetic transcription rules can be defined in terms of a small set of general operations. The general operations themselves are hard-coded in the software, but an unlimited number of language-specific rules can be defined on the basis of these operations. This flexible method allows the system to produce automatic phonetic transcription for every language that has, at least partly, regular orthography.

The proposed model has four components, which will be described in the following subsections. The components operate independently of one another, but they should be applied in the order in which they are listed. The first component alone produces sufficient results for most purposes. If needed, the additional three components can be used, together, for fine-tuning.

4.1 Pre-Prosody Transcription

This component takes as input the vector of words to-be-transcribed, a table of pre-prosody transcription rules (Table 1), and a table containing sets of symbols and strings, called “phono-orthographic groups” (Table 2).[4] The pre-prosody transcription[5] applies the transcription rules in successive order to the list of words. Entities



Word	Transcription
action	akʃən
bank	baŋk
cry	kɻaɪ
may	meɪ
question	kwesʃən
unhappy	ʌnhapi

Figure 2: Transcribed words

[4] The tables of rules are generated manually, in principle. CPA has a set of editable tables that contain proposed rules for several languages (see also 2.1).

[5] The term ‘pre-prosody’ indicates that the transcription rules applied by this component disregard the prosodic properties of the word, including syllable structure and stress pattern.

	Target	**Output**	**Type**	**Preceding environment**	**Following environment**
1	ph	f			
2	c				k
3	ay	eɪ		#?	#
4	[cg]	[sj]			[Front_vowel]
5	[Consonant][Consonant]	[Consonant]	Degemination		
6	◌ّ		Lengthening		
7	[V_diac][C_diac]	[C_diac][V_diac]	Metathesis		

Table 1: Pre-prosody transcription rules

defined in the table of phono-orthographic groups may be called by transcription rules. The output of the process is a vector of phonetic forms corresponding to the written words (Figure 2).

The table of pre-prosody transcription rules has five fields (Table 1): (1) Target: the input to the rule, i.e., the string to be converted. All rules must have a value for the target string. The other fields are optional. (2) Output: the string replacing the target; if left empty, the target string will be deleted. (3) Type: the type of operation to be performed by the rule; if left empty, simple substitution will be performed (see below for other types of rules). (4) Preceding environment, and (5) Following environment: these fields are used for formulating context-sensitive rules. When either field is not empty, the transcription rule will apply only to words in which the target string is preceded by the 'preceding environment string' and/or followed by the 'following environment string'.

Substitution rules can be used for simple grapheme-to-phoneme conversions, which can be either context-free or context-sensitive. For example, rule 1 in Table 1 is a context-free substitution of *ph* by *f* in words such as *phone* (/foʊn/). Rule 2 in Table 1 is an example for context-sensitive rule – deleting *c* before *k* in words such as *back* (/bæk/).

Substitution rules may include wildcards to define more general entities. Three types of wildcards are defined in the software: a question mark (?) stands for any single character in the target, output or environment strings; an asterisk (*) stands for any number of successive characters; and, a hash sign (#) represents word boundaries. Wildcards can be used for defining phonological and morphological words patterns. For example, rule 3 in Table 1 captures the pronunciation of *ay* at the end of three-letter English words, such as *bay* (/beɪ/). The question mark in this rule indicates that the rule applies to *ay* sequences preceded by a single character. The hash signs suggest that the rule applies only when word boundaries are present at both edges.

Group	**Members**
cg	c,g
sj	s,ʤ
Front_vowel	e,i,y
Consonant	b,d,f,g,l,m,n,p,r,s,t,z
C_diac	◌ּ,◌ׂ,◌ׁ,'
V_diac	◌ְ,◌ֱ,◌ֹ,◌ָ,◌ַ,◌ֶ,◌ֵ,◌ִ

Table 2: Phono-orthographic groups

Rules can also be generalized by the inclusion of phono-orthographic groups, defined in a separate table. Each entry in the table of phono-orthographic groups has two fields (Table 2): the name of the group, and its members. For example, the term *Front_vowel* can be used for grouping *e*, *i*, and *y*.

Transcription rules can include phono-orthographic groups by enclosing the name of the group between brackets (e.g., [Front_vowel]). When a group is embedded in a transcription rule, the algorithm converts the compact rule into a set of simple rules, each applying to a different member of the group. For example, rule 4 in Table 1 uses groups to capture the pronunciation of *c* and *g* before front vowels (/s/ in *cent* /sent/ and /ʤ/ in *gene* /ʤiːn/, respectively). This single, compact rule stands for six simple rules: c→s/_e, c→s/_i, c→s/_y, g→ʤ/_e, g→ʤ/_i, g→ʤ/_y (where the formula A→B/_X is read: A becomes B before X).

In addition to substitution rules, several types of special operations can be used by specifying the name of the operation in the 'Type' field in the table of rules. Three types of operations are defined in the software: *degemination*, *lengthening*, and *metathesis*.

Degemination is used for collapsing a sequence of two identical phones when pronounced as a single, short sound. For example, rule 5 in Table 1 collapses sequences of identical consonants (e.g., *mm* is pronounced as a single *m* in *hammer*).

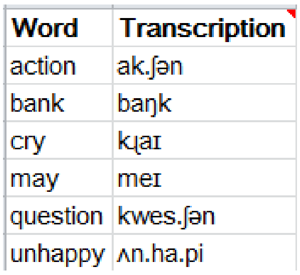
Lengthening realizes the function of diacritical marks of lengthening/gemination. For example, rule 6 in Table 1 realizes the function of the Arabic Shaddah (e.g., the letter م is pronounced /m/ in its plain form, but as /mm/ when modified by a Shaddah, i.e.,مّ).

Finally, *metathesis* switches the order of elements in sequences of two phono-orthographic groups (i.e., if the target contains a member of group 1 followed by a member of group 2, they are switched in the output). For example, in pointed Hebrew scripts, diacritics are used for indicating vowels as well as for modifying the phonetic value of consonants. A single letter can host both consonant and vowel diacritics (the C_diac and V_diac groups in Table 2, respectively). Thus, the string בַּ, pronounced /ba/, is composed of the letter ב (representing the consonants /b/ and /v/), a consonant diacritic ◌ּ (specifying the consonant /b/), and a vowel diacritic ◌ַ (representing the vowel /a/). Although the order in which these diacritics are attached to the letter does not affect the visual form of the text, it is important for the purpose of phonetic transcription – consonant diacritics must be attached before vowel diacritics. For example, the string בַּ can be formed by combining the three elements in two ways: ב+◌ּ+◌ַ, or ב+◌ַ+◌ּ. However, only the first order reflects the phonological structure of the string. A rule of metathesis can be defined to switch the order in sequences of vowel diacritics + consonant diacritic to guarantee correct ordering (rule 7 in Table 1).

After performing the pre-prosody transcription, certain modifications might be needed due to phonological processes related to prosodic structure. This post-prosody transcription (see 4.4) requires that the phonetic forms be parsed into syllables and have stress markers assigned to them. These components are described below.

4.2 Syllabification

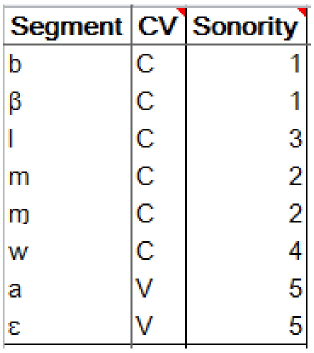
This component takes as input a vector of phonetic word forms, a list of binary parameters and parameter weights, and a phonetic table. The output is a vector of syllabified phonetic word forms (Figure 3). The basic sites of syllable boundaries are around local sonority minima (e.g., the boldfaced consonants in fæ**m**ə**l**i ‘family’ → fæ.**m**ə.**l**i).



Word	Transcription
action	ak.ʃən
bank	baŋk
cry	kɹaɪ
may	meɪ
question	kwes.ʃən
unhappy	ʌn.ha.pi

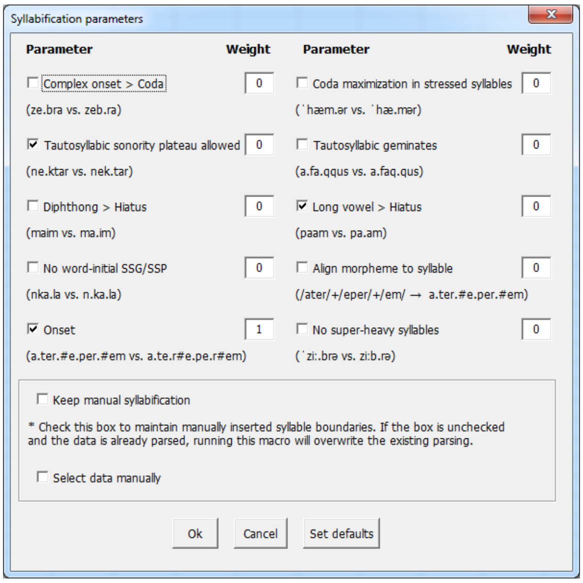
Figure 3: Syllabified phonetic word forms

In order to determine sites of sonority minimum, the syllabification procedure converts the phonetic word forms into strings of sonority levels. Sonority levels are non-negative integers specified for each phone in the phonetic table of CPA (Figure 4). For example, if fricatives, nasals, liquids, and vowels have sonority levels of 1, 2, 3, and 5, respectively, the sonority-level representation of *fæməli* will be 152535. In this representation, 2 and 3 are local sonority minima (Sonority minima at word edges are ignored).



Segment	CV	Sonority
b	C	1
β	C	1
l	C	3
m	C	2
m	C	2
w	C	4
a	V	5
ɛ	V	5

Figure 4: The phonetic table



Syllabification parameters

Parameter	Weight	Parameter	Weight
☐ Complex onset > Coda (ze.bra vs. zeb.ra)	0	☐ Coda maximization in stressed syllables (ˈhæm.ər vs. ˈhæ.mər)	0
☑ Tautosyllabic sonority plateau allowed (ne.ktar vs. nek.tar)	0	☐ Tautosyllabic geminates (a.fa.qqus vs. a.faq.qus)	0
☐ Diphthong > Hiatus (maim vs. ma.im)	0	☑ Long vowel > Hiatus (paam vs. pa.am)	0
☐ No word-initial SSG/SSP (nka.la vs. n.ka.la)	0	☐ Align morpheme to syllable (/ater/+/eper/+/em/ → a.ter.#e.per.#em)	0
☑ Onset (a.ter.#e.per.#em vs. a.te.r#e.pe.r#em)	1	☐ No super-heavy syllables (ˈzi:.brə vs. zi:b.rə)	0

☐ Keep manual syllabification

* Check this box to maintain manually inserted syllable boundaries. If the box is unchecked and the data is already parsed, running this macro will overwrite the existing parsing.

☐ Select data manually

Figure 5: Syllabification parameters

	Trigger	Tier	Position	Process	Result
1	[+STRID][+STRID]	Features	Coda	Vowel epenthesis	ə
2	Unstressed vowel	CV		Vowel reduction	ə
3	Sonority decrease	Sonority	Onset	Vowel epenthesis	ə

Table 3: Post-prosody transcription rules

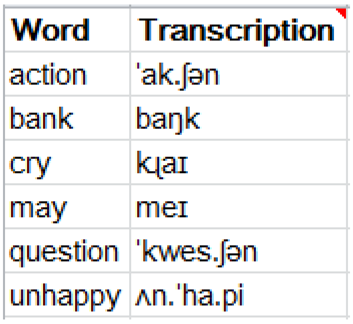
The basic sites of syllable boundaries can be adjusted by a set of binary parameters, which handle various cases, such as consonant sequences (e.g., whether /dɪspleɪ/ 'display' should be parsed as dɪs.pleɪ or dɪ.spleɪ). The system currently has 10 built-in parameters, which can be switched on and off according to the properties of the examined language (Figure 5). Some of the parameters require phones to be recognized as vowels or consonants. This information is also specified in CPA's phonetic table (Figure 4). For example, if the *Complex onset > Coda* parameter is switched on, consonant sequences will be parsed as complex onsets (dɪ.spleɪ). If the parameter is switched off, consonant sequences will be split between coda and onset positions (dɪs.pleɪ).

When two parameters are potentially in conflict, they can be ranked relative to each other by assigning different integer weights to them. For example, if the *Onset* parameter is on, onsetless syllables will be dispreferred (e.g., ˈmʌni → ˈmʌ.ni 'money'). This can be overridden (e.g., ˈmʌn.i) by switching on the *Coda maximization in stressed syllables* parameter and giving it a higher weight than the onset parameter (this requires that stress would be marked on the word before running syllabification).

4.3 Stress Assignment

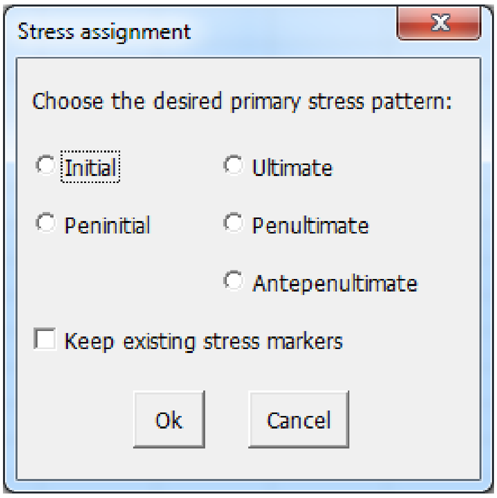
When the phonetic transcription requires modifications due to processes related to stress (e.g., reduction of unstressed vowels), stress markers should be added to the phonetic word forms. The stress assignment component of the software takes as input a vector of syllabified words and the desired stress pattern. The output is a vector of syllabified words with stress markers inserted at the appropriate positions (e.g., for the input word ak.ʃən 'action' and penultimate stress pattern, the output will be ˈak.ʃən; Figure 6).

The software has five built-in stress patterns (at present, only primary stress is handled): Initial, Peninitial, Ultimate, Penultimate, and Antepenultimate (Figure 7). For languages with a non-fixed stress pattern (e.g., English), this procedure can be used for applying the most frequent stress pattern. Manual corrections can be made afterwards. If stress position depends on the number of syllables, it is possible to run stress assignment multiple times, starting with the rule for the longer words. Checking the 'Keep existing stress markers' option will prevent stress rules for shorter words from applying to longer words, for which stress has been assigned already.



Word	Transcription
action	ˈak.ʃən
bank	baŋk
cry	kɻaɪ
may	meɪ
question	ˈkwes.ʃən
unhappy	ʌn.ˈha.pi

Figure 6: Phonetic word forms with stress



Stress assignment

Choose the desired primary stress pattern:

○ Initial ○ Ultimate

○ Peninitial ○ Penultimate

○ Antepenultimate

□ Keep existing stress markers

Figure 7: Stress parameters

4.4 Post-Prosody Transcription

This components modifies phonetic word forms according to phonological rules related to prosodic structure. It takes as input a vector of phonetic word forms, a table of post-prosody transcription rules (Table 3), and a phonetic table. The procedure applies the transcription rules in successive order to the list of words.

The table of post-prosody transcription rules has five fields (Table 3): (1) Trigger: the phonological structure triggering the required modification. A trigger can be a specific element or a sequence of elements defined in the phonetic table (e.g., [+STRID][+STRID] stands for a sequence of two stridents such as /s/ and /z/). In addition, there are several types of special pre-defined triggers: The *Sonority decrease*, *Sonority increase*, and *Sonority plateau* triggers handle phone sequences in which the sonority level decreases, increases, or remains unchanged, respectively (see Figure 4). The *No vowel* trigger handles syllables with no vowels. The *Unstressed vowel* trigger handles vowels in unstressed syllables.

(2) Tier: the phonological tier relevant to the trigger. *Features* tier is used with phonological features triggers (e.g., [+STRID]), while *CV* tier is used with *No vowel* and *Unstressed vowel* triggers. *Sonority* tier is used with all sonority-related triggers.

(3) Position: for triggers applying to consonants (sonority and feature triggers), this field indicates the prosodic position (*Onset* or *Coda*) in which the trigger must be found in order to trigger the modification.

(4) Process: the type of modification applied to phonetic word forms in which the trigger is found. Currently, the software can perform two types of modifications: *Vowel epenthesis* inserts a vowel to correct ill-formed sequences, and *Vowel reduction* replaces unstressed vowels with a default neutral vowel.

(5) Result: this field specifies inserted elements (epenthetic vowels and neutral vowels).

The following examples demonstrate the application of post-prosody rules. Rule 1 in Table 3 inserts an epenthetic ə to break sequences of two stridents in coda position (e.g., makss → maksəs 'Max's', where *makss* is the output of the pre-prosody transcription, which converted *M* to *m* and *x* to *ks*, and deleted the apostrophe in Max's). Rule 2 in Table 3 replaces vowels in unstressed syllables by ə (e.g., ˈe.le.fant → ˈe.lə.fənt 'elephant', where ˈe.le.fant is the result of pre-prosody transcription, syllabification and assignment of antepenultimate stress to elephant).

5 Discussion

This paper describes a system of text-to-phonetics conversion. The system is incorporated in a general-purpose linguistic software that includes tools for building dictionaries, as well as corpus analysis functions. Thus, the described system can help studying the phonological properties of text corpora and it is also useful for creating resources for under-resourced languages. For example, it was used for creating a phonological dictionary for Hebrew (Gafni, 2019). In addition, the linguistic model of the software, by itself, can be used as a research and educational tool. The pre-prosody transcription tool, in particular, can be used for exploring and demonstrating the effect of rule-ordering – a common practice in theoretical phonology. In fact, the studied language need not have a writing system at all; the input corpus can be a list of hypothesized phonological underlying representations, and the transcription rules can be phonological rules transforming the underlying representations to surface representations.

In addition, the linguistic model can be used for calculating indices of linguistic complexity by assessing the proportion of words that have regular spelling in a given language and the number of deterministic rules needed to capture the patterns of orthographic regularity in a language. Such measures of complexity can be valuable for literacy education (e.g., Smythe et al., 2008).

It should be noted that the described system is still under development. At its current state, the transcription system can perform perfectly on completely regular orthographies with a fixed stress pattern. Several planned improvements will allow the system to handle more complex cases. For example, the stress assignment component should handle secondary stress and stress rules that are sensitive to syllable weight. In addition, the post-prosody transcription should include more options, such as referring to pretonic syllables, which are relevant sites for certain phonological processes like vowel reduction in Russian (Asherov et al., 2016).

Furthermore, the generalizability of the system can greatly improve by adding machine learning procedures, such as sequence-to-sequence models with greedy decoding (Chae et al., 2018). This will allow the system to generate rules automatically based on examples. It will also be able to handle cases of homography (e.g., whether *wind* should be transcribed /wɪnd/ (noun) or /waɪnd/ (verb)) by analyzing token frequency and contextual effects (syntax and semantics). Such improvements will make the transcription system more powerful and reliable.

References

Daniel Asherov, Alon Fishman, and Evan-Gary Cohen. 2016. Vowel Reduction in Israeli Heritage Russian. *Heritage Language Journal*, 2:113–133.

Timur Baytukalov. 2019. EasyPronunciation.com: All-in-one solution to learn pronunciation online.

Moon Jung Chae, Kyubyong Park, Linhyun Bang, Soobin Suh, Longhyuk Park, Namju Kimt, and Longhun Park. 2018. Convolutional sequence to sequence model with non-sequential greedy decoding for grapheme to phoneme conversion. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, pages 2486–2490.

Chen Gafni. 2015. Child Phonology Analyzer: processing and analyzing transcribed speech. In The Scottish Consortium for ICPhS 2015, editor, *Proceedings of the 18th International Congress of Phonetic Sciences.*, page 1–5, paper number 531, Glasgow, UK: the University of Glasgow.

Chen Gafni. 2019. General Lexicons of Hebrew: Resources for Linguistic and Psycholinguistic Research (version 1.0).

Ian Smythe, John Everatt, Nasser Al-Menaye, Xianyou He, Simone Capellini, Eva Gyarmathy, and Linda S. Siegel. 2008. Predictors of word-level literacy amongst Grade 3 children in five diverse languages. *Dyslexia*, 14(3):170–187.

Eran Tomer. 2012. *Automatic Hebrew Text Vocalization*. Ph.D. thesis, Ben-Gurion University of the Negev.

Two Discourse Tree - Based Approaches to Indexing Answers

Boris Galitsky[1] **and Dmitry Ilvovsky**[2]
[1]Oracle Inc. Redwood Shores CA
[2]National Research University Higher School of Economics
boris.galitsky@oracle.com; dilvovsky@hse.ru

Abstract

We explore anatomy of answers with respect to which text fragments from an answer are worth matching with a question and which should not be matched. We apply the Rhetorical Structure Theory to build a discourse tree of an answer and select elementary discourse units that are suitable for indexing. Manual rules for selection of these discourse units as well as automated classification based on web search engine mining are evaluated concerning improving search accuracy. We form two sets of question-answer pairs for FAQ and community QA search domains and use them for evaluation of the proposed indexing methodology, which delivers up to 16 percent improvement in search recall.

1 Introduction

Much online content is available via question-answer pairs such as frequently-asked questions stored on customer portals or internal company portals. Question-answer pairs can be an efficient manner to familiarize a user with content. In some cases, autonomous agents (chatbots) can import such question-answer pairs in order to field user questions.

But such question-answer pairs can contain content that is not central to a topic of an answer. For example, content can include text that is irrelevant or misleading, non-responsive to the particular question, or is neutral and not helpful. If irrelevant text is indexed by a keyword-based search engine, the precision of the search engine is lowered. Moreover, an autonomous agent attempting to answer a user question based on erroneously-indexed text may answer the question incorrectly, resulting in lowered user confidence in the agent. Despite the fact that standard relevance techniques such as ontology, keyword frequency models and separate discourse features (Chali et al., 2009; Jansen et al., 2014) can be applied to solve this problem, a solution is needed for identifying informative parts from all text.

In this paper we propose a new discourse-based approach to determine informative parts of an answer. This approach accesses a body of text including fragments and creates a searchable index including multiple entries, each entry corresponding to a selected fragment. We propose two different methods of fragment selection based on rules and on classification model respectively.

The paper structure is as follows. In Section 2 we introduce the methodology of ***rhetorical anatomy of an answer*** and present an example of that. In Section 3 we propose two Q/A algorithms which is the core part of our approach. In Section 4 we describe and discuss evaluation for the question answering task on a few datasets that were compiled for this research.

2 Rhetoric Anatomy of an Answer

2.1 RST and Discourse Trees

Discourse analysis was proved to be useful in different aspects of question-answering: answer extraction (Zong et al., 2011), modeling rationale in design questions (Kim et al., 2004), query expansion based on relations between sequential questions (Sun and Chai, 2007), etc.

Discourse trees (DT) originate from Rhetorical Structure Theory (RST, Mann and Thompson, 1988). RST models a logical organization of text, relying on relations between parts of text. RST simulates text coherence by forming a hierarchical, connected structure of texts via ***discourse trees***.

Rhetoric relations are split into the classes of coordinate and subordinate; these relations hold across two or more text spans and therefore implement coherence. These text spans are called elementary discourse units (EDUs). The leaves of a discourse tree correspond to EDUs, the contiguous atomic text spans. Adjacent EDUs are connected by coherence relations (e.g., attribution, sequence), forming higher-level discourse units.

The term "nuclearity" in RST refers to which text segment, fragment, or span, is more central to an author's purpose. A "***nucleus***" refers to a span of text that is more central to an author's purpose than a "***satellite***", which is less central to the topic.

More particularly, we use the determined EDUs of a discourse tree for a body of text and the relations between the EDUs to determine which EDUs should be indexed for search. Different rhetorical relations (e.g., *elaboration*, *contrast*, etc.) can employ different rules.

In general, we hypothesize that a satellite may express a detail of information that is unlikely to be explicitly queried by a user (Galitsky, 2015; Jasinskaja and Karagjosova, 2017).

2.2 Example of Analysis

Let's illustrate our analysis with a question-answer pair and a discourse tree for an answer.

Q: *How should I plan to pay for taxes resulting from converting to a Roth IRA?*
A: *To help maximize your retirement savings, it's generally a good idea to consider not using the proceeds from the conversion to pay the resulting tax costs. Instead, you should consider using cash or other savings held in nonretirement accounts. Using retirement account funds to pay the taxes will reduce the amount you would have available to potentially grow tax-free in your new Roth IRA. Additionally, if you are under 59½, using funds from your retirement account could result in an additional 10% tax penalty, which may significantly reduce the potential benefit of conversion.*

The discourse tree for the answer is shown in Figure 1, and elementary discourse units selected for indexing are circled in green.

The answer could be obtained from a source such as a Frequently Asked Questions (FAQ) database, or a question-answer index. A question-answer index can include multiple questions and corresponding answers. But some fragments in each answer are more informative to answering the corresponding question than other fragments. For example, the phrase "*it is generally a good idea*" adds little to the answer, whereas "*consider not using the proceeds from the conversion*" is informative to the user who posed the original question. Each answer in the question-answer index may provide additional insight in terms of additional questions that can be answered, which are in turn indexed, increasing the usefulness of the data. For example, "*at what age do I pay a penalty for using retirement funds?*" could be answered by the text (e.g., "*age 59 ½*"). We can determine informative text from a body of text and such additional questions that can be answered from the body of text.

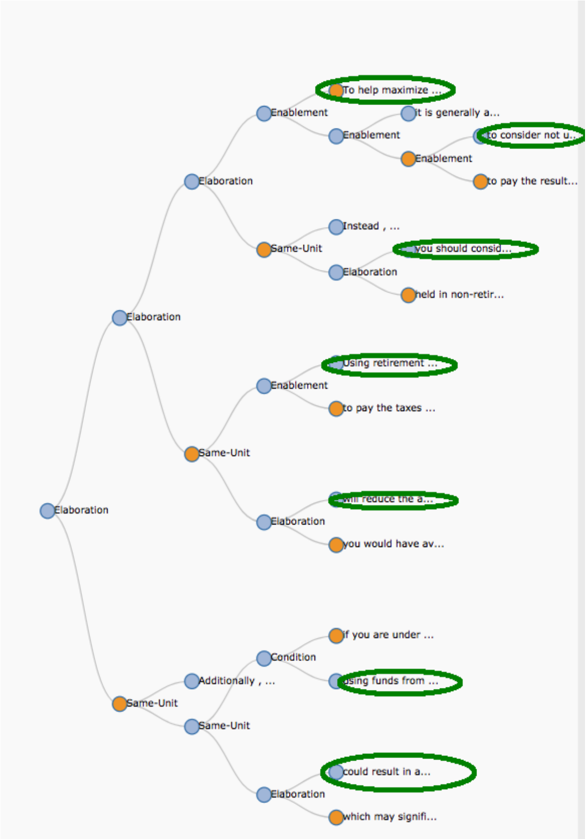


Figure 1: Discourse tree for an answer with the EDUs selected for indexing

2.3 Indexing Rules for Different Rhetorical Relations

The above hypothesis that only EDUs that are ***nucleus*** of rhetoric relations should be indexed and all ***satellite*** EDUs should not be selected for indexing is illustrated by the "*elaboration*" relationship where the nucleus expresses more important information than satellite. But the general rule described above can be subject to certain exceptions. For example, under certain conditions, the "*contrast*" relation can require indexing of the satellite rather than the nucleus. Additionally, for the “*same-unit*” and “*joint*” relations, both the nucleus and the satellite are indexed. Different rhetoric relations can have different rules, as shown in Table 1 below.

3 The Methodology of Question Answering

The developed methodology of the DT-based analysis of answers is going to be applied in the following way, given an index of Q/A pairs:

1. Search a user query against an index of available questions;
2. If no or too few results, generate additional search queries from the answers indexed by proposed approach;
3. If still no or too few results, search against original answers.

We now focus on 2) and consider two methods for indexing answers: **rule-based** and **classification-based**.

Relation	**Example**	**Indexing rule**
Elaboration	To achieve some state [nucleus] | *do this and that* [satellite]	Nucleus
Enablement	A query may be of the form “*how to achieve some state?*” but less likely be of the form “*what can I achieve doing this and that?*”	Nucleus
Condition	A query may be of the form “*how to achieve some state?*” but less likely of the form “*what can I achieve doing this and that?*”	When the question is of the type “***when/where/under what condition …***”, index the ***if*** part (the satellite).
Contrast	Index the nucleus. The satellite includes facts which are unusual, unexpected.	
Same-Unit, Joint	Index both nucleus and satellite because of the symmetric relationship of same-unit.	

Table 1: Indexing rules for rhetorical relations

Once our method construct the indexes, we can build the online search algorithm which combines default functionality provided by the Lucene search engine and syntactic similarity between answer and a query. Search results candidate are selected by Lucene and then matched with the query via finding a maximal common sub-parse tree (Galitsky, 2017).

3.1 Rule-Based Indexing

We take question-answer pairs and create, for each answer, a discourse tree using RST-parser (Surdeanu et al., 2015; Joty et al., 2013). For each non-terminal node in each answer, we then identify a rhetorical relationship associated with the non-terminal node and label each terminal node associated with the non-terminal node as either a ***nucleus*** or a ***satellite***. Then we apply a set of rules (see Table 1) associated with the rhetorical relationships and select, based on the rule, one or more of the fragment associated with the nucleus or the fragment associated with the satellite. Finally, we create a ***searchable index*** of additional questions which includes multiple entries corresponding to one of the selected fragments for the answers.

3.2 Classification-Based Indexing

We use machine learning to learn rules such as those depicted in Table 1. A machine learning problem is formulated as a classification problem that classifies EDUs into a first class that is suitable for indexing (i.e., ***informative***) and forming alternative questions for an answer and a second class that is not suitable for indexing (i.e., ***not informative***).

To accumulate training question-answer pairs with marked answers, we ran selection of queries against short texts. Because longer queries are necessary to assure a corresponding match is nontrivial, we used public question-answer Yahoo! Answers dataset (Webscope, 2017). More specifically, questions from this dataset were formed from a first sentence of the dataset and executed as queries by Microsoft Cognitive Services (Bing Search engine API). Search results which are short texts (4-6 sentences) were selected as such texts suitable for parsing and discourse analysis. Matched fragments of these texts were taken as elements of the training set. Such fragments from the top ten or more pages of search result formed a positive dataset, i.e. informative fragments. For the negative dataset, fragments with matched keywords from the set of lower ranked (100-1000+) search results pages were taken, as these results are assumed to be less relevant.

We applied SVM tree kernel learning (Moschitti, 2006; Severyn and Moschitti, 2012) to train the model since this algorithm is capable to learn directly on a parse tree structure.

4 Datasets and Evaluation

We used a few datasets to evaluate the contribution of our methodology to search quality.

Yahoo! Answer (Webscope, 2017) subset of question-answer pairs with broad topics where main question is a single sentence (possibly, compound) with ten-fifteen keywords. The dataset includes various domains, and domain knowledge coverage is shallow.

Financial questions[1] scraped from Fidelity.com. This dataset demonstrates how search relevance improvement may occur in a vertical domain with reasonable coverage.

Car repair conversations[2] selected from www.2carpros.com including car problem descriptions and recommendation on how to rectify them. These pairs were extracted from dialogues as first and second utterances.

For each search session, we only consider the first results and reject the others. For all these datasets we assume that there is only one correct answer (from the Q/A pair) and the rest of answers are incorrect.

Evaluation results for the proposed methodology are presented in Table 3. Recall of the baseline search is on average 78% including the improvement by 8% by using syntactic generalization on top of Lucene search (not shown). The relevance of this system is determined by many factors and is therefore not very insightful, so we focus at the change in recall (Δ), from this search system to the one extended by the proposed approach.

Dataset	**Question/Answer**	**Total #**	**# generated AQ / # sent**	**Avg # words**
Yahoo! Answers	**Q**	3700	5.5	12.3
	A	3700	8.1	124.1
Fidelity	**Q**	500	3.4	6.2
	A	500	6.2	118.0
Car Repair	**Q**	10000	4.2	5.5
	A	10000	7.0	141.3

Table 2: Dataset statistics

The proposed method delivers about 13 % improvements in the recall have the precision almost unaffected, for the Nucleus/Satellite rules. There is a further 3% improvement by using the automated classifier of EDUs. Since the deployment of such classifier in a domain-dependent manner is associated with substantial efforts, it is not necessarily recommended when this 3% improvement in search accuracy is not critical.

We also compare performance of the proposed search on the extended framework derived from SQuAD 2.0 dataset (Rajpurkar et al., 2018) and applied to *why?* and how-*to?* questions. Instead of addressing a question to a single Wikipedia text as standard evaluations do, we run them against all text. We use our approach vs neural extractive reading comprehension one and exceed recall of BiDaf (Gardner et al., 2017) and DeepPavlov (Burtsev et al., 2018) by at least 8% with the search engine trained on our corpus (Table 4).

Dataset / Method	**Baseline**		**Nucleus /Satellite rules,** improvement		**Classification-based,** improvement	
	R	P	ΔR, %	ΔP, %	ΔR, %	ΔP, %
Yahoo! Answers	79	74	+12.5	+0.1	+14	-0.04
Fidelity	77	80	+10	-0.1	+6	+0.1
Car Repair	79	81	+**16**	+0.0	+**18**	+0.0

Table 3: Evaluation results for new datasets

[1] https://github.com/bgalitsky/relevance-based-on-parse-trees/examples/Fidelity_FAQs_AnswerAnatomyDataset1.csv.zip

[2] https://github.com/bgalitsky/relevance-based-on-parse-trees/examples/CarRepairData_AnswerAnatomyDataset2.csv.zip.

Method	**Recall**	**Precision**
BiDaf (AllenNLP)	68	71
DeepPavlov	67	72
Rule-based	75	71
Classification-based	**76**	71

Table 4: Evaluation results for SQuAD dataset

5 Conclusions

In the search engines and chat bot industry, whole texts are usually indexed for search. Because of that, frequently irrelevant answers are delivered because their insignificant keywords (the ones providing auxiliary information and not central for the document) were matched. To overcome this well-known problem, only questions from Q/A pairs are indexed, which dramatically decreases the search recall. To address this limitation of indexing, we proposed and evaluated our approach of indexing only those EDUs of text which are determined to be important (and therefore form alternative questions). This substantially improves the recall in applications such as FAQ search where only questions of Q/A pairs are indexed.

Acknowledgements

Section 3 (algorithms of question answering) were written by Dmitry Ilvovsky supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia. Section 4 (experimental investigations) was prepared within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'. The rest of the paper were written and performed at Oracle Corp.

References

Chali, Y. Shafiq R. Joty, and Sadid A. Hasan. 2009. *Complex question answering: unsupervised learning approaches and experiments.* J. Artif. Int. Res. 35, 1 (May 2009), 1-47.

Galitsky, B, Ilvovsky, D. and Kuznetsov SO. 2015. *Rhetoric Map of an Answer to Compound Queries.* ACL-2, 681–686.

Galitsky, B. 2017. *Matching parse thickets for open domain question answering*. Data & Knowledge Engineering, Volume 107, January 2017, Pages 24-50.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.H., Peters, M., Schmitz, M., & Zettlemoyer, L.S. *A Deep Semantic Natural Language Processing Platform*. arXiv:1803.07640.

Jansen, P., M. Surdeanu, and P. Clark. 2014. *Discourse Complements Lexical Semantics for Nonfactoid Answer Reranking.* ACL.

Jasinskaja, K., Karagjosova, E. 2017. *Rhetorical Relations: The Companion to Semantics.* Oxford: Wiley.

Joty, Shafiq R, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. *Combining intra-and multi- sentential rhetorical parsing for document-level discourse analysis.* In ACL (1), pages 486–496.

Kim, S., Bracewell, R., Wallace, K. 2004. *From Discourse Analysis to Answering Design Questions*. International Workshop on the Application of Language and Semantic Technologies to support Knowledge Management Processes (EKAW 2004). At: Whittlebury Hall, Northamptonshire, UK.

M. Burtsev, A. Seliverstov, R. Airapetyan, M. Arkhipov, D. Baymurzina, N. Bushkov, O. Gureenkova, T. Khakhulin, Y. Kuratov, D. Kuznetsov, A. Litinsky, V. Logacheva, A. Lymar, V. Malykh, M. Petrov, V. Polulyakh, L. Pugachev, A. Sorokin, M. Vikhreva, M. Zaynutdinov. 2018. *DeepPavlov: Open-Source Library for Dialogue Systems. ACL-System Demonstrations*. p. 122–127.

M. Sun, J. Y. Chai. 2007. *Discourse Processing for Context Question Answering Based on Linguistic Knowledge.* Knowledge-Based Systems 20(6)(6): 511-526

Mann, William and Sandra Thompson. 1988. *Rhetorical structure theory: Towards a functional theory of text organization. Text* - Interdisciplinary Journal for the Study of Discourse, 8(3):243–281.

Moschitti, A. 2006. *Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees*. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany.

Pranav Rajpurkar, Robin Jia, Percy Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. arxiv.org/abs/1806.03822

Severyn, A. Moschitti, A. 2012. *Fast Support Vector Machines for Convolution Tree Kernels.* Data Mining Knowledge Discovery 25: 325-357.

Surdeanu, M., Thomas Hicks, and Marco A. Valenzuela-Escarcega. 2015. *Two Practical Rhetorical*

Structure Theory Parsers. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies: Software Demonstrations (NAACL HLT), 2015.

Webscope 2017. Yahoo! Answers Dataset. https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

Zong, H., Yu, Z., Guo, J., Xian, Y., Li, J. 2011. *An answer extraction method based on discourse structure and rank learning.* 7th International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE).

Discourse-Based Approach to Involvement of Background Knowledge for Question Answering

Boris Galitsky[1] **and Dmitry Ilvovsky**[2]
[1]Oracle Inc. Redwood Shores CA
[2]National Research University Higher School of Economics
boris.galitsky@oracle.com; dilvovsky@hse.ru

Abstract

We introduce a concept of a *virtual discourse tree* to improve question answering (Q/A) recall for complex, multi-sentence questions. Augmenting the discourse tree of an answer with tree fragments obtained from text corpora playing the role of ontology, we obtain on the fly a canonical discourse representation of this answer that is independent of the thought structure of a given author. This mechanism is critical for finding an answer that is not only relevant in terms of questions entities but also in terms of inter-relations between these entities in an answer and its style. We evaluate the Q/A system enabled with virtual discourse trees and observe a substantial increase of performance answering complex questions such as Yahoo! Answers and www.2carpros.com.

1 Introduction

In spite of the great success of search technologies, the problem of leveraging background knowledge is still on the agenda of search engineering, for both conventional and learning-based systems. Background knowledge ontologies are difficult and expensive to build, and knowledge graphs – based approaches usually have a limited expressiveness and coverage. In this study we explore how a discourse analysis (which is domain-independent) can substitute certain features of ontology-based search. There are few popular discourse theories describing how Discourse Trees (DT) can be constructed from the text. In our work we used Rhetorical Structure Theory (RST, Mann and Thompson, 1988).

Ontologies are in great demand for answering complex, multi-sentence questions with a precise answer in such domain as finance, legal, and health. In the educational domain this type of questions is referred to as *convergent*: answers to these types of questions are usually within a very limited range of acceptable accuracy. These may be at several different levels of cognition including comprehension, application, analysis, or ones where the answerer makes inferences or conjectures based on material read, presented or known. Answering convergent questions is an underexplored Q/A domain that can leverage discourse analysis (Kuyten et al, 2015).

Discourse trees have became a standard for representing how thoughts are organized in text, in particular in a paragraph of text, such as an answer. Discourse-level analysis has been shown to assist in a number of NLP tasks where learning linguistic structures is essential (Louis et al., 2010; Lioma et al., 2012). DTs outline the relationship in between entities being introduced by an author. Obviously, there are multiple ways the same entities and their attributes are introduced, and not all rhetoric relations that hold between these entities occur in a DT for a given paragraph.

When DTs are used to coordinate questions and answers, we would want to obtain an "ideal" DT for an answer, where all rhetoric relations between involved entities occur. To do that, we need to augment an actual (available) DT of answer instance with a certain rhetorical relations which are missing in the given answer instance but can be mined from text corpora or from the web. Hence to verify that an answer *A* is good for a given question *Q*, we first verify that their DTs (*DT-A* and *DT-Q*) agree and after that we usually need to augment the *DT-A* with fragments of other DTs to make sure all entities in *Q* are communicated (addressed) in augmented *DT-A*.

Hence instead of relying on an ontology that would have definitions of entities which are missing in a candidate answer we mine for the rhetorical relations between these entities online. This procedure allows us to avoid an offline building of bulky and costly ontologies. At the same time, the proposed approach can be implemented on top of a conventional search engine.

The paper structure is as follows. In Section 2 we compare the related work with our proposal. In Section 3 we introduce the concept of a ***virtual discourse tree*** and present a number of examples illustrating how they can be used and constructed. In Section 4 we propose Q/A filtering algorithm which is the core part of our approach. In Section 5 we describe and discuss evaluation for the question answering task on a few datasets that were compiled for this research.

2 Related Work

2.1 Discourse and IR

Typically, every part in most coherent text has some plausible reason for its presence, some function that it performs to the overall semantics of the text. Rhetorical relations, e.g. *contrast*, *cause*, *explanation*, describe how the parts of a text are linked to each other. Rhetorical relations indicate the different ways in which the parts of a text are linked to each other to form a coherent whole.

Marir and Haouam (2004) introduced a thematic relationship between parts of text using RST based on cue phrases to determine the set of rhetorical relations. Once these structures are determined, they are put in an index, which can then be searched not only by keywords, as traditional information retrieval systems do, but also by rhetorical relations.

It was observed (Teufel and Moens, 2002) that different rhetorical relations perform differently across evaluation measures and query sets. The four rhetorical relations that improve performance over the baseline consistently for all evaluation measures and query sets are: *background, cause-result, condition* and *topic-comment*. Topic-comment is one of the overall best-performing rhetorical relations, which in simple terms means that boosting the weight of the topical part of a document improves its estimation of relevance. Regretfully these relations are relatively rare.

Sun and Chai (2007) investigated the role of discourse processing and its implication on query expansion for a sequence of questions in scenario-based context Q/A. They consider a sequence of questions as a mini discourse. An empirical examination of three discourse theoretic models indicates that their discourse-based approach can significantly improve Q/A performance over a baseline of plain reference resolution.

In a different task (Wang et al, 2010) authors parse Web user forum threads to determine the discourse dependencies between posts in order to improve information access over Web forum archives.

Suwandaratna and Perera (2010) present a re-ranking approach for Web search that uses discourse structure. They report a heuristic algorithm for refining search results based on their rhetorical relations. Their implementation and evaluation is partly based on a series of ad-hoc choices, making it hard to compare with other approaches. They report a positive user-based evaluation of their system for ten test cases.

Since rhetoric parsers for English (Joty et al., 2013, Surdeanu, 2015) have become more available and accurate, their application in search engine indexing is becoming more feasible. Precision and recall of search systems ignore discourse level information and users do not find products, services and information they need. It was shown that discourse features are valuable for passage re-ranking (Jansen et al., 2014). DTs have been also found to assist in answer indexing to make search more relevant: query keyword should occur in nucleus rather than a satellite of a rhetoric relation (Galitsky et al., 2015). In this study we go beyond leveraging discourse features and construct DTs from actual candidate answers and also virtual DTs for necessary background knowledge.

2.2 Discourse Analysis and Entities

At any point in the discourse, some entities are considered more salient than others (occurring in nucleus parts of DTs), and consequently are expected to exhibit different properties. In Centering Theory (Poesio et al., 2004), entity importance determines how they are realized in an utterance, including pronominalized relation between them.

Barzilay and Lapata (2008) automatically abstracts a text into a set of entity transition sequences and records distributional, syntactic, and referential information about discourse entities. The authors formulated the coherence assessment as a learning task and show that their entity-based representation is well-suited for ranking-based generation and text classification tasks.

Nguyen and Joty (2017) presented a local coherence model based on a convolutional neural network that operates over the distributed representation of entity transitions in the grid representation of a text, can model sufficiently long entity transitions and can incorporate entity-specific features without losing generalization power.

Kuyten et al., (2015) developed a search engine that leverages the discourse structure in documents to overcome the limitations associated with the bag-of-words document representations in information retrieval. This system does not address the problem of rhetoric coordination between *Q* and *A*, but given a *Q*, this search engine can retrieve both relevant A and individual statements from A that describe some rhetorical relations to the query.

Our approach is to discover ontological relations between entities on the fly, finding document fragments where a rhetorical relation links these entities. Once all such text fragments are found, we add the respective DT fragments as virtual DTs to our main answer DT.

3 Answering Questions via Discourse Trees

3.1 Virtual Discourse Tree

The baseline requirement for an *A* to be relevant to *Q* is that entities (*E*) of *A* cover the entities of *Q*:

$$E\text{-}Q \subseteq E\text{-}A. \qquad (1)$$

Naturally, some *E-A* (entities in an answer) are not explicitly mentioned in *Q* but are needed to provide a recommendation yielded by *Q* (recipe-type *A*).

The next step for an *A* to be good for *Q* is to follow the logical flow of *Q*. Since it is hard to establish relations between entities *E*, which are domain dependent, we try to approximate these relations by using logical flow of *Q* and *A*, expressible in domain-independent terms, such as rhetorical relation. Hence we require a certain correspondence between DT-Q and DT-A, considering additional labels for DT nodes by ***entities*** (we denote such DT as EDT):

$$EDT\text{-}Q \sim EDT\text{-}A. \qquad (2)$$

However a common case is that some entities *E* are not explicitly mentioned in *Q* but instead are assumed. Moreover, some entities in *A* used to answer *Q* do not occur in *A* but instead are substituted by more specific or general entities do. How would we know that these more specific entities are indeed addressing issues from *Q*? We need some external, additional source which we call ***virtual*** *EDT-A* to establish these relationships.

This source contains the information on inter-relationships between E which is omitted in *Q* and/or *A* but is assumed to be known by the interlocutor. For an automated Q/A system, we want to obtain this knowledge at the discourse level:

$$EDT\text{-}Q \sim EDT\text{-}A + virtual\ EDT\text{-}A. \qquad (3)$$

3.2 Discourse Trees for Answer and Question

We start with a simple example:

Q: *What is an advantage of electric car?*

A: *No need for gas.*

How can search engine figure out that *A* is a good one for *Q*? We have an abstract general-sense entity *advantage* and a regular noun entity *car*. We need to link explicit entities in *A* {*need, gas*}. Fragments of a possible *virtual EDT-A* are shown below:

Q: *[When driving the cruise control][the engine will turn off][when I want to accelerate ,][although the check engine light was off .] [I have turned on the ignition][and listen for the engine pump running][to see][if it is building up vacuum .] [Could there be a problem with the brake sensor under the dash ?] [Looks like there could be a little play in the plug.]*

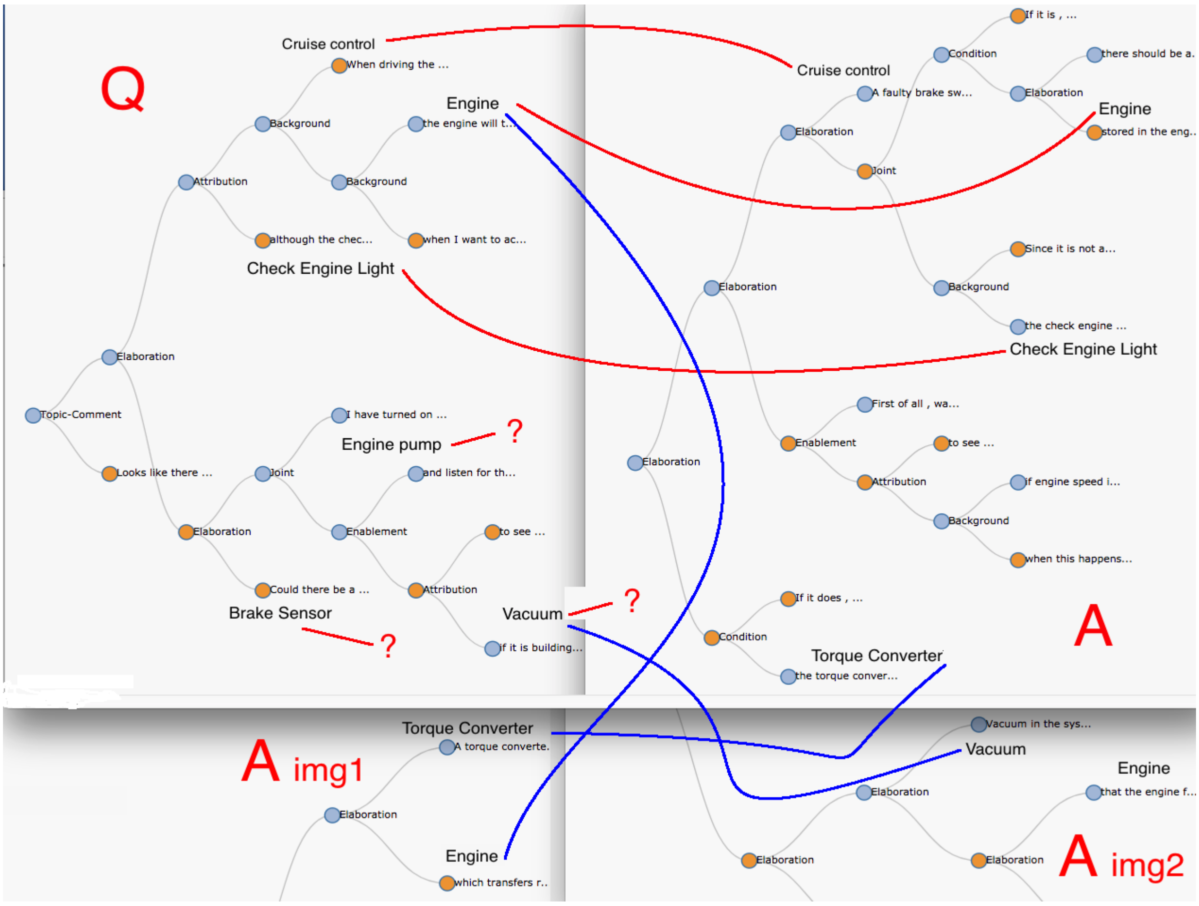
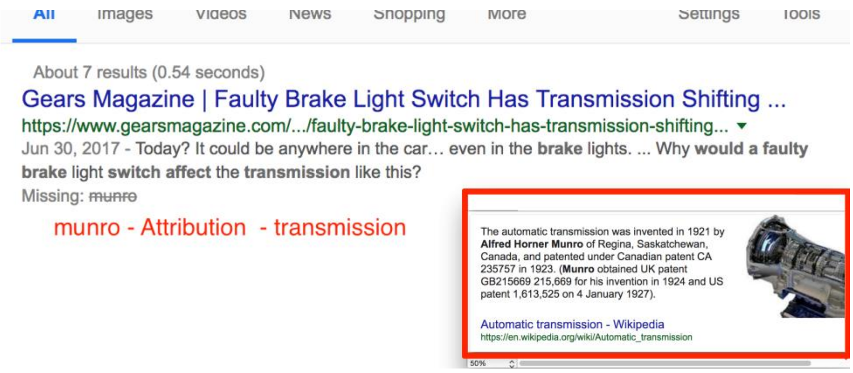


Figure 1: DTs of Q, A and imaginary $DT\text{-}A_{img1}$ and $DT\text{-}A_{img2}$



About 7 results (0.54 seconds)

Gears Magazine | Faulty Brake Light Switch Has Transmission Shifting ...

https://www.gearsmagazine.com/.../faulty-brake-light-switch-has-transmission-shifting...

Jun 30, 2017 - Today? It could be anywhere in the car… even in the **brake** lights. ... Why **would** a **faulty brake** light **switch affect** the **transmission** like this?

Missing: ~~munro~~

munro - Attribution - transmission

The automatic transmission was invented in 1921 by **Alfred Horner Munro** of Regina, Saskatchewan, Canada, and patented under Canadian patent CA 235757 in 1923. (**Munro** obtained UK patent GB215669 215,669 for his invention in 1924 and US patent 1,613,525 on 4 January 1927).

Automatic transmission - Wikipedia

https://en.wikipedia.org/wiki/Automatic_transmission

Figure 2: How Virtual DTs would enable Google search to explain missing keywords

A: *[A faulty brake switch can effect the cruise control .] [If it is,][there should be a code][stored in the engine control module .] [Since it is not an emissions fault ,][the check engine light will not illuminate .] [First of all, watch the tachometer][to see][if engine speed increases 200 rpm][when this happens .] [If it does ,][the torque converter is unlocking transmission .]*

We do not need to know the details concerning how this *Enablement* occurs, we just need evidence that these rhetorical links exist. We could have used semantic linked between entities but for that we would need a domain-specific ontology.

Let us explain how a match between a *Q* and an *A* is facilitated by DTs (Fig. 1). *A* explains a situation and also offer some interpretation, as well as recommends a certain course of action. *A* introduces extra entities which are not in *Q*, and needs to involve background knowledge to communicate how they are related to *E-Q*. We do it by setting a correspondence between *E-Q* and *E-A*, shown by the horizontal curly (red) arcs.

Notice that some entities E_0 in *Q* are *unaddressed*: they are not mentioned in *A*. E_0-*Q* includes {*Engine pump*, *Brake sensor* and *Vacuum*}. It means that either *A* is not fully relevant to *Q* omitting some of its entities E_0 or it uses some other entities instead. Are E_0-*Q* ignored in *A*? To verify the latter possibility, we need to apply some form of background knowledge finding entities E_{img} which are linked to both E_0-*Q* and *E-A*.

It is unclear how *E-A = Torque Convertor* is connected to *Q*. To verify this connection, we obtain a fragment of text from Wikipedia (or another source) about *Torque Convertor*, build *DT*-A_{img1} (shown on the left-bottom of Fig. 1) and observe that it is connected with *Engine* via rhetoric relation of *elaboration*. Hence we confirm that *E-A = Torque Convertor* is indeed relevant for *Q (a vertical blue arc).*

It is also unclear how *E-Q pump* is addressed in *Q*. We find a document on the web about *Engine Pump* and *Vacuum* and attempt to connect them to *E-A*. It turns out that *DT*-A_{img2} connects *Vacuum* and *Engine* via *elaboration*.

Hence the combined *DT-A* includes real *DT-A* plus *DT*-A_{img1} and *DT*-A_{img2} . Both real and virtual DTs are necessary to demonstrate that an answer is relevant by employing background knowledge in a domain independent manner: no offline ontology construction is required.

Search relevance is then measured as the inverse number of unaddressed E_0–*Q* once *DT-A* is augmented with virtual *DT*-A_{img} . This relevance is then added to a default one.

Fig. 2 shows an example how Virtual DT component would improve a web search. Currently, search engines show certain keywords they do not identify in a given search result. However, it is possible to indicate how these keywords are relevant to the search result by finding documents where these unidentified keywords are rhetorically connected with the ones occurring in the query. This feature would naturally improve the answer relevance on one hand and provide an "explainability" for the user on how her keywords are addressed in the answer. In the default search, *munro* is missing. However, by trying to rhetorically connect *munro* with the entities in the question, the Virtual DT approach finds out that *Munro* is an

Algorithm 1 Filtering Algorithm

```
Input: Question
Parameter: Background knowledge B
Output: Most relevant Answer
Build EDT-Q.
Obtain E-Q
Form a query for E-A
Obtain a set of candidate answers As
for each Ac in As do
    Build discourse tree for the
    answer DT-Ac.
    Establish mapping E-Q $\rightarrow$ E-Ac
    Identify $E_0$ -Q.
    Form queries from $E_0$–Q and $E_0$– Ac
    (entities which are not in $E_0$–Q)
    Obtain search results from B for queries
    Build imaginary DTs-Ac.
    Calculate the score = $|E_0|$
end for
Select A with the best score
return A
```

inventor of automatic transmission. DT fragment is shown with rhetorical relation *Attribution*, as well as the Wikipedia source for virtual DT.

4 Question Answering Approach

4.1 Question Answering Filtering Algorithm

Given a *Question*, we outline an algorithm (Algorithm 1) that finds the most relevant *Answer* such that it has as much of *E-Q* addressed by *E-A*, having a source for virtual DTs (background knowledge) *B*.

Discourse trees are constructed automatically using state-of-the-art RST-parser (Surdeanu et.al, 2015).

4.2 Learning on Q/A Pairs

Besides this algorithm, we outline a machine learning approach to classify <*EDT-Q, EDT-A*> pair as correct or incorrect. The training set should include good Q/A pairs and bad Q/A pairs. Therefore a DT-kernel learning approach (SVM TK, Joty and Moschitti, 2014, Galitsky, 2017, 2018) is selected which applies SVM learning to a set of all sub-DTs of the DT for Q/A pair. Tree kernel family of approaches is not very sensitive to errors in parsing (syntactic and rhetoric) because erroneous sub-trees are mostly random and will unlikely be common among different elements of a training set.

Learning framework is available on our GitHub repository.

5 Evaluation

5.1 Experiments on "Convergent" Q/A Datasets

Traditional Q/A datasets for factoid and non-factoid questions, as well as SemEval and neural Q/A evaluations are not suitable since the questions are shorter and not as complicated to observe a potential contribution of discourse-level analysis. For our first evaluation, we formed two convergent Q/A sets.

Yahoo! Answer [1] set of question-answer pairs with broad topics. Out of the set of 140k user questions we selected 3300 of those, which included three to five sentences. Answers for most questions are fairly detailed so no filtering by sentence length was applied to the answers.

Car repair conversations (available online [2]) selected from www.2carpros.com including 9300 Q/A pairs of car problem descriptions vs recommendation on how to rectify them. These pairs were extracted from dialogues as first and second utterances so that a question is one to three sentences and answer is three to six sentences in length. Each dialogue is a comprehensive, cohesive sequence of questions and problem solving recommendations. Most recommendations include a set of conditions to check and actions to perform, not necessarily in the same terms as the problem was formulated. Therefore, traditional search engineering based on keyword statistics performs poorly on this dataset; both semantic and syntactic similarities between Q and A are low.

Source	**Yahoo! Answers**			**Car Repair**		
Search method	**P**	**R**	**F1**	**P**	**R**	**F1**
Baseline (Lucene search engine)	41.8	42.9	42.3	42.5	37.4	39.8
|E-Q ∩ E-A|	53.0	57.8	55.3	54.6	49.3	51.8
|EDT-Q∩EDT-A|	66.3	64.1	65.1	66.8	60.3	63.4
|EDT-Q∩EDT-A +EDT-A_{imgi}|	76.3	78.1	77.2± 3.4	72.1	72.0	72.0± 3.6
SVM TK for <EDT-Q, EDT-A +EDT-A_{imgi}>	83.5	82.1	82.8± 3.1	80.8	78.5	79.6± 4.1
Human assessment of SVM TK for <EDT-Q∩EDT-A +EDT-A_{imgi}>	81.9	79.7	80.8± 7.1	80.3	81.0	80.7± 6.8

Table 1: Evaluation results on convergent Q/A datasets

For each of these sets, we form the positive one from actual Q/A pairs and the negative one from $Q/A_{similar\text{-}entities}$: $E\text{-}A_{similar\text{-}entities}$ has a strong overlap with E-A, although $A_{similar\text{-}entities}$ is not really correct, comprehensive and exact answer. Hence Q/A is reduced to a classification task measured via precision and recall of relating a Q/A pair into a class of correct pairs.

Top two rows in Table 1 show the baseline performance of Q/A and demonstrate that in a complicated domain transition from keyword to

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l

[2] https://github.com/bgalitsky/relevance-based-on-parse-trees/examples/CarRepairData_AnswerAnatomyDataset2.csv.zip.

matched entities delivers more than 13% performance boost. For the baseline we used standard implementation of Lucene search engine based on matching keywords.

The bottom three rows show the Q/A quality when discourse analysis is applied. Assuring a rule-based correspondence between DT-A and DT-Q gives 13% increase over the baseline, and using virtual DT gives further 10%. Finally, proceeding from rule-based to machine learned Q/A correspondence (SVM TK) gives the performance gain of about 7%.

The difference between the best performing *SVM TK for <EDT-Q ∩ EDT-A+EDT-A*$_{img}$*i>* row and the above row is only the machine learning algorithm: representation is the same.

The bottom row shows the human evaluation of Q/A on a reduced dataset of 200 questions for each domain. We used human evaluation to make sure the way we form the training dataset reflects the Q/A relevance as perceived by humans. This is important to confirm, in particular, that the negative dataset includes unsatisfactory answers. For a 1/3 fraction of this dataset we measured Krippendorff's alpha measure for the inter-annotator agreement (two annotators) which exceeds 80%.

To summarize this experiment, the tree kernel learning of virtual discourse trees turned out to be a preferred approach. The contribution of virtual DTs might be insignificant for simpler, shorter, factoid questions when traditional measures of similarity between Q and A work well. However, we demonstrate that involvement of background knowledge via virtual DTs for complex convergent questions requiring entailment is significant.

5.2 Experiments on a Standard Q/A Dataset

We also compare the performance of virtual DT Q/A with neural extractive reading comprehension approaches (Table 2). We made this comparison on the *why?* and h*ow-to?* questions from SQuAD 2.0 (Rajpurkar et al., 2018), the dataset with unanswerable questions which look similar to answerable ones. In total 460 questions were selected.

Deep learning systems can often locate the correct answer to a question in a short text, but experience difficulties on questions for which the correct answer is not stated in the context. Rajpurkar et al. (2018) trained their system on SQuAD and evaluated on the unseen questions. Whereas a deep learning system gets 86% F1 on SQuAD 1.1, it achieves only 66% on SQuAD 2.0 where some questions should not be answered.

Approach	**F1**	**Reference**
BiDaf (Allen NLP) (Gardner et al., 2017)	64.8	Our experiments
DeepPavlov (Burtsev et al., 2018)	61.0	Our experiments
Microsoft Asia (Hu et al., 2018)	74.2	As reported by the authors (full dataset)
SVM TK for <EDT-Q, EDT-A +EDT-A$_{imgi}$>	73.3	Current study

Table 2: Evaluation results on the SQuAD 2.0 dataset

We applied the model trained on Yahoo!Answers and Car Repair to our subset of questions from SQuAD 2.0. Because most unanswerable questions contain entities or entity types which do not occur in text, virtual DT is a good means to handle such cases. At the same time, by the nature of neural learning, it is hard to learn to refuse to answer. The best performance on SQuAD 2.0 for the totality of questions, including much simpler ones than our formed 460 questions dataset, is achieved by (Hu et al., 2018) and exceeds our model by less than 1%.

The model of Hu et al. is specific to Wikipedia pages and the way questions are formulated, whereas our model learns once and for all which discourse structures is correlated with which forms of background knowledge. We believe this performance, achieved by training and testing on the same kind of Q/A dataset is comparable with the results of the general model of the current study with the focus on convergent *why/how to* questions.

6 Conclusions

Answering questions in the domain of this study is a significantly more complex task than factoid Q/A such as Stanford Q/A dataset, where it is just necessary to involve one or two entities and their parameters. To answer a "how to solve a problem" question, one needs to maintain the logical flow connecting the entities in the questions. Since some entities from *Q* are inevitably omitted, these would need to be restored from some background knowledge text about these omitted entities and the ones presented in *Q*. Moreover, a logical flow needs to complement that of the *Q*. The complexity of multi-sentence convergent questions, which was evaluated in, for example (Chen et al., 2017) is way below that one of a real user asking questions in the domains of the current study. Factoid, Wikipedia-targeted questions usually have fewer entities and simpler links between entities than the ones where virtual DT technique is necessary. At the same time, neural network – based approach require a huge training set of Q/A pairs which is rarely available in industrial, practical Q/A domains.

In spite of the great success of statistical and deep learning from a vast set of Q/A pairs, it is still hard to answer questions underrepresented in a training set. Most of the failures of learning approach occur when the user feels that the needed background knowledge is absent. The proposed technique does not require extensive training sets for all Q/A pairs which can be potentially encountered in real time. Instead, we consult necessary texts on demand in real time and avoid maintaining huge training sets on one hand and tackling extensive manually built ontologies on the other hand. Hence we propose a solution to one of the hardest and most sought after problem in AI of how to rely on background knowledge in industrial applications.

Domain-specific ontologies such as the ones related to mechanical problems with cars are very hard and costly to build. In this work we proposed a substitute via domain-independent discourse level analysis where we attempt to cover unaddressed parts of *DT-A* on the fly, finding text fragments in a background knowledge corpus such as Wikipedia. Hence we can do without an ontology that would have to maintain relations between involved entities.

The proposed virtual DT feature of a Q/A system delivers a substantial increase of performance answering complex convergent questions, where it is important to take into account all entities from a question. We observed that relying on rhetoric agreement between Q and A (matching their DTs) improves Q/A F1 by more than 10% compared to the relevance-only focused baseline. Moreover, employing virtual DTs gives us further 10% improvement.

Since we explored the complementarity relation between *DT-A* and *DT-Q* and proposed a way to identify virtual *DT-A* on demand, the learning feature space is substantially reduced and learning from an available dataset of a limited size such as car repair becomes plausible.

Acknowledgements

Section 3 (algorithm to build a discourse representation for the domain knowledge) were written by Dmitry Ilvovsky supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia. Sections 4 and 5.2 (question answering algorithm, experimental investigations) were prepared within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'. The rest of the paper were written and performed at Oracle Corp.

References

A. Louis, A. Joshi, A. Nenkova. 2010. Discourse indicators for content selection in summarization. SIGDIAL, pp. 147–156.

Alexander Hogenboom, Flavius Frasincar, Franciska de Jong, and Uzay Kaymak. 2015. Using rhetorical structure in sentiment analysis. Communications of the ACM 58(7):69–77.

Boris Galitsky, D. Ilvovsky, and S. Kuznetsov. 2015. Rhetoric Map of an Answer to Compound Queries. ACL-2, 681–686.

Boris Galitsky, D. Ilvovsky, and S. Kuznetsov. 2018. Detecting logical argumentation in text via communicative discourse tree. JETAI. pp 637-663.

Boris Galitsky. 2017. Discovering Rhetorical Agreement between a Request and Response. Dialogue & Discourse 8(2) 167-205.

Boris Galitsky. 2014. Learning parse structure of paragraphs and its applications in search. Engineering Applications of Artificial Intelligence. 32, 160-84.

Chali, Y. Shafiq R. Joty, and Sadid A. Hasan. 2009. Complex question answering: unsupervised learning approaches and experiments. J. Artif. Int. Res. 35, 1 (May 2009), 1-47.

Chen D, A Fisch, J Weston, A Bordes. 2017. Reading Wikipedia to answer open-domain questions. https://arxiv.org/abs/1704.00051.

Christina Lioma, Birger Larsen, and Wei Lu. 2012. Rhetorical relations for information retrieval. SIGIR, Portland, OR, pp. 931-940.

D. T. Nguyen and S. Joty. A Neural Local Coherence Model. 2017. ACL. pp. 1320-1330.

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.H., Peters, M., Schmitz, M., & Zettlemoyer, L.S. A Deep Semantic Natural Language Processing Platform. arXiv:1803.07640.

Jansen, P., M. Surdeanu, and Clark P. 2014. Discourse Complements Lexical Semantics for Nonfactoid Answer Reranking. ACL.

Joty, Shafiq R and A. Moschitti. Discriminative Reranking of Discourse Parses Using Tree Kernels. EMNLP 2014.

Joty, Shafiq R, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi- sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

M. Burtsev, A. Seliverstov, R. Airapetyan, M. Arkhipov, D. Baymurzina, N. Bushkov, O. Gureenkova, T. Khakhulin, Y. Kuratov, D. Kuznetsov, A. Litinsky, V. Logacheva, A. Lymar, V. Malykh, M. Petrov, V. Polulyakh, L. Pugachev, A. Sorokin, M. Vikhreva, M. Zaynutdinov. 2018. DeepPavlov: Open-Source Library for Dialogue Systems. ACL-System Demonstrations, p. 122–127. 2018.

M. Sun and J. Y. Chai. 2017. Discourse processing for context question answering based on linguistic knowledge. Know.-Based Syst., 20:511–526, August 2007.

Marir F. and K. Haouam. 2004. Rhetorical structure theory for content-based indexing and retrieval of Web documents, ITRE 2004, pp. 160-164.

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, Dongsheng Li. 2018. Read + Verify: Machine Reading Comprehension with Unanswerable Questions. arXiv:1808.05759

P. Kuyten, D. Bollegala, B. Hollerit, H. Prendinger and K. Aizawa. 2015. A Discourse Search Engine Based on Rhetorical Structure Theory. Advances in Information Retrieva (ECIR). pp 80--91.

Poesio, M., R. Stevenson, B. Di Eugenio, and J. Hitzeman. 2004. Centering: A parametric theory and its instantiations. Computational Linguistics, 30(3):309–363.

Pranav Rajpurkar, Robin Jia, Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. arxiv.org/abs/1806.03822

Pranav Rajpurkar, Zhang, Jian; Lopyrev, Konstantin; Liang, Percy. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. in EMNLP 2016.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. Comput. Linguist. 34, 1 (March 2008), 1-34.

S. Teufel and M. Moens. Summarizing scientific articles: Experiments with relevance and rhetorical status. Computational Linguistics, 28(4):409–445. 2002.

Surdeanu, Mihai, Thomas Hicks, and Marco A. Valenzuela-Escarcega. 2015. Two Practical Rhetorical Structure Theory Parsers. NAACL HLT.

Suwandaratna, N. and U. Perera. 2010. Discourse marker based topic identification and search results refining. In Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on, pages 119–125.

Wang, W., Su, J., Tan, C.L. 2010. Kernel Based Discourse Rela-tion Recognition with Temporal Ordering Information. ACL.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. Text 8(3):243–281.

Yangfeng Ji and Noah Smith. 2017. A Neural Discourse Structure for Text Categorization. ACL 2017.

On a Chatbot Providing Virtual Dialogues

Boris Galitsky[1], **Dmitry Ilvovsky**[2], **and Elizaveta Goncharova**[2]
[1]Oracle Inc. Redwood Shores CA
[2]National Research University Higher School of Economics
boris.galitsky@oracle.com; dilvovsky@hse.ru; egoncharova@hse.ru

Abstract

We present a chatbot that delivers content in the form of virtual dialogues automatically produced from the plain texts that are extracted and selected from the documents. This virtual dialogue content is provided in the form of answers derived from the found and selected documents split into fragments, and questions that are automatically generated for these answers based on the initial text.

1 Introduction

Presentation of knowledge in dialogue format is a popular way to communicate information effectively. It has been demonstrated in games, news, commercials, and educational entertainment. Usability studies have shown that for information acquirers, dialogues often communicate information more effectively and persuade stronger than a monologue most of times (Cox et al., 1999, Craig et al., 2000).

We demo a chatbot that delivers content in the form of virtual dialogues automatically produced from plain texts extracted and selected from documents. Given an initial query, this chatbot finds documents, extracts topics from them, organizes these topics in clusters according to conflicting viewpoints, receives from the user clarification on which cluster is most relevant to her opinion, and provides the content for this cluster. This content is provided in the form of a virtual dialogue where the answers are derived from the found and selected documents split into fragments, and questions are automatically generated for these answers.

Once the proper piece of content is identified, users frequently like to consume it in the form of frequently asked question pages, discussion forums and blogs, rather then formal lengthy document. However, for the majority of knowledge domains, from legal and medical to engineering, most reliable information is only available as documents and web pages. Hence we convert plain documents into dialogues, imitating multiple people conversing on the specific topic of interest.

A virtual dialogue is defined as a multi-turn dialogue between imaginary agents obtained as a result of content transformation. It is designed with the goal of effective information representation and is intended to look as close as possible to a genuine dialogue. Virtual dialogues as search results turn out to be more effective means of information access in comparison with original documents provided by a conventional chatbot or a search engine.

2 Related Systems

Piwek et al. (2007) were pioneers of automated construction of dialogues, proposing Text2Dialogue system. The authors provided a theoretical foundation of the mapping that the system performs from RST structures to Dialogue representation structures. The authors introduced a number of requirements for a dialogue generation system (robustness, extensibility, and variation and control) and reported on the evaluation of the mapping rules.

An important body of work concerns tutorial dialogue systems. Some of the work in that area focuses on authoring tools for generating questions, hints, and prompts. Typically, these are, however, single utterances by a single interlocutor, rather than an entire conversation between two agents. Some researchers have concentrated on generating questions together with possible answers such as multiple choice test items, but this work is restricted to a very specific type of question–answer pairs (Mitkov et al., 2006).

Proceedings of Recent Advances in Natural Language Processing, pages 382–387,
Varna, Bulgaria, Sep 2–4 2019.

Conversion a text into a dialogue is different from the dialogue generation problem; the former is a training set–based foundation for the latter. Response generation for dialogue can be viewed as a source-to-target transduction problem. (Sordoni et al., 2015) rescores the outputs of a phrasal machine translation-based conversation system with a neural model incorporating prior context. Recent progress in sequence-to-sequence models has been leveraged to build end-to-end dialogue systems that firstly applies an utterance message to a distributed vector representation using an encoder, then generates a response from this representation.

(Li et al., 2016) simulate dialogues between two virtual agents, using policy gradient methods to reward sequences that display three useful conversational properties: “informativity”, coherence, and ease of answering.

We measured comparable dialogue effectiveness, properties such as the speed of arrival to a search result, a decision and domain coverage, in the current study.

Dialogue acts are an important source which differentiates between a plain text and a dialogue. Proposed algorithm of virtual dialogues can assist with building domain-specific chatbot training datasets. Recently released dataset, DailyDialog (Li et al., 2017), is the only dataset that has utterances annotated with dialogue acts and is large enough for learning conversation models.

3 Demo Description

3.1 Dialogue Construction from Plain Text

To form a dialogue from text sharing information or explaining how to do things, we need to split it into parts which will serve as answers. Then for each answer a question needs to be formed. The cohesiveness of the resultant dialogue should be assured by the integrity of the original text; the questions are designed to “interrupt” the speaker similar to how journalists do interviews.

We employ a general mechanism of conversion of a paragraph of text of various styles and genres into a dialogue form. The paragraph is split into text fragments serving as a set of answers, and questions are automatically formed from some of these text fragments. The problem of building dialogue from text T is formulated as splitting it into a sequence of answers $A = [A_1 \ldots A_n]$ to form a dialogue

$[A_1, <Q_1, A_2>, \ldots, <Q_{n-1}, A_n>]$,

where A_i answers Q_{i-1} and possibly previous question, and $\cup A_i = T$. Q_{i-1} needs to be derived from the whole or a part of A_i by linguistic means and generalization; also some inventiveness may be required to make these questions sound natural. To achieve it, we try to find a semantically similar phrase on the web and merge it with the candidate question.

The main foundation of our dialogue construction algorithm is Rhetorical Structure Theory (RST, Mann and Thompson, 1988). RST represents the flow of entities in text via Discourse Tree – a hierarchical structure that sets inter-relations between text fragments: what elaborates on what, what explains what, what is attributed to what, what is contradicting what, etc. Such text fragments are called elementary discourse units (EDUs). Most rhetorical relations are binary anti-symmetric, specifying which EDU has more important (nucleus) compared to less important (satellite).

A dialogue is formed from text by the following rule: once nucleus EDU is finished, and before satellite EDU starts, questions against this satellite EDU is inserted. In terms of dialogue flow between a text author and a person asking question, the latter “interrupts” the author to ask his question such that the satellite EDU and possibly consecutive text would be an answer to this question. The question is supposed to be about the entity from the nucleus, but this nucleus does not contain an answer to this question. The person asking questions only interrupts the text author when his question sounds suitable; it does not have to be asked for any nucleus-satellite transition.

Once we split a text into EDUs, we know which text fragments will serve as answer to questions: satellites of all relations. *Elaboration* rhetorical relation is default and *What*-question to a verb phrase is formed. *Background* relation yields another *What*-question for the satellite ‘…*as <predicate>-<subject>’*. Finally, *Attribution* relation is a basis of “What/who is source” question.

A trivial approach to question generation would be to just convert satellite EDU into a question. But it would make it too specific and unnatural, such as ‘*the linchpin of its strategy handled just a* *small fraction of the tests then sold to whom*?’. Instead, a natural dialogue should be formed with more general questions like ‘*What does its strategy handle?’*.

An example of converting a text into a virtual dialogue is shown in Fig. 1. Answers are obtained by splitting text into EDUs, and questions are inserted in text before satellite EDUs. Questions are shown in angle brackets and bolded. Each rhetorical relation in this example such as contrast ranges over a nucleus and a satellite. Each leave of this discourse tree starts with ‘TEXT’ (Fig. 1).

The reader of a virtual dialogue might feel that the interviewer is guessing what the speaker is going to answer for each question. A discourse – tree based approach does not deliver most natural dialogues however it is a systematic method of building ones without distorting the logical flow of answers.

elaboration (LeftToRight)
  attribution (RightToLeft)
<**who provided the evidence of responsibility**>
TEXT: Dutch accident investigators say
    TEXT: that evidence points to pro-Russian rebels as being responsible for shooting down plane.
  contrast (RightToLeft)
    attribution (RightToLeft)
      TEXT: The report indicates
      joint
        TEXT: where the missile was fired from
        elaboration (LeftToRight)
          <**what else does report indicate?**>
          TEXT: and identifies
          TEXT: who was in control and pins the downing of the plane on the pro-Russian rebels .
    elaboration (LeftToRight)
      attribution (RightToLeft)
        TEXT: However , the Investigative Committee of the Russian Federation believes
        elaboration (LeftToRight)
          TEXT: that the plane was hit by a missile from the air
          <**where was it produced?**>
          TEXT: which was not produced in Russia .
      attribution (RightToLeft)
        TEXT: At the same time, rebels deny <who denied about who controlled the territory>
        TEXT: that they controlled the territory from which the missile was supposedly fired .

Fig. 1. A discourse tree for a paragraph of text with questions formulated for satellite EDUs as answers

3.2 System Architecture

System Architecture for building a dialogue from text is shown in Fig. 2. Each paragraph of a document is converted into a dialogue via building a communicative discourse tree for it and then building questions from its Satellite Elementary Discourse Units. Current chatbot is a development of the previously built tool that conducted task-oriented conventional dialogues (Galitsky et al., 2017).

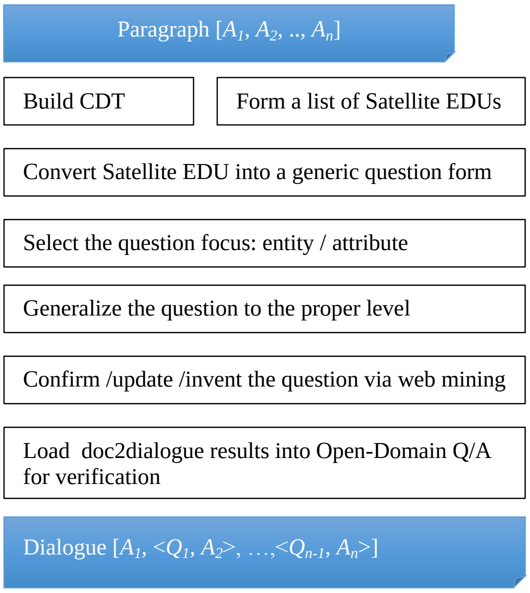

Paragraph [A_1, A_2, .., A_n]
Build CDT
Form a list of Satellite EDUs
Convert Satellite EDU into a generic question form
Select the question focus: entity / attribute
Generalize the question to the proper level
Confirm /update /invent the question via web mining
Load doc2dialogue results into Open-Domain Q/A for verification
Dialogue [A_1, <Q_1, A_2>, …,<Q_{n-1}, A_n>]

Fig. 2. System architecture

4 Evaluation of Effectiveness

Evaluating the effectiveness of information delivery via virtual dialogues, we compare the conventional chatbot sessions where users were given plain-text answers, and the ones where users were given content via virtual dialogues.

We present the results on comparative usability of conventional dialogue and virtual dialogue. We assess dialogues with respect to following usability properties.

The speed of arriving to the sought piece of information. It is measured as a number of iteration (a number of user utterances) preceding the final reply of the chatbot that gave an answer wanted by the user. We measure the number of steps only if the user confirms that she accepts the answer.

The speed of arriving to a decision to commit a transaction such as purchase or reservation or product selection. A user is expected to accumulate sufficient information, and this information such as reviews should be convincing enough for making such decision;

We also measure **how many entities** (in linguistic sense) were explored during a session with the chatbot. We are interested in how thorough and comprehensive the chatbot session is, how much a user actually learns from it. This assessment is sometimes opposite to the above two measures but is nevertheless important for understanding the overall usability of various conversational modes.

Table 1. Evaluation of comparative effectiveness of conventional and virtual dialogues

	Conventional dialogues			Virtual dialogues		
	# of iterations till found	# iterations till decision	Coverage of exploration # of entities	# of iterations till found	# iterations till decision	Coverюexploration # of entities
Conventional only	4.6	6.3	10.8	-	-	-
Virtual only	-	-	-	4.1	6.0	13.7
Conventional followed by virtual	4.0	5.7	7.6	6.1	11.3	15.1
Virtual followed by conventional	5.6	7.1	12.3	3.7	7.0	11.5

We do not compare precision and recall of search sessions with either dialogue mode since the same information is delivered, but in distinct modes.

In the first and second rows, we assess the stand-alone systems. One can observe that virtual dialogues take less iteration on average for information access and about the same number of iterations for decisions as conventional dialogues do.

Notice that the bottom row, the chat scenario proceeds from right to left. In the bottom two rows, we observe the usability of the hybrid system. When a conventional dialogue is followed by a virtual one, a lower portion of users is satisfied by the first step in comparison to the inverse architecture, where virtual is followed by conventional.

5 Sample Chatbot Session

We present an exploratory session that combines information delivery in the form of traditional textual answers (conventional dialogue) and also in a form of a virtual dialogue.

Bot>*ask a new question*
User>**advantages and new features of 5G**
Bot>exploring….

I believe these are the main topics of your query: is that what you meant? Please select demonstrating the benefits of the technology[1]. wide range of people from student'[2]. 5g new radio nr specification'[3]. … next generation mobile networks alliance[4]. the next stage in mobile technology[5].

User> **next stage in technology (or [5])**

Bot> Put simply, it’s the next stage in mobile technology. It follows 4G and 4G LTE and builds on what they offer, delivering everything at a faster speed. …

Are you OK with this answer? yes/more/no/specify [different topic] / reduce search to web domain / virtual dialogue

User > **virtual dialogue**
Bot> This is what other people are chatting about your topic:

User1>**What Are the Benefits of Technology?**
Bot1> If you hopped over to Ericsson, you'd find a "5G avenue" with multiple exhibits demonstrating the benefits of the technology…
User1>**How are these benefits leveraged by new tablets?**
Bot1>Samsung kicked off its press conference not with its new tablets, but 5G is the next wireless network in telecom industry …
User2>**Are the features right for them?**
Bot1>It is important for people to start learning about 5G technology now so that they can decide if the features are right for them...
User2 >**Which are the dual 4G VoLTE sim smartphones in India?**
Bot1> The current 4G VoLTE smartphones with dual SIM slots that are available in the Indian markets may support 4G on single or both SIM slots.

Fig. 3. Chatbot session with conventional and virtual dialogue

The dialogue starts from the user question, ‘*advantages and new features of 5G’*. The chatbot consults the sources (in this case, public URLs) and extracts the content from each page (or documents) expected to be relevant for the query. In this example, seven URLs were processed, from domain-specific to general knowledge portals like Quora.com. The chatbot forms the list of topics extracted from these search results so that the user might select one of his interest.

Once the chatbot forms the topics for clarification of the user search intent, it shows them as a list.

The user selects his topic of interest and requests a specific answer via the topic number of the topic expression. Once the answer is read, there are multiple options:

- navigate to the next answer from the chatbot list;
- navigate to a specific answer from the chatbot list;
- reject this answer and attempt to reformulate the query;
- reduce search to a specified web domain (such as quota.com, for example);
- proceed in the same direction to more search results in the form of a virtual dialogue;
- accept the answer and conclude the session.

The user selects the last option and the chatbot builds a virtual dialogue. It is a conversation between an imaginary people but the topic stays the same, matching the original query. Virtual dialogues are shown in frames. As long as an imaginary chatbot responds to the same person, the dialog is intended to stay cohesive; coreferences in the follow-up questions are maintained. The main dialogue can be viewed as a one in the meta-level, and the object-level dialogue is naturally embedded into the meta-level one.

Now the user can either browse the built virtual dialogue or search it to find a fragment of conversation which is relevant to the user current exploration intent. The user now types the query ‘*Are the features right for me?*’ and gets directed to the virtual dialogue fragment where some other users are discussing if the technology is ‘*right for them’*. The search matches the query either against the fragments of an original text, generated questions or both.

6 Conclusions

We proposed a novel mode of chatbot interaction via virtual dialogue. It addresses sparseness of dialogue data on one hand and convincingness, perceived authenticity of information presented via dialogues on the other hand. We quantitatively evaluated improvement of user satisfaction with virtual dialogue in comparison to regular chatbot replies and confirmed the strong points of the former. We conclude that virtual dialogue is an important feature related to social search to be leveraged by a chatbot.

Chatbot demo videos (please, check **10 min** video) and **instructions** on how to use it are available at https://github.com/bgalitsky/relevance-based-on-parse-trees in the “**What is new?**” section.

7 Acknowledgements

The article was prepared within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project '5-100'.

References

Mann, William and Sandra Thompson. 1988. *Rhetorical structure theory: Towards a functional theory of text organization. Text* - Interdisciplinary Journal for the Study of Discourse, 8(3):243–281.

Joty, Shafiq R, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi- sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Galitsky, B, Ilvovsky, D. and Kuznetsov SO. 2015. Rhetoric Map of an Answer to Compound Queries. ACL-2, 681–686.

Kipper, K. Korhonen, A., Ryant, N. and Palmer, M. 2008. A large-scale classification of English verbs. Language Resources and Evaluation Journal, 42, pp. 21-40.

Cox, R J. McKendree, R. Tobin, J. Lee, and T. Mayes. Vicarious learning from dialogue and discourse: A controlled comparison. Instructional Science, 27:431– 458, 1999.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. InProc. of NAACL-HLT, May–June.

Craig, S, B. Gholson, M. Ventura, A. Graesser, and the Tutoring Research Group. Overhearing dialogues and monologues in virtual tutoring sessions: Effects on questioning and vicarious learning. International Journal of Artificial Intelligence in Education, 11:242–253, 2000.

Li Y, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. arXiv preprint arXiv:1710.03957.

Piwek, Paul; Hernault, Hugo; Prendinger, Helmut and Ishizuka, Mitsuru (2007). T2D: Generating Dialogues Between Virtual Agents Automatically from Text. Lecture Notes in Artificial Intelligence, Springer, Berlin Heidelberg, pp. 161–174.

Mitkov R, L. A. Ha, and N. Karamanis. A computer-aided environment for generating multiple-choice test items. Natural Language Engineering: Special Issue on using NLP for Educational Applications, 12(2):177–194, 2006.

Boris Galitsky and Dmitry Ilvovsky. 2017. Chatbot with a discourse structure-driven dialogue management. EACL System Demonstrations.

Assessing Socioeconomic Status of Twitter Users: A Survey

Dhouha Ghazouani[1], **Luigi Lancieri**[2], **Habib Ounelli**[1] and **Chaker Jebari**[3]

[1]**Faculty of Sciences of Tunis, University Campus, Tunis 1060, Tunisia**
(dhouha.ghazouani,habib.ounelli)@fst.utm.tn
[2]**Univ.Lille, Research Center in Signal and Automatic Computing of Lille, F-59000 Lille, France**
luigi.lancieri@univ-lille.fr
[3]**Information Technology Department, Colleges of Applied Sciences, Ibri, Oman**
jebarichaker@yahoo.fr

Abstract

Every day, the emotion and opinion of different people across the world are reflected in the form of short messages using microblogging platforms. Despite the existence of enormous potential introduced by this data source, the Twitter community is still ambiguous and is not fully explored yet. While there are a huge number of studies examining the possibilities of inferring gender and age, there exist hardly researches on socioeconomic status (SES) inference of Twitter users. As socioeconomic status is essential to treating diverse questions linked to human behavior in several fields (sociology, demography, public health, etc.), we conducted a comprehensive literature review of SES studies, inference methods, and metrics. With reference to the research on literature's results, we came to outline the most critical challenges for researchers. To the best of our knowledge, this paper is the first review that introduces the different aspects of SES inference. Indeed, this article provides the benefits for practitioners who aim to process and explore Twitter SES inference.

1 Introduction

The ability to identify the socioeconomic status of social media users accurately is beneficial for the individual scale as well as the societal one. This field starts to be a well-explored research domain. The difficulty to identify the socioeconmic status of authors and the lack of explicit personal information have brought with them some challenge for computer scientists.

Nowadays, Twitter's monthly active members exceed 300 millions. These members generate over 500 million conversations (tweets) daily [1]. These conversations are short text messages including a maximum of 140 characters (recently extended to 280). Indeed, this shortage of characters leads to unstructured and noisy texts to the point that natural language processing (NLP) tools cannot manage successfully (Ritter et al., 2011). Moreover, more deduction is required to detect the underlying features of Twitter users. In this regard, researchers and specialized centers will explore and analyze the available demographic information of Twitter users. The results provided by the Pew Research Center (Smith and Brenner, 2012), a subsidiary of the Pew Charitable Trust, show that the majority of Twitter users in the United States are young, with high educational level and exposing a bigger political interest. Authors concluded that focusing on a community characterized by high level of involvement in societal issues (Li et al., 2015) could be fruitful. In a first study (Preoţiuc-Pietro et al., 2015a) prove that language use in social media is an indicator of user's occupational class. In a second study, (Preoţiuc-Pietro et al., 2015b) provide a comparison between income and psycho-demographic traits of Twitter users; among the results of this research they concluded that the rich users expose less emotional status but more neutral content, expressing anger and fear, but less surprise, sadness, and disgust. Recently, (Flekova et al., 2016) found that the writing style can also indicate the income of the users. The higher income is an indicator of education and conscientiousness. Moreover, (Volkova and Bachrach, 2016) concluded that the highly educated users have a stronger tendency to express less sadness and are likely to show more neutral opinions.

User's socioeconomic status (SES) is the most

[1] https://about.twitter.com/company

important predictors of a person's morbidity and mortality experience (Kitagawa and Hauser, 1973; Marmot et al., 1987). The significant impact of SES on public health renders its definition and measurement of critical importance. When the SES is low, it does not only involves poverty and poor health, but it also affects the educational achievements hence the whole society. Thus, the research of (Morgan et al., 2009) finds that children from low-SES households develop a slow academic behavior than children belonging to higher SES groups. For these reasons, marketing campaigns, as well as economic and sociological studies, have found it interesting to determine the socioeconomic status of particular persons.

This article is going to be divided into six sections. After the introduction, section 2 will discuss the metrics of socioeconomic status used with Twitter data. Section 3 will discuss SES indicators and its features. Section 4 will examine the different techniques employed in SES inference and section 5 will present the data collection and analysis process. Section 6 is going to be the conclusion for this article opening the horizons for further discussions.

2 Evaluation of Socioeconomic Status

Socioeconomic status (SES) can be defined as one's possession of social, financial, cultural, and human capital resources. Parental and neighborhood properties are considered as additional components (Cowan et al., 2012). We can also note that SES is a complex unit of measurement of a person's economic and sociological standing, like for instance, his prestige, power or else his economic well-being (Hoff et al., 2002; Oakes and Rossi, 2003). Consequently, one can conclude that the SES is a complex measure of evaluation that differs from a research to another because it takes into account the work experience, the economic position, or the social status.

The concept of the SES detection in literature goes back to the beginning of the 20^{th} century (Chapman and Sims, 1925). In the 20^{th} century the evaluation of SES was based on questions like: How many years did your father go to school?", Do you have a telephone?" or Do you work out of school hours?". Currently, there is an agreement that SES is influenced by three significant factors: the cultural (comprised of skills, capacities, and knowledge), the social (social network combined with the status and power of the people in that network) and the material capital (Jones et al., 2007). Similarly, the primary metrics of the SES are education, occupation, income levels and wealth or lifestyle (Van Berkel-Van Schaik and Tax, 1990; White, 1982).

People are usually divided into groups according to these metrics, from the least advantaged to the most advantaged, medium, or high SES. **Education** is one of the widely used indicator and it is considered by many to be the canonical element of SES because of its influence on later income and occupation (Krieger et al., 1997). This index can be defined by two dimensions: the field of education and the level at which the education was followed. **Income** reflects spending power, housing, diet, and medical care. **Occupation** measures like prestige, responsibility, physical activity, as well as work exposures. The occupational status influences the social capital of individuals and it strengthen the connection with more professional people enjoying wealth and power. Similarly, education indicates skills requisite for acquiring a positive social, psychological, and economic resources (Antonovsky, 1967). Likewise, social classes are measurements that, like SES, aim to locate ones position in the social hierarchy. classes are social categories sharing subjectively-salient attributes used by people to rank those categories within a system of economic stratification" (Wright and Ritzer, 2003). By refering to the definition presented by (Wright and Ritzer, 2003), classes refer to how people are objectively located in distributions of material inequality".

Before the rise of social networks, different studies have looked into other data sources from various domains, like internet browsing behaviors, written texts, telephone conversations, real-world mobile network and communication records.

- (French, 1959): introduced the relationship between different measures of 232 undergraduate students and their future jobs. This work concluded that occupational membership could be predicted with the use of variables such as the ability of persons in using mathematical and verbal symbols, the social class of family and the personality components.

- (Schmidt and Strauss, 1975): have also designed the relationship between the types of occupation and the particular demographic attributes such as gender, race, experience, education, and location. Their study identified biases and the types of discrimination that can possibly exist in various types of occupations.

The recent excessive use of online social media and the user-generated content in microblogging platforms such as google+, Facebook, or Sine Weibo [2] has allowed the study of author profiling on an unprecedented scale.

- (Li et al., 2014): proposed a framework for assessing the user's features on Twitter using Google+ API. They constructed a publicly available dataset using distant supervision. They submitted their model on three user profile attributes, i.e., Job, Spouse and Education.

- (Zhong et al., 2015): investigated the predictive power of location check-in, extracted from points of interest of Sina Weibo. In order to determine the demographic attributes of the users such as education background (university and non-university), marital status (single, courtship, in love or married) using human mobility as an informative and fundamental user behavior. They developed a comprehensive location to profile (L2P) framework to detect temporality, spatiality, and location knowledge at the same time.

- (Sullivan et al., 2018) has recently reported that Facebook has patented technology that utilizes a sample decision tree to determine its users' social class. Decision tree uses as an input information about a user's demographic information, device ownership, internet usage, household data, etc. The output provides a probability that the user belongs to a given socioeconomic class: working class, middle class or upper class.

Recent studies tackled the inference of socioeconomic characteristics of Twitter users.

- (Lerman et al., 2016): analyze a large corpus of geo-referenced tweets posted by social media users from US metropolitan areas. They measure emotions expressed in the tweets posted from a particular area with the inference of socioeconomic characteristics. They collect Twitter accounts which users are located in Los Angeles. Concerning the sentiment analysis, they used SentiStrength [3]. The study shows that people with higher incomes are associated with weaker social ties.

- (Quercia et al., 2012): treat the relationship between sentiment expressed in tweets and the community socioeconomic well-being. In their research, they collect Twitter accounts which users are located in London. Concerning the sentiment analysis, they used word count technique and the maximum entropy classifier. Socio-demographic data obtained from Index of Multiple Deprivation scores (composite score based on income, employment, education, health, crime, housing, and the environmental quality for each community) of each of the 78 census areas in London.

3 SES Features and Indicators

The quality of features influences the value of a machine learning pattern from which it originates. Microblogging platforms offer a different number of potential features. Different traditional text-based corpora features are used to explore the relationship between these characteristics. Different types of indicators can help infer the SES of Twitter users used over years. This idea is going to be developed later in the article.

3.1 Message Content

Twitter message text represents the backbone of most research works within the field of SES inference as this helps to understand the context of messages themselves. The messages of the social media platform include abbreviations and non-standard formulation as there is no precise rule of writing since most of the tweets are sent via mobile phones.

In his thesis, (Mentink, 2016) used Bag-of-Words to analyze the discussed topics of users. (Preoţiuc-Pietro et al., 2015b) used clustering algorithms to build a list of most frequent unigrams and then they reached their vector representations, consequently using Word2Vec model to compute dense word vectors (grouping words into clusters or topics). While (Lampos et al., 2016) applied spectral clustering to derive clusters of 1-gram that

[2] http://www.weibo.com/signup/signup.php

[3] http://sentistrength.wlv.ac.uk/

capture some potential topics and linguistic expressions, (Preoţiuc-Pietro et al., 2015a) used Normalized Pointwise Mutual Information (NPMI) to compute word to word similarity, then applied singular value decomposition (SVD) to obtain an embedding of words into a low-dimensional space. A good approach of content analysis would take into consideration all possible instances of SES indicators being expressed within the message. For most studies, the use of message content aims at inferring morphological characteristics and language use.

(Barberá, 2016) used the emoji characters as features (bag-of-emoji) and the author used word counts as another features (bag-of-words) with the application of TF-IDF transformation. In order to obtain a robust result, the most successful techniques used employ message content initially, alongside other features.

3.2 User Profiles

Although it must be admitted that in creating a new Twitter account, personal information are limited, however, they can give beneficial insights for the SES of particular users. Users' profiles contain a different number of metadata such as the user's biography, followers, name, and location. The expectation is that a user's biography offers an important source of demographic data. However, Twitter users' biography is left empty for 48% of users, and others do not supply good-quality information (Culotta et al., 2016). (Preoţiuc-Pietro et al., 2015a) use the profile information of the account to capture users with self-disclosed occupations by annotating the user description field. (Lampos et al., 2016) use also profile description field of UK Twitter users to search for occupation mentions. In order to infer the user's socioeconomic status, most studies use description field and attempt to search for related information given by a particular user. These data are also useful in order to validate other SES features inferred from tweet messages.

3.3 Social Network Relations

The followers of a user represent a good indicator of their SES. Following reciprocal relationship can provide evidence of strong user connection. Some indicators can group regular exchanges of messages or frequent mention to names in messages. The number of tweets, mentions, links, hashtags and retweets, the number of followers, friends and the ratios of tweets to retweets are considered as statistical features. (Lampos et al., 2016) use these features to compile a set of latent topics that Twitter users were communicating. (Culotta et al., 2016) use the Twitter REST API and *followers/ids* request to sample many followers for each account, and the results are ordered with the most recent following first. And with the same methods, they use *friends/ids* API request to collect a list of friends. The example of (Barberá, 2016) best illustrates this idea it enables to overcome the collection of information about the entire network of a particular user that is costly and requiring multiple API calls, focuses on verified accounts. (Ikeda et al., 2013) use a community-based method with the extraction of the community from follower/followee relations followed by estimation of the demographics of the extracted communities. The demographic category of each community group is estimated using text-based method and the use of Fast Modularity Community Structure Inference Algorithm. Some studies assume that people within a given social class tend to have similar lifestyles using their income levels and common experience. Their interaction is called homophily. In the same context (Aletras and Chamberlain, 2018) use the information extracted from the extended networks of Twitter users in order to predict their occupational class and their income. They demonstrated that user's social network and their language use are complementary.

3.4 Spatial Information

The majority of smartphones are now equipped with Global Positioning System (GPS) functions and they work with geo-satellites which accurately infer the user's location with latitudes and longitudes coordinates. This would be an optional field for a particular user to enable due to their privacy choice. This indicator is very helpful when the person is mobile and usually updates their location profile. (Bokányi et al., 2017) obtained 63 million of Twitter geolocated messages from the area of the United States and assigned a county to each tweet. Once aggregated, daily tweeting activity allows to measure human activities and constitutes an important socioeconomic indicator whether a particular user is employed or not. In order to build a social class dataset, some studies attempt to show that the wealthier the place, the richer the users who usually visit it. (Miranda Filho et al., 2014) used the lifestyle and the wealth of neighborhood people typically visit to label Brazilian users into various social classes. Then, they utilized Foursquare to label places according to the wealth of the neighborhood. They selected users who had at least one Foursquare interaction (Foursquare interactions include check-in (the user told a friend he/she was at a given place), tips (the user posts tips and opinion about a given place) and mayorship (title given to the most frequent user in a given location in the past 60 days). (Zhong et al., 2015) investigate the predictive power of location and the mobility to infer users' demographics with the use of location to profile (L2P) framework. The data crawling module accumulates user profiles and location check-in with corresponding information on Sine Weibo.

3.5 Temporal Information

Twitter enables researchers to analyze human activities during the 24 hours of the day because they are biologically bound to exhibit daily periodic behavior. In this context (Bokányi et al., 2017) aggregate monday to friday relative tweeting activities for each hour in each US County to form an average workday activity pattern, assuming that the activity patterns form a linear subspace of the 24-hour ”time-space”. This study shows that this measure correlates with county employment and unemployment rates in relation to lifestyles connected to regular working hours. The relationship between daily activity patterns and employment data can be captured using Twitter data.

3.6 Demographic Attributes

Some researchers attempted to include demographics as features. Age, for example, has a vital role in income prediction. Old people earn significantly more than young ones. Higher age leads to, on average, more work experience and education, which is translated into higher income. (Flekova et al., 2016) explored the relationship between stylistic and syntactic features, authors' age, and income, to conclude that the hypothesis of numerous feature type writing style and age use is predictive of income.

4 Inference Methods for SES Evaluation on Twitter

Different techniques have been used in the past and are being employed now to improve the accuracy of SES inference methodologies and algorithms. This burgeoning field lends techniques ranging from different areas of study involving machine learning, statistics, natural language processing to regression models. Various methods achieved different levels of success. The effectiveness and granularity levels produced by these methods continue to be improved.

Most recent researchers use a three-step methodology to infer the SES. First, they collect available information about a number of Twitter users. Secondly, they develop the classification method using additional data (number of followers, the content of tweets). And finally, they classify users who do not provide any concrete information according to SES. (Preoţiuc-Pietro et al., 2015a) for example, extracts occupation information from Twitter user profiles and uses text analysis to categorize users into occupational classes.

In general, a common approach to demographic inference is supervised classification, from a training set of labeled users, a model is fit to predict user features from the content of their writings. In other words, inferring user characteristics is framed as a predictive task validated on held-out data. This is done by establishing regression or classification methods.

4.1 Regression Methods

Various techniques for the inference of SES of Twitter users have been adopted from data mining and machine learning techniques. Some studies used the linear regression method, others used non-linear regression method and a third party used a hybrid approach that combines both linear and non-linear methods. A standard non-linear method does not inform which features are the most important in the predictive task. Then, the interpretability of linear methods allows performing an extensive qualitative and quantitative analysis of the input features. (Flekova et al., 2016) used both linear with Elastic Net regularization methods and non-linear with Support Vector regression together with an RBF kernel method. The authors found that machine learning regression methods can be used to predict and analyze user's income. (Lampos et al., 2016) used a non-linear generative learning approach, which consists of Gaussian Process (GP) and Kernel, to classify Twitter users according to SES as having upper, middle or lower level. Further, in (Preoţiuc-Pietro et al., 2015b), the authors used similar methods to study the user behavior and its power to predict income. It is important to note that GPs is a Bayesian nonparametric statistical framework that formulates priority functions. (Hasanuzzaman et al., 2017) used linear and non-linear methods. The linear method is a logistic regression with Elastic Net regularization. In order to capture the non-linear relationship between a user's temporal orientation and their income, the authors used GP for regression. (Culotta et al., 2016) used a regression model for the prediction in order to understand the demographics of users. Due to the high dimensionality of features, the authors used elastic net regularization. Since each output variable consists of subject categories of demographic characteristics. They used a multitask variant of the elastic net to ensure the same features as selected for each category.

4.2 Classification Methods

(Mentink, 2016) employed two different approaches to classify the users in the dataset. The first is named the individual approach, it determines the performing classifier per feature group and consequently combines them via a soft-voting ensemble method. The second is named the combined approach, it calculates the performance scores for all possible combinations of classifiers and their respective ensemble (also via soft-voting). The author used Logistic Regression, Support Vector Machines, Naive Bayes and Random Forest algorithms. The author runs the algorithm to determine what occupation and what education-level label should be given to a particular user, to overcome the data imbalance, noise and bias, (Chen and Pei, 2014) used a typical imbalance classification approach which uses multiple classifier systems (MCS) and a sampling method which is a class-based random sampling method an extension of random under-sampling. The objective is to classify users according to their occupation. (Miranda Filho et al., 2014) evaluated a large number of classifiers using their WEKA version to generate classification models, including multinomial Naive Bayes (MNB), Support Vector Machine (SVM), and Random Forest. As MNB is more efficient than other algorithms, the authors used this method to infer social class for each particular user.

5 Tweet Gathering and Analysis

Messages on Twitter are publicly accessible in the online domain and can be gathered for study purposes. This availability makes Twitter an efficient tool in retrieving and analyzing public messages by allowing its users to become social sensors within the population.

5.1 Data Corpuses and Ressources

The corpus size of tweets grouped have varied from relatively small datasets to as large as three billion tweets (Mentink, 2016). The time span of the data collected was usually in the range of a few weeks to a couple of months, and sometimes a year. Table 1 shows some datasets and their sizes over the past years. First, the REST API is helpful for gathering particular user tweets, allowing the backtracking of their timeline. For example, to collect their most recent 3.200 tweets. Second, the streaming API that manages the tweets as they are being broadcast would only be able to receive 1% of the Firehose. Twitter data partners furnish a premium service that supplies messages covering a longer duration as well as 100% access to the Firehose. (Preoţiuc-Pietro et al., 2015a) created a publicly available data-set [4] of users, including their profile information and historical text content as well as a label to their occupational class from the "Standard Occupational Classification" taxonomy.

This public available dataset used by several researchers containing a group of $5,191$ users in total. However, the extraction of social network information of some accounts are not allowed. These accounts may have been annulled or become private. For example (Aletras and Chamberlain, 2018) reported results of $4,625$ users, from the original subset, that are still publicly available. Various studies (Preoţiuc-Pietro et al., 2015b; Lampos et al., 2016; Preoţiuc-Pietro et al., 2015a; Flekova et al., 2016) in the dataset creation mapped Twitter users to their income or their job title using standardized job classification taxonomy. The Standard Occupational Classification (SOC) is a UK-governmental system developed by the Office of National Statistics (ONS) for listing and grouping occupations. Jobs are organized hierarchically based on skill requirements and content.

(Culotta et al., 2016) mapped Twitter users according to their educational level (No College, College, Grad School) and other traits using Quantcast.com, an audience measurement society that tracks the demographics of users of millions of websites. The estimated demographics of a large number of sites are publicly accessible through the

[4] https://sites.sas.upenn.edu/danielpr/data

References	Corpus size	Period Covered	Corpus Origin
(Miranda Filho et al., 2014)	15.435 Users	Sep'13-Oct'13	Brazilian
(Preoţiuc-Pietro et al., 2015b)	10.796.836	Aug'14	US
(Barberá, 2016)	1.000.000.000	Jul'13-May'14	US
(Lampos et al., 2016)	2.082.651	Feb'14-Mar'15	US
(Mentink, 2016)	3.000.000.000	Nov'14-Oct'15	Dutch
(Hu et al., 2016)	9.800 Users		US
(Bokányi et al., 2017)	63.000.000	Jan'14 and Oct'14	US
(van Dalen et al., 2017)	2.700.000	Sep'16	Dutch
(Abitbol et al., 2018)	170.000.000	Jul'14-May'17	French
(Levy Abitbol et al., 2019)	90.369.215	Aug'14-Jul'15	French

Table 1: Datasets and Collection Periods of Some Studies.

use of searchable web interface. For each variable, Quantcast gives the expected percentage of visitors to a website with a given demographic.

5.2 Results and Metrics

The conclusions reached by different studies have been significantly improved over time with regards to increased accuracy and other measurements. Table 2 shows some techniques and their results over the past years. This has been driven by improvements in algorithms and inclusion of more useful features. It is important to note that the effectiveness and the reliability of occupation representativeness increase when estimating profession, using non-standard and out-of-vocabulary (OOV) occupation names. In this context, to overcome the limitation of the work of (Sloan et al., 2015) and (Mac Kim et al., 2016), (Kim et al., 2016) builded a machine learning model attempts to capture linguistically noisy or open-ended occupations in Twitter. This induces in more reliable occupation representativeness.

Different approaches have been introduced to compare the performance and results of the methods. They include accuracy and use of two other standard metrics: Pearson's correlation coefficient and Mean Absolute Error (MAE). To validate the effectiveness of the approaches against different baselines, the k-fold cross validation has been well utilized for precision, recall and F-measure: the standard metrics for classification methods.

Over time, accuracy levels of results have continued to be improved starting from 2013 when the inference of users' occupation was used. taking into consideration other information such as Twitter links, friends, user tweets, profiles, and other metadata associated with the message. Furthermore, with the adoption of various features such as user profile features, users psycho-demographic features, or user emotion features, accuracy has improved with the recent studies of (Mentink, 2016; Lampos et al., 2016) achieving a 75% accuracy.

6 Conclusion and Future Prospectives

The study of socioeconomic status inference is one of the most active field of information retrieval. Such works are positioned at a crossroads of multiple disciplines.

Different studies that introduced the inference of hidden user characteristics (Al Zamal et al., 2012; Miranda Filho et al., 2014; Volkova et al., 2015) are salient in the field. The results of these works are not only of interest to statistics agencies but also necessary for studies in the social science (targeted advertising, personalized recommendations of user posts and the possibility of extracting authoritative users (Pennacchiotti and Popescu, 2011)). It is important to introduce the role of SES in politics such as the works of (Barberá and Rivero, 2015), (Burckhardt et al., 2016), (Kalsnes et al., 2017), (Vargo and Hopp, 2017) and (Brown-Iannuzzi et al., 2017). Twitter is increasingly considered as politically transformative communication technology that allows citizens and politicians to connect, communicate, and interact easily (Chadwick, 2006). The flaw in previous studies of political behavior using Twitter data is the lack of information about the sociodemographic characteristics of individual users. Policy makers have recently suggested introducing well-being community which will help governments do a better job at directing public policy towards promoting quality of life.

The inference of SES is an ambitious problem as it may belong to a combination of environmen-

References	Technique	Accuracy (%)	Class
(Ikeda et al., 2013)	Hybrid Method	71.60	Occupation
(Siswanto and Khodra, 2013)	Machine Learning	77.00	Occupation
(Miranda Filho et al., 2014)	Machine Learning	73.00	Social Class
(Preoţiuc-Pietro et al., 2015a)	Gaussian Process	52.70	Occupation
(Mentink, 2016)	Hybrid Method	75.00	SES
(Lampos et al., 2016)	Gaussian Process	75.00	SES
(Poulston et al., 2016)	SVM Classifier	50.47	SES
(van Dalen et al., 2017)	Logistic Regression	72.00	Income
(Hasanuzzaman et al., 2017)	Supervise Learning	74.40	Income
(Aletras and Chamberlain, 2018)	Gaussian Process Regression (GPR)	50.44	Occupation

Table 2: Results and Techniques Used Over the Past Years.

tal variables and individual characteristics. Some of these characteristics can be easier determined like gender or age while others, are sometimes complicated with privacy issues and relying to some degree on self-definition, are harder to determine like occupation, ethnicity, education level or home location. Neverthless, there are many challenges in the inference of SES for Twitter users. Manual classification and data sampling are time-consuming, hard process and not scalable. Models are learned by referring to a datasets which were manually labeled using Amazon Mechanical Turk at a high monetary cost. Another issue is that people often misrepresent themselves on various online social platforms. This can lead to false data interpretations which as a result can affect the accuracy of the research. Automated detection tools are based on the supposition that users will introduce information on their demographic background through profile information or metadata. While it is not possible to expect that all users do this, those who did were a random group of the Twitter population, then we would not expect to discover conflicts in prevalence rates for sociodemographic characteristics (Sloan et al., 2015). Another problem is that Twitter data cannot represent all the populace as discussed previously. (Sloan, 2017) treated the issue of using human validation to find the accuracy of methods applying profile data to assign users to occupational groups and, he deduced that this process could provide misclassifications due to users reporting their hobbies and interests rather than their actual occupations (e.g, writer, artist). Another limit is that deriving income statistics from job labels is not a suitable method.

Given the findings presented above, the following are important issues to address in future Twitter socio-demographic inference studies. First, there is a need to look at the relationship between a user's actual demographic characteristics and how demographic categorization tools classify that user as a function of how profile information is presented and a virtual identity constructed. To conclude, there is a need to link Twitter profiles and survey data. Researchers can start theorizing better working machine learning models to improve accuracy and scalability. In addition, the methodologies used in different research projects can be coupled to increase efficiency. Another purpose for future research projects is to construct a less human effort, low computational cost and focus on the construction of a stronger evaluation framework.

References

Jacob Levy Abitbol, Márton Karsai, Jean-Philippe Magué, Jean-Pierre Chevrot, and Eric Fleury. 2018. Socioeconomic dependencies of linguistic patterns in twitter: A multivariate analysis. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, pages 1125–1134.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of twitter users from neighbors. *ICWSM* 270:2012.

Nikolaos Aletras and Benjamin Paul Chamberlain. 2018. Predicting twitter user socioeconomic attributes with network and language information. *arXiv preprint arXiv:1804.04095* .

Aaron Antonovsky. 1967. Social class, life expectancy and overall mortality. *The Milbank Memorial Fund Quarterly* 45(2):31–73.

Pablo Barberá. 2016. Less is more? how demographic sample weights can improve public opinion estimates based on twitter data. Technical report, Working Paper.

Pablo Barberá and Gonzalo Rivero. 2015. Understanding the political representativeness of twitter users. *Social Science Computer Review* 33(6):712–729.

Eszter Bokányi, Zoltán Lábszki, and Gábor Vattay. 2017. Prediction of employment and unemployment rates from twitter daily rhythms in the us. *EPJ Data Science* 6(1):14.

Jazmin L Brown-Iannuzzi, Kristjen B Lundberg, and Stephanie McKee. 2017. The politics of socioeconomic status: how socioeconomic status may influence political attitudes and engagement. *Current opinion in psychology* 18:11–14.

Philipp Burckhardt, Raymond Duch, and Akitaka Matsuo. 2016. Tweet as a tool for election forecast: Uk 2015. general election as an example. *En: Third annual meefing of the Asian Polifical Methodology Society in Beijing* .

Andrew Chadwick. 2006. Internet politics: States, citizens, and new communication technologies. *New York, NY* .

J Crosby Chapman and Verner Martin Sims. 1925. The quantitative measurement of certain aspects of socio-economic status. *Journal of Educational Psychology* 16(6):380.

Ying Chen and Bei Pei. 2014. Weakly-supervised occupation detection for micro-blogging users. In *Natural Language Processing and Chinese Computing*, Springer, pages 299–310.

Charles D Cowan, Robert M Hauser, R Kominski, Henry M Levin, S Lucas, S Morgan, and C Chapman. 2012. Improving the measurement of socioeconomic status for the national assessment of educational progress: A theoretical foundation. *National Center for Education Statistics. Retrieved from http://files. eric. ed. gov/fulltext/ED542101. pdf* .

Aron Culotta, Nirmal Kumar Ravi, and Jennifer Cutler. 2016. Predicting twitter user demographics using distant supervision from website traffic data. *Journal of Artificial Intelligence Research* 55:389–408.

Lucie Flekova, Daniel Preoţiuc-Pietro, and Lyle Ungar. 2016. Exploring stylistic variation with age and income on twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 313–319.

Wendell L French. 1959. Can a man's occupation be predicted? *Journal of Counseling Psychology* 6(2):95.

Mohammed Hasanuzzaman, Sabyasachi Kamila, Mandeep Kaur, Sriparna Saha, and Asif Ekbal. 2017. Temporal orientation of tweets for predicting income of users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 659–665.

Erika Hoff, Brett Laursen, Twila Tardif, et al. 2002. Socioeconomic status and parenting. *Handbook of parenting Volume 2: Biology and ecology of parenting* 8(2):231–252.

Tianran Hu, Haoyuan Xiao, Jiebo Luo, and Thuyvy Thi Nguyen. 2016. What the language you tweet says about your occupation. In *Tenth International AAAI Conference on Web and Social Media*.

Kazushi Ikeda, Gen Hattori, Chihiro Ono, Hideki Asoh, and Teruo Higashino. 2013. Twitter user profiling based on text and community mining for market analysis. *Knowledge-Based Systems* 51:35–47.

Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2007. I know what you did last summer: query logs and user privacy. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, pages 909–914.

Bente Kalsnes, Anders Olof Larsson, and Gunn Sara Enli. 2017. The social media logic of political interaction: Exploring citizens and politicians relationship on facebook and twitter. *First Monday* 22(2).

Sunghwan Mac Kim, Stephen Wan, and Cécile Paris. 2016. Occupational representativeness in twitter. In *Proceedings of the 21st Australasian Document Computing Symposium*. ACM, pages 57–64.

Evelyn M Kitagawa and Philip M Hauser. 1973. Differential mortality in the united states: A study in socioeconomic epidemiology .

Nancy Krieger, David R Williams, and Nancy E Moss. 1997. Measuring social class in us public health research: concepts, methodologies, and guidelines. *Annual review of public health* 18(1):341–378.

Vasileios Lampos, Nikolaos Aletras, Jens K Geyti, Bin Zou, and Ingemar J Cox. 2016. Inferring the socioeconomic status of social media users based on behaviour and language. In *European Conference on Information Retrieval*. Springer, pages 689–695.

Kristina Lerman, Megha Arora, Luciano Gallegos, Ponnurangam Kumaraguru, and David Garcia. 2016. Emotions, demographics and sociability in twitter interactions. In *ICWSM*. pages 201–210.

Jacob Levy Abitbol, Eric Fleury, and Márton Karsai. 2019. Optimal proxy selection for socioeconomic status inference on twitter. *Complexity* 2019.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 165–174.

Yong Li, Mengjiong Qian, Depeng Jin, Pan Hui, and Athanasios V Vasilakos. 2015. Revealing the efficiency of information diffusion in online social networks of microblog. *Information Sciences* 293:383–389.

Sunghwan Mac Kim, Stephen Wan, Cécile Paris, Jin Brian, and Bella Robinson. 2016. The effects of data collection methods in twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*. pages 86–91.

Michael G Marmot, Manolis Kogevinas, and Maryann A Elston. 1987. Social/economic status and disease. *Annual review of public health* 8(1):111–135.

Fons Mentink. 2016. *Machine driven predictions of the socio-economic status of Twitter users*. Master's thesis, University of Twente.

Renato Miranda Filho, Guilherme R Borges, Jussara M Almeida, and Gisele L Pappa. 2014. Inferring user social class in online social networks. In *SNAKDD*. pages 10–1.

Paul L Morgan, George Farkas, Marianne M Hillemeier, and Steven Maczuga. 2009. Risk factors for learning-related behavior problems at 24 months of age: Population-based estimates. *Journal of abnormal child psychology* 37(3):401.

J Michael Oakes and Peter H Rossi. 2003. The measurement of ses in health research: current practice and steps toward a new approach. *Social science & medicine* 56(4):769–784.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 430–438.

Adam Poulston, Mark Stevenson, and Kalina Bontcheva. 2016. User profiling with geo-located posts and demographic data. In *Proceedings of the First Workshop on NLP and Computational Social Science*. pages 43–48.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1754–1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying user income through language, behaviour and affect in social media. *PloS one* 10(9):e0138717.

Daniele Quercia, Jonathan Ellis, Licia Capra, and Jon Crowcroft. 2012. Tracking gross community happiness from tweets. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, pages 965–968.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 1524–1534.

Peter Schmidt and Robert P Strauss. 1975. The prediction of occupation using multiple logit models. *International Economic Review* pages 471–486.

Elisafina Siswanto and Masayu Leylia Khodra. 2013. Predicting latent attributes of twitter user by employing lexical features. In *Information Technology and Electrical Engineering (ICITEE), 2013 International Conference on*. IEEE, pages 176–180.

Luke Sloan. 2017. Who tweets in the united kingdom? profiling the twitter population using the british social attitudes survey 2015. *Social Media+ Society* 3(1):2056305117698981.

Luke Sloan, Jeffrey Morgan, Pete Burnap, and Matthew Williams. 2015. Who tweets? deriving the demographic characteristics of age, occupation and social class from twitter user meta-data. *PloS one* 10(3):e0115545.

Aaron Smith and Joanna Brenner. 2012. Twitter use 2012. *Pew Internet & American Life Project* 4.

Brendan M Sullivan, Gopikrishna Karthikeyan, Zuli Liu, Wouter Lode Paul Massa, and Mahima Gupta. 2018. Socioeconomic group classification based on user features. US Patent App. 15/221,587.

AB Van Berkel-Van Schaik and B Tax. 1990. Towards a standard operationalisation of socioeconomic status for epidemiological and socio-medical research. *Rijswijk: ministerie van WVC* .

Reinder Gerard van Dalen, Léon Redmar Melein, and Barbara Plank. 2017. Profiling dutch authors on twitter: Discovering political preference and income level. *Computational Linguistics in the Netherlands Journal* 7:79–92.

Chris J Vargo and Toby Hopp. 2017. Socioeconomic status, social capital, and partisan polarity as predictors of political incivility on twitter: a congressional district-level analysis. *Social Science Computer Review* 35(1):10–32.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1567–1578.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media. In *AAAI*. pages 4296–4297.

Karl R White. 1982. The relation between socioeconomic status and academic achievement. *Psychological bulletin* 91(3):461.

Erik Olin Wright and G Ritzer. 2003. Encyclopedia of social theory.

Yuan Zhong, Nicholas Jing Yuan, Wen Zhong, Fuzheng Zhang, and Xing Xie. 2015. You are where you go: Inferring demographic attributes from location check-ins. In *Proceedings of the eighth ACM international conference on web search and data mining*. ACM, pages 295–304.

Divide and Extract –
Disentangling Clause Splitting and Proposition Extraction

Darina Gold **Torsten Zesch**
Language Technology Lab
University of Duisburg-Essen, Germany
{darina.gold,torsten.zesch}@uni-due.de

Abstract

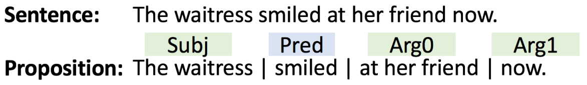
Proposition extraction from sentences is an important task for information extraction systems. Evaluation of such systems usually conflates two aspects: splitting complex sentences into clauses and the extraction of propositions. It is thus difficult to independently determine the quality of the proposition extraction step.

We create a manually annotated proposition dataset from sentences taken from restaurant reviews that distinguishes between clauses that need to be split and those that do not. The resulting proposition evaluation dataset allows us to independently compare the performance of proposition extraction systems on simple and complex clauses.

Although performance drastically drops on more complex sentences, we show that the same systems perform best on both simple and complex clauses. Furthermore, we show that specific kinds of subordinate clauses pose difficulties to most systems.

1 Introduction

Propositions are predicate-centered tuples consisting of the *verb*, the *subject*, and other *arguments* such as *objects* and *modifiers*. For example in Figure 1, "smiled" is the *predicate* and the other elements are *arguments*. The first argument is reserved for the role of the *subject*, in this case "The waitress", while "at her friend" and "now" are arguments, without further sub-specification. Propositions are used in language understanding tasks such as relation extraction (Riedel et al., 2013; Petroni et al., 2015), information retrieval (Löser et al., 2011; Giri et al., 2017), question answering (Khot et al., 2017), word analogy detection (Stanovsky et al., 2015), knowledge base construction (Dong et al., 2014; Stanovsky and Dagan, 2016), summarization (Melli et al., 2006), or other tasks that need comparative operations, such as equality, entailment, or contradiction, on phrases or sentences.


Sentence: The waitress smiled at her friend now.

Proposition: The waitress | smiled | at her friend | now.

Figure 1: Example Sentence and Extracted Proposition

The main goal of this paper is to empirically measure the influence of sentence complexity on the performance of proposition extraction systems. Complexity worsens the extraction of dependencies, on which propositions are built. Hence, proposition extraction performance should decrease with increasing sentence complexity.

The contribution of this work is threefold a) a gold standard corpus for propositions[1], b) an analysis of proposition extraction systems without the influence of complex sentences, and c) an analysis of proposition extraction systems with the influence of complex sentences.

The knowledge of how proposition extraction systems perform on complex sentences will 1) help to identify the system that deals with them best 2) by showing the difficulty with complexity, give a direction towards which proposition extraction systems can be improved.

If different systems perform well on simple or complex sentences, the complexity distinction could help to identify the complexity of a sentence. The complexity of a sentence would then give a direction towards which system would be better to use.

2 Related Work

Proposition are relational tupels extracted from sentences in the form of *predicate-argument struc-*

[1] https://github.com/MeDarina/review_propositions

Proceedings of Recent Advances in Natural Language Processing, pages 399–408,
Varna, Bulgaria, Sep 2–4 2019.

tures (Marcus et al., 1994). There are proposition models that further distinguish between the type of arguments. They do not only identify the subject, but more complex roles such as temporal and locational objects or causal clauses.

Besides the theory and formalization of proposition, proposition extraction systems have performance issues on real data.

2.1 Comparison of Proposition Systems

Although there have been comparative studies of proposition extraction systems, there has been no extensive study on the impact of sentence complexity on proposition extraction system performance.

Comparative Studies Niklaus et al. (2018) presented an overview of proposition extraction systems and classified them into the classic categories of learning-based, rule-based, and clause-based approaches, as well as approaches capturing inter-propositional relationships. They described the specific problems each system tackles as well as gaps on the overall evolution of proposition extraction systems.

Schneider et al. (2017) present a benchmark for analyzing errors in proposition extraction systems. Their classes are *wrong boundaries*, *redundant extraction*, *wrong extraction*, *uninformative extraction*, *missing extraction*, and *out of scope*. Their pre-defined classes do not map directly to sentence complexity, although *wrong boundaries* and *out of scope* would also be of some interest in an even more detailed error analysis.

Furthermore, according to Stanovsky and Dagan (2016) and Niklaus et al. (2018) there are no common guidelines and followingly no gold standard defining a valid extraction.

Systems Table 1 shows the outputs from different systems, our baselines, and our gold standard.

In their study, Gashteovski et al. (2017) aim at finding a system with minimal attributes, meaning that hedging[2] and attributes expressed e.g. through relative clauses or adjectives, can be optionally removed. Thus, they use recall and two kinds of precision in the evaluation in order to account for the feature of minimality. To explain this in more detail does not lie within the scope of this paper. Gashteovski et al. (2017) evaluates OLLIE (Mausam et al., 2012), ClausIE (Del Corro and Gemulla, 2013), and Stanford OIE (Angeli et al., 2015) against their own system.

Sentence	The waitress smiled at her friend now		
Systems	**Subject**	**Predicate**	**Other Elements**
Allen	The waitress	smiled	at her friend | now
ClausIE	The waitress	smiled	at her friend now
	The waitress	smiled	now
	her	has	friend
ReVerb	The waitress	now smiled at	her friend
Stanford	waitress	smiled at	her friend
	waitress	now smiled at	her friend
OLLIE	The waitress	now smiled at	her friend
OpenIE	The waitress	smiled	now | at her friend
BL1	The	waitress	smiled at her friend now
BL2	The waitress	smiled	at her friend now
Us	The waitress	smiled	at her friend | now

Table 1: Output of Proposition Extraction Systems and Our Two Baselines for the Sentence *The waitress smiled at her friend now*

Stanovsky et al. (2018) evaluates ClausIE, PropS (Stanovsky et al., 2016), and Open IE-4 against their new system, that we will call *Allen* (Stanovsky et al., 2018) herein, using precision-recall, area under the curve, and F1-score. They compare the individual proposition elements. For a proposition to be judged as correct, the predicate and the syntactic heads of the arguments need to be the same as the gold standard.

Saha et al. (2018) evaluate ClausIE, OpenIE-4, and CALMIE (a part of OpenIE) using precision. With the findings of this comparison, they introduce a new version of their system, OpenIE-5[3],

In all described comparisons, the system of the respective authors is the best, which makes sense as it addresses the issue shown by the authors.

2.2 Propositions from Simple Sentences

According to Saha et al. (2018) conjunctive sentences are one of the issues in proposition extraction, as conjunctions are a challenge to dependency parsers (Ficler and Goldberg, 2016) which proposition extraction systems are mostly built upon. Hence, Saha et al. (2018) built a system that automatically creates simple sentences from sentences with several conjunctions that are used for proposition extraction. For the proposition extraction of the simple sentences they used ClausIE and OpenIE. They evaluated their data using three different proposition datasets. The correctness of the extracted proposition from the original sentence were evaluated manually. In their study, simple sentences were sentences without conjunctions.

[2] In pragmatics, hedging is a textual construction that lessens the impact of an utterance. It is often expressed through modal verbs, adjectives, or adverbs.

[3] http://knowitall.github.io/openie/

Quirk (1985) defines a *simple sentence* as a sentence consisting of exactly one independent clause that does not contain any further clause as one of its elements. Hence, a *complex sentence* consists of more than one clause. This is also the definition that we use in our study.

2.3 Crowdsourcing Gold Standard Propositions

Recent work used crowdsourcing for creating and evaluating proposition extraction (Michael et al., 2018; FitzGerald et al., 2018) in the setting of question answering. In short, they asked their crowdworkers to produce questions and answers in a way that resulted in the extraction of their predicates and arguments, without directly asking for predicate-argument structures.

3 Corpus Creation

We create a corpus to evaluate the performance of proposition extraction systems entangled with and disentangled from the task of clause splitting.

Our source corpus is the portion of the Aspect Based Sentiment Analysis (ABSA) task (Pontiki et al., 2014) concerned with restaurant reviews within one aspect – *service*. We use all 423 sentences that were annotated with this aspect. In a preliminary step, we produce a corpus of *reduced* sentences. To examine the influence of sentence complexity, we classify the reduced sentences as either 1) *simple* sentences, meaning sentences with potentially just one proposition, and 2) *complex* sentences, meaning sentences with potentially multiple propositions. Then, we produce propositions from the reduced sentences using expert annotation and evaluate it by calculating the inter-annotator agreement.

Our corpus contains 2,181 sentences (class distribution in Table 2) and 2,526 propositions.

3.1 Preliminary Step: Creating Reduced Sentences

As a preliminary step, we created a gold corpus of reduced sentences formed from originally more complex sentences.

To do so, we use 423 sentences from review texts[4]. As these are quite difficult for producing propositions, even for humans, we included a preliminary step of creating *reduced sentences*. A *reduced sentence* is a sentence that contains only a portion of the original sentence, e.g. the original sentence “The server was cool and served food and drinks” could be reduced to “The server was cool” or “The server served food”. The intention behind this step was to create sentences with one proposition only. Hence, the guidelines contained rules such as decomposing conjunctive sentences or creating independent sentences from relative clauses.[5] We perform this preliminary step via crowdsourcing and evaluate it qualitatively.

Definition of Reduced Sentences We instructed our workers to produce reduced sentences from the original sentence. To prevent nested structures, a reduced sentence was not allowed to be split in further reduced sentences, at least within the output of one worker.[6] Ideally, the crowdworkers could have created sentences that contain exactly one proposition. However, this might even be a difficult task for experts, as there are non-trivial sentence constructions that would need long guidelines to create sentences with exactly one proposition. However, our guidelines insured that sentences were reduced in comparison to the original version, if possible. In this way, we are able to create a sufficiently big set of both simple and more complex sentences, as shown in Table 2.

Crowdsourcing We used Amazon Turk for crowdsourcing our data. Michael et al. (2018) crowdsourced gold data for evaluating propositions. The sentence reduction performed here and also in Saha et al. (2018) is very similar to syntactic sentence simplification as performed by Lee and Don (2017). We paid 0.04 $ per HIT and 0.01 $ for each further reduced sentence. Each sentence was reduced by 3 workers. In this process, 2181 unique reduced sentences, which are all used in the following corpus creation process, were created from 423 original sentences.

Evaluation of Reduced Sentences To measure the quality of the crowdsourced reduced sentences, we chose 100 random reduced sentences together with their original sentence and evalu-

[4] Online users’ restaurant reviews are a fruitful domain for proposition extraction, as propositions extracted from reviews would be useful for several user-centered tasks, as they would allow to display only information pieces of interest.

[5] However, this step turned out to be more difficult than expected, as some sentences contained several factors that could be reduced. However, this did not influence our goal of determining the influence of sentence complexity.

[6] The annotation instructions are also available on our Github page.

Complexity Class	# of Occurrences
No Verb	101
Simple	1,648
Complex	432
All	2,181

Table 2: Distribution of Sentence Complexity Classes in Our Reduced Sentence Set

ated their correctness using the following non-exclusive categories: ORIGINALSIMPLE, REDUCED, SIMPLE, GRAMMAR, and INFERENCE (see Table 3b).

In Table 3a, we provide an exemplary sentence for each category, except for ORIGINALSIMPLE, as it means that the original is already a simple sentence, containing only one proposition which cannot be further reduced. 20 sentences in the random sample were categorized as being ORIGINALSIMPLE. However, some workers still tried to reduce some of these sentences – 2 of them were grammatically incorrect (GRAMMAR) and 3 fell into the class INFERENCE. This means that their content was not explicitly mentioned in the original sentence, but was lexically inferred.

There were 66 REDUCED sentences, meaning that the sentences have been successfully reduced. 60 of the REDUCED resulted in SIMPLE sentences, which means that they contained only one proposition after the reduction, and 6 were simpler than the original sentence, but contained more than one proposition.

We believe that the results are usable as is, as the error rate is quite low – only 17 of the reduced sentences in the random sample were incorrect (GRAMMAR and INFERENCE), as many of the GRAMMAR errors stem from the original sentence. Furthermore, we show that our reduction step was necessary to produce enough simple sentences for our experiment, as 80% of the random sample were originally complex.

3.2 Creating Propositions from Simple Sentences

To evaluate the performance of proposition extraction systems, we created a gold standard corpus for propositions from the reduced sentences.

In this paper, we follow the most simple possible annotation, similar to Stanovsky et al. (2018).

We want to extract English propositions with one main verb and all arguments that are linked to it. In our notation, the first position of the proposition is the subject, the second is the predicate and the order of the other elements is irrelevant.[7]

The arguments may also contain further propositions, e.g. here, the sentence "I think their food is great" is split in two propositions – "I | think | their food is great" and "their food | is | great ". This definition is restrictive in that it asks for exactly two propositions in the given example. Additionally, it is not bound to a clearly defined theory (as there is no clearly defined theory on propositions). However, it is the representation that is needed to extract information from reviews, as it would help to reduce redundancies, e.g. by clustering sentences such as "Their food is great" and "I think their food is great". Furthermore, we are not interested in inferred information, e.g. "They | have | food" from the previously discussed sentence. This choice will also be reflected in the performance of systems that do not adhere to our understanding of propositions. However, this does not necessarily cloud the performance comparison of simple and complex sentences, as we will still measure the influence of sentence complexity. Each sentence is processed by two annotators and the disagreements are curated in a subsequent step.

Creation As the creation of propositions is not a trivial task, due to many different cases that need to be explained in the guidelines[8], this task should be performed by people who were trained longer than a crowdsourcing platform allows for. Thus, we produced proposition annotations in a double-annotation process by three graduate students[9]. The disagreements were curated by the first author of the paper. The result of the curation builds the gold standard. The gold standard, all annotations, and the guidelines are available.

3.3 Evaluation of Proposition Creation

To evaluate our dataset, we report inter-annotator agreement as well as agreement with the curator

[7] We are not interested in different types of objects and modifiers, similar to Stanford, OpenIE, and AllenNLP, and thus we do not discuss this information. For a better overview, we asked the annotators to present the other elements in their order of occurrence.

[8] The guidelines include explanations of what predicates, arguments, and nested propositions are. This in itself is not difficult. However, such instructions consume more time and need more training, as simple mistakes are made by untrained annotators. We saw this in a training set for this task, that is not included or discussed here due to space restrictions.

[9] The result is shown in Table 4. A1 annotated the whole set, while A2 and A3 annotated parts.

Original Sentence	The server was cool and served food and drinks.
REDUCED	The server was cool and served food.
SIMPLE	The server was cool.
GRAMMAR	The server was.
INFERENCE	The server is good.

(a) Classification Examples

Sentence Class	#
ORIGINALSIMPLE	20
REDUCED	66
SIMPLE	87
GRAMMAR	5
INFERENCE	12

(b) Distribution of Classification

Table 3: Classification of Reduced Sentences

on both the proposition (see Table 4a) and proposition element level (see Table 4b).

Evaluation Metric In order to see differences in the annotation, we performed inter-annotator agreement using %-agreement (accuracy). We use the same measure for system performance, which enables a direct comparison. Although we are aware that agreement is ignorant of chance agreement, we believe that it is the best measure for this problem, as chance agreement is quite low in the case of this complex annotation problem. Furthermore, it is difficult to interpret these results in comparison to other works. As previously described, there are no clear guidelines for propositions and also no manual gold datasets created explicitly for this purpose. We could compare the results of our inter-annotator agreement to similar tasks, where sentences are split into components, as e.g. answers prepared for question answering, paraphrase alignment, translation alignment etc. However, they also have different setups and evaluation metrics and it is out of the scope of this work to discuss these differences.

Levels of Evaluation We perform the evaluation on two levels - *proposition level* and *proposition element level*. On the proposition level, we calculate the agreement of whole propositions. On the proposition element level we calculate the agreement of individual elements of the propositions whilst taking their label (subject, predicate, or other element) into account.

Inter-annotator Agreement Table 4a shows that the inter-annotator agreement on the proposition level is .39 and .53 on complex sentences and .61 and .71 on simple sentences. These agreement differences show that clause splitting is also difficult for humans.

Agreement with Curator The agreement with the curator is .05 to .19 higher than the inter-annotator agreement. The agreement on the proposition element level is .67 and .7 on complex sentences and .83 and .85 for simple sentences - nearly double of the whole proposition agreement.

	Simple		**Complex**		**All**	
	A1	Gold	A1	Gold	A1	Gold
A1	-	.80	-	.66	-	.76
A2	.71	.79	.53	.63	.66	.74
A3	.61	.66	.39	.48	.57	.62

(a) Inter-Annotator Agreement on Propositions

	Simple		**Complex**		**All**	
	A1	Gold	A1	Gold	A1	Gold
A1	-	.90	-	.77	-	.86
A2	.85	.79	.70	.63	.81	.74
A3	.83	.83	.67	.70	.80	.80

(b) Inter-Annotator Agreement on Proposition Elements

Table 4: Inter-Annotator Agreement in Accuracy

4 Evaluation of Proposition Extraction Systems

Similar to Saha et al. (2018); Schneider et al. (2017) and Niklaus et al. (2018), we evaluate proposition system performance. They do not, however, regard the task of proposition extraction disentangled from the intrinsic subtask of clause splitting. By showing the performance of both simple and complex sentences, we are furthermore able to show the impact of clause splitting.

4.1 Setup

To identify the system that performs best when disentangled from the task of clause splitting, we use the herein produced corpus to analyze and evaluate the performance of various proposition extraction systems as used in evaluations by Stanovsky and Dagan (2016), Gashteovski et al. (2017), Saha et al. (2018), and Stanovsky et al.

(2018). Hence, we will analyze proposition extraction performance using AllenNLP, ClausIE, ReVerb, Stanford Open Information Extraction, OLLIE, and OpenIE-5.[10] Furthermore, we provide two baseline systems.

We use %-agreement to measure the performance of systems. We want full agreement, not just matching phrase heads, as performed by Stanovsky et al. (2018). Furthermore, we evaluate only agreement, as in our setup the argument or the predicate matching is what we are interested in, meaning we do not need precision and recall in our setting. In this way, our evaluation setup is similar to Saha et al. (2018), who also identified specific issues in proposition extraction systems.

As in inter-annotator agreement, we calculate agreement on two levels: proposition and proposition element level. The results of the performance comparison is shown in Table 5.

4.2 Baselines

We provide two baselines in order to better compare the systems. Both baselines create propositions with three elements at most: subject, predicate, and one other element. The first baseline (BL1) takes the first word as subject, the second word as predicate and the rest as one other element. The second baseline (BL2) is a little more engineered and uses POS-tags. It makes a proposition for each verb. All words before the verb are the subject and all words after the verb are one other element. Examples for the baselines are shown in the Table 1. The baselines are kept simple on purpose to show how simple algorithms can solve the given problem. A baseline that appears intuitive is using a dependency parser and filtering for the root and its dependants. However, deciding which parts are its dependents and especially the span of arguments is ambiguous. This would not be a baseline, it would be a rule-based system that is not out of the box. Hence, we decided not to do it.

4.3 System Performance

Table 5a shows that performance of proposition extraction on whole propositions is equally bad for both simple and complex sentences. Table 5b shows that performance on proposition elements is much better than on proposition level. Furthermore, the table shows that for all systems but ReVerb, the performance is much better on the simple sentences, which was expected.

It is also interesting that although the performance of both baselines on whole propositions is 0, the performance of the second baseline on proposition elements is competitive. This shows, that the task of proposition extraction can, to a big part, be solved by correct verb extraction. It outperforms ReVerb, Stanford, and on simple and complex sentences also OLLIE. The second baseline performs a little worse on all sentences, as these also include sentences without a verb and this baseline is verb-based. This shows that either the automatic systems have problems with the extraction of verbs or they have deeper issues, e.g. they do not extract from a lot of sentences, as is discussed in Section 4.4.1. The second baseline performs almost equally on both simple and complex sentences. This may show correct verb extraction alone solves only a particular portion of proposition extraction.

Other systems, especially the two best ones, perform about two times better on the simple sentences but then have a much bigger drop on the complex sentences. This may show that clause splitting has a bigger impact on better or probably more intelligent systems than on more simple systems.

On both levels, OpenIE is the best system, very closely followed by Allen, whereas the other systems are well-beaten.

4.4 Analysis of System Performance

Identifying further problems except clause splitting could improve current proposition extraction systems. On the one hand there are sub-issues in clause splitting. On the other hand, there are issues besides clause splitting.

In the case of ClausIE and ReVerb, many further clauses and also arguments are cut, as these consist of a maximum of three elements, which makes the comparison difficult.

4.4.1 General Issues

We first manually examined some potential issues in the proposition extraction from simple sentences. After the manual analysis of potential issues, we calculated the system performance if the issue would be eliminated. One big issue we found is **missing propositions**, meaning that systems do

[10] We will not use MinIE (Gashteovski et al., 2017), as it is an extension of ClausIE providing additional information such as modality and whether an argument is necessary or unnecessary, which is disregarded in this work.

Systems	Simple	Complex	All
Allen	.08	.09	.08
ClausIE	.06	.09	.07
ReVerb	.02	.02	.02
Stanford	.01	.01	.01
OLLIE	.03	.04	.03
OpenIE	**.09**	**.12**	**.09**
BL1	.00	.00	.00
BL2	.00	.00	.00

(a) System Performance on Propositions

Systems	Simple	Complex	All
Allen	.50	.40	.46
ClausIE	.37	.36	.36
ReVerb	.15	.14	.14
Stanford	.20	.09	.17
OLLIE	.24	.19	.22
OpenIE	**.51**	**.42**	**.47**
BL1	.05	.04	.05
BL2	.26	.24	.21

(b) System Performance on Proposition Elements

Table 5: System Performance Measured in Accuracy

Systems	Missing	Conditional	Temporal
Allen	.08	.13	.19
ClausIE	.06	.11	.13
ReVerb	.03	.00	.03
Stanford	.02	.00	.00
OLLIE	.04	.06	.02
OpenIE	.10	.19	.17

(a) System Performance on Propositions Excluding Specific Issues

Systems	Missing	Conditional	Temporal
Allen	.50	.57	.55
ClausIE	.38	.40	.38
Stanford	.26	.03	.14
ReVerb	.32	.00	.21
OLLIE	.31	.00	.20
OpenIE	.54	.53	.50

(b) System Performance on Proposition Elements Excluding Specific Issues

Table 6: System Performance Excluding Specific Issues

not always extract propositions. Except for the missing propositions, there was no big difference in the system performance with or without the issue. Also, some systems have different models of propositions, which may also affect their performance. On the one hand, there are issues with previous steps, e.g. **negations** or **quantifiers** are ignored. On the other hand, there are issues with formatting, e.g. a different treatment of **prepositions** or conditionals.

Missing Propositions One big issue is that proposition extraction systems often do not produce any extraction from a sentence. Unsurprisingly, this issue is bigger among the systems that do not perform well - namely ReVerb (58% of sentences do not have an extraction), Stanford (39%), and OLLIE (33%), whereas the better performing systems have much lower rates - Allen (3%), ClausIE (4%), and OpenIE (10%). In ReVerb, Stanford, and OLLIE we could not find a clear reason why there are no extractions. In the case of Allen, there are only no extractions from sentences without verbs.[11] ClausIE and OpenIE have no extractions from sentences that are missing a verb or a subject. Additionally, OpenIE has no extractions from existential clauses.

In Table 6a, where we show the performances of systems on full propositions without the discussed issues, it is shown that systems perform slightly better when eliminating missing propositions from simple sentences. However, the improvement is clearer in Table 6b on the element level. Especially for the systems that had more missing propositions, namely Stanford, ReVerb, and OLLIE, the change is between .06 - .17.

Conjunctions As already stated by Saha et al. (2018), conjunctive sentences pose an issue to proposition extraction systems. In our case, we wanted to separate all conjunctive sentences in individual propositions, e.g. the sentence “The waitress smiled at her friend and at me.” contains the propositions “The waitress | smiled | at her friend” and “The waitress | smiled | at me.”. OpenIE and Stanford have the same guidelines on conjunctions, whereas Allen, ClausIE, and ReVerb keep the conjuncted elements together – from the previous sentence they would create one proposition – “The waitress | smiled | at her friend and me.”.

Negations Stanford does not extract from negated

[11] These sentences are classified as neither simple nor complex, but are included in all.

sentences and Allen has problems with negated sentences missing a verb. The rest can deal with negations. These specific problems are difficult to show in numbers, as they are rare – only about 7% of the sentences contained negations.

Prepositions OLLIE, ReVerb, and Stanford place the prepositions with the predicate, whereas all other systems as well as our gold standard place it with the associated argument, as is shown in the example in Table 1. For these cases we would need adjusted evaluations that ignore this difference.

Quantifiers Stanford ignores "every" in propositions.

4.4.2 Issues with Complex Sentences

We looked at issues within complex clauses, namely conditional and temporal clauses.

Conditional Clauses In some cases, Allen, ClausIE, OLLIE, and OpenIE extract the if-clause for the argument, but delete the "if", which leads to disagreements on both full proposition and proposition element level. Comparing the performance on all complex clauses as shown in Table 5a to complex clauses without conditional clauses, as shown in Table 6a, all systems, except for ReVerb and Stanford, clearly perform better. Allen is better by .04 and OpenIE by .05, which shows that they have the biggest issues with conditional clauses. On proposition element level this becomes even clearer. Here, the three better systems, ClausIE, Allen, and OpenIE perform .04 - .17 better without conditional clauses.

Temporal Clauses Conceptually, Allen, OLLIE, and OpenIE extract temporal clauses correctly, but have some problems if the sentence is too long. Stanford cuts out the "when". For temporal clauses, the performance is similar to conditional clauses. The three better systems perform .06 -.11 better on full proposition level, and .02-.09 better on proposition element level. Stanford and OLLIE perform worse without the temporal clauses.

5 Summary

In this work, we described a method on how to create a dataset of reduced sentences from originally complex ones. We created an English dataset according to this method and further classified this dataset as simple and complex. It can be used for further evaluation of proposition extraction systems. The dataset enabled us to research the performance of proposition extraction detached from the task of clause splitting.

On the one hand, we showed that sentence complexity has a measurable impact on proposition extraction performance of both humans and machines. Hence, one step towards improving the performance of such systems, is the improvement of clause splitting. Furthermore, we believe that the performance of the original complex sentences, without the preliminary reduction step, would pose an even bigger problem to proposition systems, which implies that using these systems on real data could be problematic.

On the other hand, our study also showed that the ranking of systems is similar among simple and complex sentences. This means, that the best performing systems among simple sentences that are disentangled from the task of clause splitting, are also the best in complex sentences, where clause splitting also needs to be performed. This may mean that to find the overall best system, one does not need to classify between simple and complex sentences. However, it is necessary to find that sentence complexity is one problem of proposition extraction.

Also, our intelligent baseline system, that was able to extract verbs, outperformed three of the systems. However, the better systems did not only perform much better, but they were also more affected by sentence complexity.

Additionally, we looked into further problems of proposition extraction systems. The main issues in complex sentences that we could identify were conditional and temporal clauses.

6 Future Work

In future work, we plan to enlarge the corpus in order to use it for studies on user-specific recommendations. We plan to display proposition-like information to the user to provide more specific information than is given by a long sentence. This work may help in clause splitting, as we not only provide a gold standard for it, but also describe a method on how to create it. Furthermore, we plan to built a proposition extraction system based on the findings from this paper.

Acknowledgement

We would like to thank Venelin Kovatchev and Marie Bexte for their annotations. This work has been funded by Deutsche Forschungsgemeinschaft within the project ASSURE.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 344–354.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-Based Open Information Extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. ACM.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Jessica Ficler and Yoav Goldberg. 2016. A Neural Network for Coordination Boundary Prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-Scale QA-SRL Parsing. *arXiv preprint arXiv:1805.05377*.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. Minie: minimizing facts in open information extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640.

Rachayita Giri, Yosha Porwal, Vaibhavi Shukla, Palak Chadha, and Rishabh Kaushal. 2017. Approaches for information retrieval in legal documents. In *Contemporary Computing (IC3), 2017 Tenth International Conference on*, pages 1–6. IEEE.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering Complex Questions Using Open Information Extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 311–316.

John Lee and J Buddhika K Pathirage Don. 2017. Splitting Complex English Sentences. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 50–55.

Alexander Löser, Sebastian Arnold, and Tillmann Fiehn. 2011. The GoOlap Fact Retrieval Framework. In *European Business Intelligence Summer School*, pages 84–97. Springer.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Gabor Melli, Zhongmin Shi, Yang Wang, Yudong Liu, Anoop Sarkar, and Fred Popowich. 2006. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2006 Summarization Task. In *Proceedings of the 6th Document Understanding Conference (DUC 2006)*.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. 2018. Crowdsourcing Question-Answer Meaning Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2, pages 560–568.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3866–3878.

Fabio Petroni, Luciano Del Corro, and Rainer Gemulla. 2015. CORE: Context-Aware Open Relation Extraction with Factorization Machines. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1763–1773.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2014)*, pages 27–35.

Randolph Quirk. 1985. *A grammar of contemporary English*, 11. impression edition. Longman, London.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Swarnadeep Saha et al. 2018. Open Information Extraction from Conjunctive Sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299.

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 11–18.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a Large Benchmark for Ipen Information Extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305.

Gabriel Stanovsky, Ido Dagan, et al. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 303–308.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting More Out Of Syntax with PROPS. *arXiv preprint arXiv:1603.01648*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 885–895.

Sparse Coding in Authorship Attribution for Polish Tweets

Piotr Grzybowski
Wrocław University
of Science and Technology
Wrocław, Poland
p.grzybowski2@gmail.com

Ewa Juralewicz
Wrocław University
of Science and Technology
Wrocław, Poland
evajuralewicz@gmail.com

Maciej Piasecki
Wrocław University
of Science and Technology
Wrocław, Poland
maciej.piasecki@pwr.edu.pl

Abstract

The study explores application of a simple Convolutional Neural Network for the problem of authorship attribution of tweets written in Polish. In our solution we use two-step compression of tweets using Byte Pair Encoding algorithm and vectorisation as an input to the distributional model generated for the large corpus of Polish tweets by word2vec algorithm. Our method achieves results comparable to the state-of-the-art approaches for the similar task on English tweets and expresses a very good performance in the classification of Polish tweets. We tested the proposed method in relation to the number of authors and tweets per author. We also juxtaposed results for authors with different topic backgrounds against each other.

1 Introduction

The problem of authorship attribution is one of the major areas of text classification. However, the issue is usually undertaken in the context of longer texts, such as book fragments, journal articles or emails. In recent years, mass media channels started playing a huge role in social life and made it possible to participate in global discussions, especially through social media. Twitter is one of the most influential social media platforms where anyone can share their opinion in form of a short text. Along with the growing influence of such platforms and limited user verification possibilities, importance of verifying the authorship of tweets and other short texts published on social media platforms has grown considerably. Such need is also motivated by moral responsibility of providing reliable media channels and discriminating propaganda messages.

Inspired by the results obtained in (Shrestha et al., 2017) as well as the simplicity of their method, we decided to verify its abilities in terms of classifying tweets written by selected Polish influencers. We treat the problem as a multiclass classification of texts. Our method differs from the original approach in terms of a chosen method for text encoding and next the way of obtaining its distributional vector representation as well as the scope of usage of data prepared in the processing pipeline, i.e. we use different data, unrelated to that used for classification, to train the distributional model.

2 Related Works

A big part of research done in terms of authorship attribution is relevant for big chunks of texts, where the sample of author's writing is relatively big (Gollub et al., 2013), (Frantzeskou et al., 2007), (Koppel et al., 2011). Recently, a lot of work has been done regarding authorship attribution of short texts, especially tweets due to their accessibility. The problem is challenging in comparison to the classification of longer texts as it is harder to maintain the classification accuracy along with the input pruning (Koppel and Winter, 2014). Some methods are based on stylistic features (Macleod and Grant, 2011) or word and character n-grams (Schwartz et al., 2013), (Sapkota et al., 2015).

Considering the fact that tweets may be of highly varying character with usage of multiple special characters, notorious misspellings and mixes of languages, character n-grams seem to be intuitively best-fit for the problem. Moreover, such approach appears to hold both topic-specific and morphology-specific information on the text (Koppel et al., 2011)

Proceedings of Recent Advances in Natural Language Processing, pages 409–417,
Varna, Bulgaria, Sep 2–4 2019.

(Sapkota et al., 2015). There were a couple of attempts to classify tweets based on their character n-gram representations (Schwartz et al., 2013) (Sari et al., 2017) (Zhang et al., 2015) with different approaches to extracting a subset of meaningful characters (Plakias and Stamatatos, 2008) (Sapkota et al., 2015). In terms of a classifier used for such a task, CNNs have been recently widely explored and proved successful for text analysis (Sierra et al., 2017) (Ruder et al., 2016).

A combination of the two approaches appears in (Shrestha et al., 2017) where a text is classified based on a sequence of input characters. The method uses straightforward sequences of characters and their n-grams which are then embedded and processed in a CNN classifier. Another approach which is using a byte pair encoding algorithm for text encoding together with the application of the same CNN classifier, proves that the method is comparable to the state-of-the-art despite another level of text compression (Wang, 2018). The latter, however, uses a limited range of characters with at least punctuation and white space characters ignored. Additionally, in both works the embedding layer is trained simultaneously with the classifier, which makes it impossible to transfer the acquired knowledge to external data sets.

3 Contribution

In our work we extend the solution presented in (Shrestha et al., 2017) by using *SentencePiece* with *Byte-Pair Encoding* algorithms (Kudo and Richardson, 2018), without exclusion of any characters. Moreover, instead of using a data-specific text embedding trained simultaneously with the classifier, we teach a separate distributional language model on a corpus of Polish tweets. We also test robustness of our solution by alternating topics in the training and testing corpora, as well as using disjunctive time windows for texts in both corpora.

4 Data

For the needs of the development and testing of our solution we collected two separate data sets:

Tweet Corpus – 1 020 234 Polish tweets (including 95 435 564 characters in total) obtained through Twitter API[1] by applying Polish language filter (automatic Twitter filter that can produce some noise) and in addition tracking the top 100 words from the frequency list generated for the Polish language (Kazojć, 2009) in the search query,

Influencer Set – 138 486 tweets written by 28 Polish influencers who were chosen manually.

The *Tweet Corpus* consists mostly of tweets in Polish. We decided not to filter out tweets including fragments in foreign languages as it is a common practice for the Twitter users to write their content with injections of other languages. Data has been gathered using **twint** Python module (Zacharias and Poldi, 2018).

For the *Influencer Set* we selected the authors mainly due to their activity and the number of followers. In addition we pre-selected four categories of authors, namely: *journalists* (8 authors), *politicians* (6 authors), *publicists* (4 authors) and *computer gamers* (2 authors). The topic-related groups are extrated as one of the goals of the experiments was to verify whether the subject on which particular users tweet is a strong distinguishing factor which affects obtained results. As the activity of different authors is diversified, the *Influencer Set* is highly unbalanced with the number of tweets per author deviating from 314 to 18 204 tweets per account. This problem was mediated by subsampling during the experiments.

5 Text Representation

Tweets are very short texts and do not include many repeated occurrences of typical stylometry markers like functional words. We can observe prevalence of the information content over typical stylistic markers. Thus, we need to search for semantic elements reoccurring for a single author, as well as characteristic idiosyncrasies of his language, e.g. words or expressions.

Tweet mostly include many abbreviations, typos, or language errors that result in relatively high variability of the language and complexity of the statistical picture. Thus, we need

[1] https://twitter.com

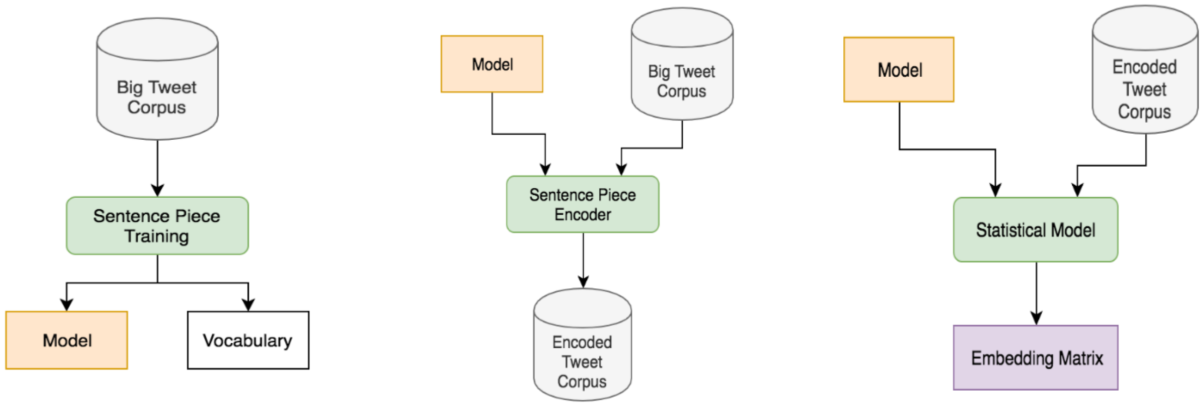


Figure 1: High-level structure of the processing pipeline used for obtaining text representation in our solution. From left to right: encoding model training, data encoding, embedding training.

to transform the original texts into a representation reducing this complexity. Very often, e.g. (Shrestha et al., 2017), tweets are represented by n-grams instead of words. However, the number of different n-grams is still very high. Instead of using n-grams, we processed the tweets with the help of the *Byte Pair Encoding* algorithm from *SentencePiece* library. *Byte Pair Encoding* (henceforth BPE) is a text compression method that constructs a tree-like structure of codes: recursively, two adjacent characters or symbols are encoded with a code represented by a single character that does not occur in the whole text. All punctuation and white characters are treated in the same way as other symbols. So there is no need for tokenisation. The algorithm starts from the most frequent pairs of characters (and next codes) and ends when all sequences are covered or it has reached the vocabulary size. Due to its recursive work the codes mostly represent not only bigrams but also higher-level n-grams, e.g. frequent words or even expressions. In all our experiments we used the BPE vocabulary size of: *4000 codes*. We tested vocabulary sizes from 1000 to 8000 codes and we did not notice improvement in performance for vocabulary size larger than 4000 codes.

BPE is often used for text compression, but our purpose was to make it a basis for a subword distributional semantics model. Texts encoded by BPE (i.e. tweets) were transformed into sequences of BPE codes separated by white spaces and delivered in such a form to the *Skip gram* algorithm from the *word2vec* library (Mikolov et al., 2013). As a result, every code receives its vector representation. The whole process is presented in Figure 1.

As codes represent different character sequences: from bi-grams, through sub-words up to even sub-expressions, we obtain a distributional semantics model which describes text units of varied granularity that reflect to some extent the statistical granularity of a corpus used for building the particular BPE model. A side effect is that vectors are also built for codes representing single letters that are rather meaningless, but this causes no harm to the overall properties of the model, as it will be visible.

The vector size in *Skip gram* was set to 300 elements. As the maximum length of a single tweet is 140 characters, so we made the representation of a single tweet to be a matrix of 140 code vectors[2]. In case BPE encoding for a tweet uses less than 140 codes (that often happens) the rest of the matrix is padded with a special null vector, i.e. not produced by *word2vec*.

In order to visualise the internal character of a BPE-encoded text, we present the histogram of initial tweet lengths in Figure 2, and contrast it with the histogram of the BPE-encoded tweet lengths in Figure 3. It can be noticed that the tweet lengths has become shorter, more evenly distributed with a slight dominance of the shorter representations. So,

[2] In the worst case of the weakest BPE mode codes correspond to single characters (symbols) in text, i.e. no more than 140 codes per a single tweet.

BPE-encoding results in a more packed text representation based sometimes on a few codes (representing sequences of letters).

Figure 4 presents the lengths of letter sequences represented by BPE codes obtained for the domain of politicians. The histograms are based on a subset of *Influencer Set*, containing tweets of 6 authors. When we compare it with the histogram of the whole domain in Figure 5 it can be observed that the code dictionary which is specific for the given domain contains slightly larger number of longer specialised codes. If we take a look into this domain specific dictionary we can find codes representing sometimes whole words that seem to be quite accidentally included into the dictionary. Thus, a dictionary built on the basis of a large corpus can be better suited to analysis of the authors' style that will be visible in Section 7 and especially in the results presented in Tables 4 and 3.

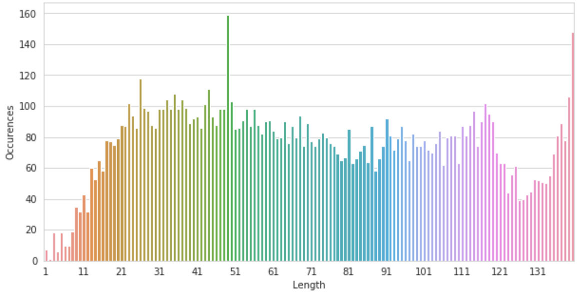


Figure 2: Histogram of tweet lengths for 28 635 tweets of politicians.

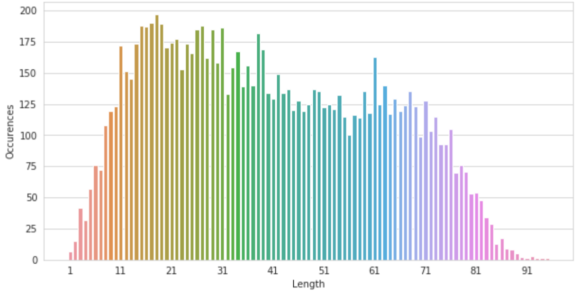


Figure 3: Histogram of BPE-encoded tweet lengths for 28 635 tweets of politicians.

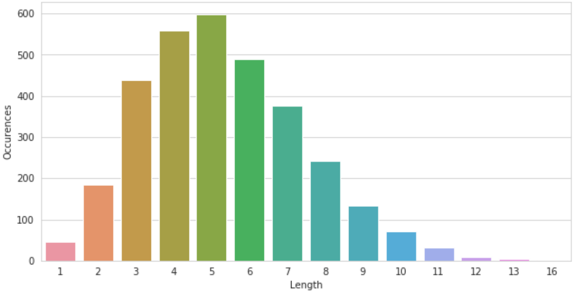


Figure 4: Histogram of BPE code lengths obtained by training the model on tweets of politicians (28 635 tweets).

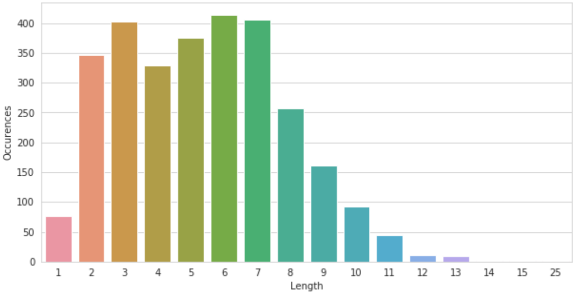


Figure 5: Histogram of BPE code lengths obtained by training the model on *Tweet Corpus* (1 020 234 tweets).

6 Classification Model

Our classification model follows the main lines *N-gram CNN* (Shrestha et al., 2017), but with BPE-based distributional representation in place of the original n-gram based one, i.e. instead of dividing the text into n-grams of characters directly from tweets, we split them into symbols obtained from the SentencePiece model. BPE-based model significantly limits the domain of possible symbols to the most common ones. It saves the vocabulary and the embedding matrix size. The neural network model architecture used in our research is visualised in Figure 6.

The architecture consists of an embedding layer and three parallel convolutional layers. Each convolutional layer contains 500 filters of sizes 3 × 3, 4 × 4 and 5 × 5 and is followed by max-pooling and dropout with 0.2 probability. The convolutional layers are merged and passed to a single dense layer with a 100 units and ReLU activation which is followed

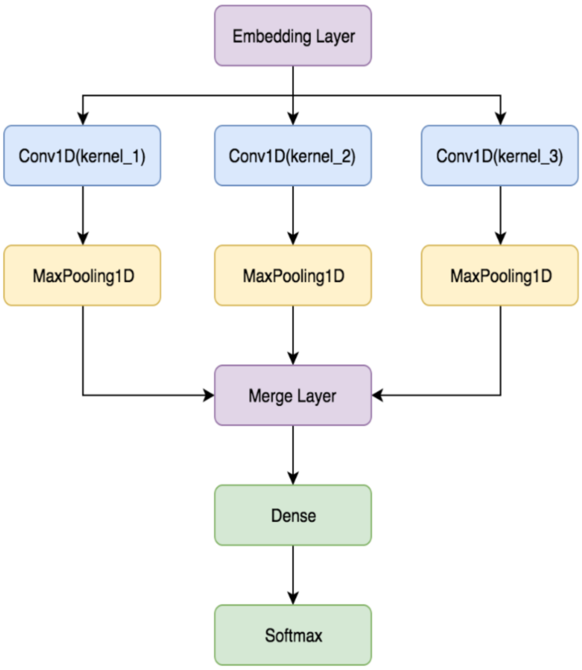


Figure 6: High-level structure of the N-gram CNN model used for author classification.

by dense layer with softmax activation.

Texts on the input to the embedding layer are represented by vectors pre-trained on the BPE encoded *Tweet Corpus*. During training the classifier, this layer was not completely frozen. Instead, it was adapted but with a significantly reduced learning rate as compared to the one used in the remaining layers. In this way, we manage to preserve high level generalisation obtained during training on the tweet corpus while slightly adapting to the training data domain.

The hyperparameters were slightly tuned from their initial state but their values are comparable to the ones used in the model of (Shrestha et al., 2017). We used Adam optimiser with 0.001 learning rate and categorical cross-entropy loss function. We trained the classifier with data batches of 4 samples in 5 epochs with L2 regularisation scaled by a 0.001 factor.

7 Experiments

7.1 Data Preprocessing

Before building the distributional semantics model and training the classifiers, we performed text preprocessing on tweets aimed at removing elements that seemed to be not relevant for the authors' styles and which could bias the classification process towards topic recognition. Thus, we removed all hashtags, mentions and URLs before constructing BPE encoding. The BPE models were generated from the preprocessed *Tweet Corpus* and selected subcorpora. Nonetheless, the very fact of using such extra-linguistic tweet elements, their placement and frequency might be important and relevant to authors' tweeting styles. Thus, we replaced all occurrences of extra-linguistic elements with symbols representing their types:

- *hashtags* were exchanged with '#' sign,
- *mentions* with the '@' sign,
- *URLs* with the 'ň' sign (not present in neither *Tweet Corpus* nor *Influencer Set*).

7.2 Multi-domain Authorship Attribution

In all experiments *Tweet Corpus* was used only to build a distributional semantics model and *Influencer Set* (not overlapping with the former) was used as training and testing data in a way following the k-fold cross validation scheme.

During the first experiment, we wanted to analyse the overall performance of the proposed approach on the whole *Influencer Set*. In order to balance the data set in relation to the number of tweets per author, we generated different subsets of the authors with random sub-sampling of tweets from more active authors. We wanted to investigate how many authors' styles our model can learn and how many tweets of each author are necessary to achieve satisfying results. The results are presented in Table 1.

7.3 Baseline Comparison

We compare our method against the baseline method of (Shrestha et al., 2017), using the same data set, i.e. the set presented in (Schwartz et al., 2013). This data set does not contain external data for language model training, so our approach of using a separate corpus for this task could not be applied. Thus in this experiment our solution differs only in the method for text encoding – the baseline

Tweets	Acc - 2	Acc - 4	Acc - 8	Acc - 12	Acc - 17	Acc - 23	Acc - 26	Acc - 28
16000	97.82 %	-	-	-	-	-	-	-
8000	97.41 %	90.89 %	-	-	-	-	-	-
4000	95.12 %	87.42 %	84.08 %	-	-	-	-	-
2000	93.25 %	84.11 %	81.19 %	72.44 %	71.34 %	-	-	-
1000	92.53 %	82.37 %	73.18 %	66.13 %	62.38 %	57.81 %	-	-
500	87.50 %	81.25 %	71.46 %	57.25 %	49.29 %	49.61 %	51.77 %	-
250	88.03 %	69.52 %	63.00 %	44.67 %	48.47 %	38.43 %	40.23 %	42.14 %
100	82.51 %	63.25 %	58.62 %	42.08 %	39.41 %	31.11 %	32.48 %	30.53 %

Table 1: Classification results for a varying number of authors and number of tweets. First column indicates the number of tweets per user used in experiment and the first row presents how many authors were classified in it. For instance, Acc-8, means the column under this cell presents accuracy results for 8-class classification (8 authors).

No. of tweets	**N-gram CNN**	**Ours**
50	**0.562**	0.528
100	**0.617**	0.609
200	**0.665**	0.657
500	0.724	**0.742**
1000	**0.761**	0.758

Table 2: Accuracy results for (Schwartz et al., 2013) data set, compared with results obtained in (Shrestha et al., 2017) for 50 authors.

method tokenises text into n-grams, while ours is using the BPE algorithm to encode tweets. For this data set we used the vocabulary size of 4000, with embedding length of 300, 128 batch sizes and 1.0 for the learning rate for Stochastic Gradient Descent in embedding training. For the classifier, we used Adam optimizer with 0.0001 learning rate, trained for 50 epochs with batches of 8 samples. Our results are averaged over 10-fold cross-validation. The baseline results are taken from (Shrestha et al., 2017), where we took the best achieved scores out of two studied architectures.

The comparison of classification accuracy for 50 authors and varying number of tweets in presented in Table 2.

As it is visible in Table 2, the baseline method is slightly better in majority of the experiment setups. However, the data set provided by (Schwartz et al., 2013) does not have an explicit train/test split or even a specified pool of 50 authors to perform classification on. Each fold has been randomly subsampled from a large pool of 7 026 authors. Moreover, for each author a specified number of tweets has been also randomly subsampled from a pool of 1 000 tweets.

The results presented in (Shrestha et al., 2017) are also not explicitly said to be averaged over folds. Considering the above mentioned preconditions, the results obtained cannot be used to unambiguously determine a superior method out of the two in a straightforward manner.

7.4 Cross-domain Encoding and Embedding

In the next group of experiments we investigated to what extent the model reflects particular domains of the authors (i.e. reacting first to the domain signal), and to what extent it represents the authors' style by themselves. Thus, we tried to classify authors from a specific domain by using the encoding-embedding model built on some other domain. Three configurations of both data sets were analysed:

1. the same data for building encoding model and tests,
2. data from another domain is used to construct encoding model,
3. the Polish *Tweet Corpus* is used for build-

Table 3: Accuracy score for authors of different domains (rows) with usage of encoding and embedding trained on different data (columns) with embedding layer active during training.

		Encoding and embedding training data					
	Author Domain	Sport	Social	Politics	Publicists	Gamers	Corpus
Classified	Sport	0.89	0.85	0.79	-	-	0.94
	Social	0.94	0.94	-	-	0.98	0.95
	Politics	0.86	-	0.89	0.79	-	0.90
	Publicists	0.87	-	0.86	0.90	-	0.88
	Gamers	-	0.96	-	-	0.97	0.99

ing the encoding mode (i.e. this is the baseline case).

For domains in which the number of authors is not the same, we subsample from the larger pool and average the results over possible combinations (e.g. Sport: 3 authors vs Gamers: 2 authors – we subsampled 3 times).

In addition, we run the experiments in two setups:

- with the embedding layer active during training a classifier
- and with embedding layer remaining frozen.

The results from the first scenario are shown in Table 3, while the effects of the second setting are presented in Table 4. The label "Corpus" means that the encoding model and the embeddings were constructed on the large *Tweet Corpus*, while the classifiers were trained on the given domain.

The lack of a value means that for the given pair we did not have enough data to build a balanced training-testing subset by subsampling to the size of the smaller domain.

8 Results

During the experiments on the whole set of authors, as expected, the results significantly decrease with the increasing number of authors taken into account and consequently the decreasing number of tweets per an author (due to the subsampling). While the results fall behind those presented in (Shrestha et al., 2017), they still surpass initial expectations. It is worth to take into consideration that the Polish language is more complex than English from the statistical point of view due to the rich morphology and a weekly constrained word order. However the latter was of minor importance as the model works mostly on the subword level and the distributional semantics model that does not depend on the word order (i.e. it is not sequential). The subword embeddings seem to be useful in decomposing Polish morphological forms into their natural components.

In addition, Polish tweets often include fragments written in English (of varied correctness) that also adds to the complexity of the problem and models. It is worth mentioning that in our experiments we used a significantly smaller amount of data than it was done in (Shrestha et al., 2017).

We initially suspected that the model would be biased by the authors' domains bias (i.e. topics of tweets or characteristic elements of the domain jargon). However, the result in the cross-domain model scenario, see Table 3 and Table 4, show a different picture. Firstly, in all cases the BPE-based embedding model built on the large corpus appeared to be superior to the domain-based model. Definitely, the difference in size of the corpora mattered in all cases in favour of *Tweet Corpus*. However, the size of the code dictionary was constant and equal to 4 000, i.e. it was quite small, and we can expect that the generalisation in the case of the big corpus was substantial. When the embedding layer got frozen in the second cross-domain experiment, all the results decreased, but only slightly. So, in the case of the limited code dictionary – providing a sparse and generalised picture of texts – the domain-focused tuning of the embedding layer appeared to not be very important.

The comparison with the method of

Table 4: Accuracy score for authors of different domains (rows) with usage of encoding and embedding trained on different data (columns) with embedding layer frozen during training.

		Encoding and embedding training data					
	Author Domain	Sport	Social	Politics	Publicists	Gamers	Corpus
Classified	Sport	0.90	0.84	0.82	-	-	0.95
	Social	0.86	0.94	-	-	0.97	0.96
	Politics	0.85	-	0.86	0.82	-	0.91
	Publicists	-	-	0.85	0.86	-	0.90
	Gamers	-	0.98	-	-	0.99	0.99

(Shrestha et al., 2017) in Table 2 showed that this reference method performs slightly better, however, the top results come from an ensemble of the two different models while our results are achieved by the single method. Moreover, our model is targeted on languages with rich inflection.

9 Conclusions and Further Research

The proposed method for the authorship attribution for Polish tweets expressed good performance for a large group of authors and large data set. It is based on the idea of an application of the Convolutional Neural Network proposed by (Shrestha et al., 2017), but it expands this approach with a distributional semantics model based on the prior application of BPE text encoding (i.e. embedding vectors are built for BPE codes, not words). As a result the proposed methods seems to be better suited for processing short texts in a highly inflectional language. It is worth to emphasise that the proposed approach is language independent, as the BPE model is driven by the statistical patterns in the training corpus.

The idea of an adaptive, coarse-grained distributional text representation for the needs of non-semantic classification seems to be attractive and opens several questions. There are various points in which the whole process could be improved, including but not being limited to:

- collecting a larger tweet corpus,
- improving language-based filtering of obtained tweets,
- investigating more closely the influence of the code dictionary size,
- comparing the proposed approach with other embedding learning methods,
- and optimise the classifier architecture.

While language filtering for using tweets in the Polish language exclusively seems to be a reasonable thing to do, it might not be perfectly adequate for the content such as tweets. We can observe a growing trend for users to post content with both languages present and the usage of such a practice may also be considered as a characteristic feature of their writing style. Language switch recognition in such a short text like tweets might be erroneous, but its tracking can improve the representation.

The question to what extent the method recognises an author's style, and to what extent this is only due to the correlation with some topics can be answered by the analysis of the confusion matrices between authors, distinctive features and stability of the recognition in time. Such an analysis could be combined with a strict filtering of tweets concerning the same events or topics and then verifying the classification performance. This is especially important due to the fact that Twitter data gets quickly irrelevant as topics change very fast.

Acknowledgments

This work is partially financed as part of the investment in the CLARIN-PL research infrastructure funded by the Polish Ministry of Science and Higher Education.

References

Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E. Chaski, and

Blake Stephen Howald. 2007. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *IJDE* 6(1).

Tim Gollub, Martin Potthast, Anna Beyer, Matthias Busse, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2013. Recent trends in digital text forensics and its evaluation. In Pamela Forner, Henning Müller, Roberto Paredes, Paolo Rosso, and Benno Stein, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 282–302.

Jerzy Kazojć. 2009. Otwarty słownik frekwencyjny leksemów v.06.2009. https://web.archive.org/web/20091116122442/http://www.open-dictionaries.com/slownikfrleks.pdf.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Lang. Resour. Eval.* 45(1):83–94. https://doi.org/10.1007/s10579-009-9111-2.

Moshe Koppel and Yaron Winter. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology* 65. https://doi.org/10.1002/asi.22954.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR* abs/1808.06226. http://arxiv.org/abs/1808.06226.

Nicci Macleod and Tim Grant. 2011. Whose tweet?: authorship analysis of micro-blogs and other short form messages.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR* abs/1310.4546. http://arxiv.org/abs/1310.4546.

Spyridon Plakias and Efstathios Stamatatos. 2008. Tensor space models for authorship identification. In John Darzentas, George A. Vouros, Spyros Vosinakis, and Argyris Arnellos, editors, *Artificial Intelligence: Theories, Models and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 239–249.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *CoRR* abs/1609.06686. http://arxiv.org/abs/1609.06686.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 93–102. https://doi.org/10.3115/v1/N15-1010.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous n-gram representations for authorship attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 267–273. https://www.aclweb.org/anthology/E17-2043.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *EMNLP*. ACL, pages 1880–1891. http://aclweb.org/anthology//D/D13/D13-1193.pdf.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 669–674. https://www.aclweb.org/anthology/E17-2106.

Sebastián Sierra, Manuel Montes y Gómez, Thamar Solorio, and Fabio A. González. 2017. Convolutional neural networks for author profiling in pan 2017. In *CLEF*.

Zhenduo Wang. 2018. Text embedding methods on different levels fortweets authorship attribution. http://www.d.umn.edu/ tpederse/Pubs/zhenduo-report.pdf.

Cody Zacharias and Francesco Poldi. 2018. Twint project. `https://github.com/twintproject/twint`.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR* abs/1509.01626. http://arxiv.org/abs/1509.01626.

Automatic Question Answering for Medical MCQs: Can It Go Further than Information Retrieval?

Le An Ha and Victoria Yaneva
Research Institute in Information and Language Processing,
University of Wolverhampton, UK
{ha.l.a, v.yaneva}@wlv.ac.uk

Abstract

We present a novel approach to automatic question answering that does not depend on the performance of an information retrieval (IR) system and does not require training data. We evaluate the system performance on a challenging set of university-level medical science multiple-choice questions. Best performance is achieved when combining a neural approach with an IR approach, both of which work independently. Unlike previous approaches, the system achieves statistically significant improvement over the random guess baseline even for questions that are labeled as challenging based on the performance of baseline solvers.

1 Introduction

Automatic question answering has seen a renewed interest in recent years as a challenge problem for evaluating machine intelligence. This has driven the development of large-scale question-answering data sets such as SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), WikiMovies benchmark (Chen et al., 2017), TriviaQA (Joshi et al., 2017) (to name a few), as well as the organisation of workshops such as the Machine Reading for Question Answering 2018 workshop[1]. In spite of the optimistic advances over crowd-sourced questions and online queries, automatic question answering for real exam questions is still a very challenging and under-explored area. For example, the Allen AI Science Challenge[2] invited researchers worldwide to develop systems that could solve standardized eight-grade science questions. The best system out of all 780 participating teams achieved a score of 59.31% correct answers using a combination of 15 gradient-boosting models (random baseline of 25%), while the authors report that using Information Retrieval (IR) alone results in a score of 55%.

The difficulties related to answering exam questions are partly due to the complexity of the reasoning involved and partly to the lack of large training data. Another significant reason is the fact that the existing approaches to question answering are dependent on the performance of IR systems and can rarely go far beyond the performance of such systems. While IR is a powerful method when answering questions where the correct answer is a string contained within a document, the systems fail when the sentences within the question do not individually hold a clue to what the correct answer might be (Clark et al., 2018). This is one of the characteristics of Multiple Choice Questions (MCQs) from the science domain that makes them so challenging for both machines and for humans.

In this paper we aim to address these shortcomings by developing an approach that: i) does not require that the training data (often unavailable) be in the form of multiple-choice questions and ii) does not depend on matching strings of text with one another. We use a challenging set of medical exam questions developed for the United States Medical Licensing Examination (USMLE®), a standardized medical exam that university students need to pass in order to obtain the right to practice medicine in the US. As such, the USMLE represents a very difficult set, requiring a high level of specialized professional knowledge and reasoning over facts. Furthermore, the USMLE contains a wide variety of question types such as selecting the most appropriate diagnosis, treatment, specific further examination needed, etc., all of which require application of clinical knowledge over facts.

[1] https://mrqa2018.github.io/

[2] https://www.kaggle.com/c/the-allen-ai-science-challenge

Proceedings of Recent Advances in Natural Language Processing, pages 418–422,
Varna, Bulgaria, Sep 2–4 2019.

Contributions We introduce and compare two approaches for automatic question answering that do not require training data in the form of MCQs, using Information Retrieval (IR) techniques and standard neural network models. Unlike previous work, our neural approach is independent of the performance of the IR system, as it does not build upon it. Thus, it is possible to achieve improvements over both systems by combining them, as each system has an individual contributions towards solving the problem. The best combination results in 18% improvement over a random guess baseline. The neural models achieve a statistically significicant improvement over the random baseline on the challenging sets. The code used in this study, as well as the public data[3] are made available at: https://bit.ly/2jNW2ym.

2 Related Work

Most of the recent work in the field focuses on answering reading comprehension questions from benchmark datasets such as SQuAD (Rajpurkar et al., 2016), the release of which ignited a rapid progress in the field. For example, Wang et al. (2017) use gated self-matching networks and report accuracy as high as 75.9% over a random guess baseline of around 4% and a logistic regression baseline of around 51%. Among the most successful approaches in other studies are ones that use neural models such as match-LSTM to build question-aware passage representation (Wang and Jiang, 2015), bi-directional attention flow networks to model question-passage pairs (Seo et al., 2016), or dynamic co-attention networks (Xiong et al., 2016).

As mentioned in the previous section, automatic question answering for science exams is a lot more challenging than for crowd-sourced reading comprehension questions. When applied to science questions, IR techniques: i) still perform somewhat close to the state-of-the-art and ii) fail on tasks where the correct answer is not specifically contained in relevant sentences. Clark et al. (2018) implement five of the best models from the studies on the reading comprehension data sets (TableILP (Khashabi et al., 2016), TupleInference (Khot et al., 2017), Neural entailment models (DecompAttn, DGEM, and DGEM -OpenIE) (Parikh et al., 2016), and BiDAF (Seo et al., 2016)), as well as IR models and test them on a total of 7787 science questions. The questions are divided into two sets, challenging and easy, and are targeted at students between the ages of 8 and 13. It is important to note that the authors define a question as being challenging or easy not on the basis of human performance or the age of the students it is targeted at, but based on whether it has been answered incorrectly by at least two of the baseline solvers. The results indicated that none of the algorithms performed significantly higher than the random guess baseline of 25% on the challenging set, while the performance on the easy set was within the range of 36% and 62%. According to the authors, a possible explanation for the low accuracy is that nearly all models use some form of information retrieval to obtain relevant sentences, and the retrieval bias in these systems is towards sentences that are very similar to the question, as opposed to sentences that individually differ but together explain the correct answer (Clark et al., 2018). Notably, the neural solvers performed poorly on the easy set, while the best result was achieved by an IR-only system.

3 Data

In the USMLE data each test item is a single-best-answer MCQ consisting of a stem (question) followed by several response options (distractors), one of which is the correct answer (key). An example of such an item is provided in Table 1. We divide our data into two sets: private and public (Table 2). The private data set consists of a total of 2,720 MCQs and they are not available to the public due to test security reasons. The public data set consists of 454 items from USMLE 2015 Step 1, USMLE 2016 Step 1, USMLE 2014 Step 2, and USMLE 2017 Step 2 sample leaflets. These are available at the USMLE website[4] and in our repository. For the purpose of this study, we have selected only those items that fulfill the following criteria: i) whose correct answer contains at least one heading from the Medical Subject Headings (MeSH[5]) database that is at most three words, and ii) have exactly 5 options that have at least one MeSH heading that is at most three words. The

[3] See Section 3. The Public data set used in this study consists of questions released as training materials by the USMLE.

[4] The items can be accessed at the USMLE web site at http://www.usmle.org/, for example: http://www.usmle.org/pdfs/step-1/samples_step1.pdf

[5] https://www.nlm.nih.gov/mesh/

A 56-year-old man comes to the emergency department because of a 4-day history of colicky right flank pain that radiates to the groin and hematuria. Ultrasound examination of the kidneys shows right-sided hydronephrosis and a dilated ureter. Which of the following is most likely to be found on urinalysis? (A) Erythrocyte casts (B) Glucose (C) Leukocyte casts (D) Oval fat bodies (E)* Uric acid crystals

Table 1: An example of an item from the USMLE exam (question 128, USMLE 2015 step 1 sample test questions)

	Public	Private
Number of Items	164	921
Average words per item	116	87

Table 2: Characteristics of the two sets

latter is in order to keep the random guess baseline at a constant for all items (20%). As a result, the final data that we have is 164 items for the public set and 922 for the private one.

4 Method

We develop and compare two methods for answering the USMLE questions, both of which do not require training data in the form of MCQs. The details of each method are described below.

4.1 IR-Based Method

We use a standard IR approach. First, we index 2012 MEDLINE abstracts using Lucene[6] with its default options. Then, for each item we build the five queries, where each query contains the stem and an option. We use three settings for the queries:

- All words (IR-All) (baseline)
- Nouns only (IR-Nouns)
- Nouns, Verbs, or Adjectives only (IR-NVA),

We then get the top 5 documents returned by Lucene and calculate the sum of the retrieval scores. The picked answer is the one that has the highest score when combined with the stem to form the query. This method is similar to the IR baseline described in Clark et al. (2018) and variations of it have been previously applied to medical MCQs for the purposes of distractor generation (Ha and Yaneva, 2018) and predicting item difficulty (Ha et al., 2019).

4.2 Neural Network Method

For this approach we train neural networks to predict the MeSH headings for each abstract. The premise of this approach is that we hypothesise that the task of answering an USMLE item could be considered to be similar to the task of identifying the topics of a snippet of text: in the case of MEDLINE indexing, indexers read the abstract, and then choose the topics that are most relevant to the abstract; whereas in the case of taking USMLE exam, test takers read the stem, and then choose the option that is most relevant to the stem. Approaching the problem this way, we can benefit from the availability of the MEDLINE data, in which each abstract has been manually (or semi-manually) assigned most relevant subject headings. We focus only on headings that appear in the options of the set of items (see above). For our set, there are around 1000 headings. Our neural networks[7] were trained using Keras[8]. We use two main structures:

- Bidirectional LSTM (LSTM). Specifications: an input layer, followed by an embedding layer and a bidirectional layer, each of size 250. The final two layers are a flattening layer and a dense layer. The classes are weighted inversely to their frequency.
- Convoluted 1d with attention (Conv1d). Specifications: an input layer, followed by an embedding layer, three convolutional layers, and a concatenating layer, each of size 250.

[6] https://lucene.apache.org/

[7] Preprocessing includes tokenization (using keras.preprocessing.text package in python), no lower case normalization, no number normalization, recording words with a min frequency of 5. The neural network models then further restrict the vocabulary to the first 200000 most frequent words. Out of vocabulary rate was 1%. Nadam optimizer was used with its default options (learning rate = 0.002, beta_1 = 0.9, beta_2 = 0.999, epsilon = None, schedule_decay = 0.004). Batch size = 128, activation function used in the last layer was Softmax.

[8] https://keras.io/

Accuracy	Public set	Private set
Baselines		
Random guess baseline	0.2 (.16-.26)	0.2 (.175-0.225)
IR-All baseline	0.25 (.18-.32)	0.332 (.302-.364)
IR		
IR-NVA	0.32* (.24-.39)	0.362* (.332-.395)
IR-Nouns	0.33* (.26-.41)	0.311* (.282-.342)
Neural		
LSTM	0.29 (.22-.37)	0.29* (.26-.32)
Conv1d attention	0.31* (.23-.37)	0.32* (.292-.353)
Ensemble(Conv1d+LSTM)	0.30* (.24-.39)	0.311* (.282-.342)
Neural +IR		
log(IR_NVA)+log(conv1d)	0.32* (.25-.40)	0.340* (.310-.373)
Neural as tie breaker	**0.37**** (.3-.45)	**0.396**** (.365-.429)
Neural correct when IR incorrect	0.29* (.21-.38)	0.276* (.24-.31)
Neural correct when IR tie	9 out of 15	26 out of 89

Table 3: Accuracy of the different systems. The values marked with * signify statistically significant difference over the random guess baseline and ** signifies statistically significant improvement over both baselines.

These are followed by an attention layer and a densely connected layer.

We train the models on 10,000,000 MEDLINE abstracts (the same set used in the IR approach), going through them twice. We experiment with pre-trained GloVe840b (Pennington et al., 2014) and word2vec[9], but the results are inferior to training the embedding layers from scratch. We then use the trained models to predict the probability of a MeSH heading in an option given the stem. We then average the probabilities if the option contains more than 1 heading.

4.3 Combined Method

We use two methods to combine the IR and neural model scores. The first method just adds the log value of the two scores together (log(IR_Noun)+log(Conv1d)). The second method uses the neural model scores as a tie breaker ('Neural as tie breaker'): if the IR method returns a single option, we take the result from the IR. If the IR method returns more than one options, we take the results from the neural model instead.

4.4 Baselines

We compare our results to two baselines: the probability of a random guess to pick the correct answer and the IR-All model described above.

5 Results and Discussion

The results from our study are presented in Table 3. Best performance is achieved by using neural model scores as a tie breaker. This result significantly outperforms both the random guess baseline and the IR-All approach.

It is interesting to note that while neural approaches alone present a significant improvement only over the random guess baseline, using neural approaches to solve ties leads to an overall increase in performance for the best combined models. The independent nature of the neural approach is best illustrated when testing its performance over items that were incorrectly solved by the best IR approaches. This is the case for 110 items from the public data set, and 587 items from the private data set, which, if we follow the definition of Clark et al. (2018), can be regarded as "challenging" since the best IR solver could not answer them correctly. In the case of Clark et al. (2018) none of the tested solvers achieved significant improvement over the random guess baseline when evaluated on the challenging questions. In our case, the neural approaches achieve 29% accuracy (32 items) for the public data set and 27.6% accuracy (162 items) for the private one, which are both statistically significant when comparing to random guess. This independence, resulting from the use of humanly produced subject headings, indicate that these headings do provide additional information with regards to the task.

[9] https://code.google.com/archive/p/word2vec/

A drawback of the neural approach proposed in this paper is that it relies on the availability of a manually indexed database such as MEDLINE. This limits the applicability of the approach to other domains, however, this may change when more resources become available in the future. It is important to note that in this restricted setting the method solves a very difficult problem better than any other approach so far. In the future, instead of using the adhoc neural network architectures presented in this paper, we plan to utilise state-of-the-art architectures such as Elmo (Peters et al., 2018) or BERT (Devlin et al., 2018), while using the prediction of MESH headings as an additional learning objective.

6 Conclusion

We presented an approach to automatic question answering that does not rely on training data in the form of MCQs and can perform independently from IR. We first train neural networks to predict the MeSH headings for a set of MEDLINE abstracts and then use the trained network to predict the correct answers of medical MCQs. Best performance was achieved when combining this approach with an information retrieval approach and the model significantly outperformed both a random guess baseline and one based on a common IR approach.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Le An Ha and Victoria Yaneva. 2018. Automatic distractor suggestion for multiple-choice tests using concept embeddings and information retrieval. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 389–398.

Le An Ha, Victoria Yaneva, Peter Baldwin, and Janet Mee. 2019. Predicting the difficulty of multiple choice questions in a high-stakesmedical exam. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. *arXiv preprint arXiv:1604.06076*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. *arXiv preprint arXiv:1704.05572*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Self-Knowledge Distillation in Natural Language Processing

Sangchul Hahn
Handong Global University
Pohang, South Korea
schahn21@gmail.com

Heeyoul Choi
Handong Global University
Pohang, South Korea
heeyoul@gmail.com

Abstract

Since deep learning became a key player in natural language processing (NLP), many deep learning models have been showing remarkable performances in a variety of NLP tasks, and in some cases, they are even outperforming humans. Such high performance can be explained by efficient knowledge representation of deep learning models. While many methods have been proposed to learn more efficient representation, knowledge distillation from pretrained deep networks suggest that we can use more information from the soft target probability to train other neural networks. In this paper, we propose a new knowledge distillation method *self-knowledge distillation*, based on the soft target probabilities of the training model itself, where multimode information is distilled from the word embedding space right below the softmax layer. Due to the time complexity, our method approximates the soft target probabilities. In experiments, we applied the proposed method to two different and fundamental NLP tasks: language model and neural machine translation. The experiment results show that our proposed method improves performance on the tasks.

1 Introduction

Deep learning has achieved the state-of-the-art performance on many machine learning tasks, such as image classification, object recognition, and neural machine translation (He et al., 2016; Redmon and Farhadi, 2017; Vaswani et al., 2017) and outperformed humans on some tasks. In deep learning, one of the critical points for success is to learn better representation of data with many layers (Bengio et al., 2013) than other machine learning algorithms. In other words, if we make a model to learn better representation of data, the model can show better performance.

In natural language processing (NLP) tasks like language modeling (LM) (Bengio et al., 2003; Mikolov et al., 2013) and neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015), when the models are trained, they are to generate many words in sentence, which is a sequence of classification steps, for each of which they choose a target word among the whole words in the dictionary. That is why LM and NMT are usually trained with the sum of cross-entropies over the target sentence. Thus, although language related tasks are more of generation rather than classification, the models estimate target probabilities with the softmax operation on the previous neural network layers and the target distributions are provided as one-hot representations. As data representation in NLP models, word symbols should also be represented as vectors.

In this paper, we focus on the word embedding and the estimation of the target distribution. In NLP, word embedding is a step to translate word symbols (indices in the vocabulary) to vectors in a continuous vector space and is considered as a standard approach to handle symbols in neural networks. When two words have semantically or syntactically similar meanings, the words are represented closely to each other in a word embedding space. Thus, even when the prediction is not exactly correct, the predicted word might not be so bad, if the estimated word is very close to the target word in the embedding space like 'programming' and 'coding'. That is, to check how wrong the prediction is, the word embedding can be used. There are several methods to obtain word embedding matrices (Mikolov et al., 2013; Pennington et al., 2014), in addition to neural language models (Bengio et al., 2003; Mikolov et al., 2010). Recently, several approaches have been proposed to make more efficient word embedding matrices, usually based on contextual information (Søgaard et al., 2017; Choi et al., 2017).

On the other hand, knowledge distillation was proposed by (Hinton et al., 2015) to train new and usually shallow networks using hidden knowledge in the probabilities produced by the pretrained networks. It shows that there is knowledge not only in the target probability corresponding to the target class but also in the other class probabilities in the estimation of the trained model. In other words, the other class probabilities can contain additional information describing the input data samples differently even when the samples are in the same class. Also, samples from different classes could produce similar distributions to each other.

In this paper, we propose a new knowledge distillation method, *self-knowledge distillation* (SKD) based on the word embedding of the training model itself. That is, self-knowledge is distilled from the predicted probabilities produced by the training model, expecting the model has more information as it is more trained. In the conventional knowledge distillation, the knowledge is distilled from the estimated probabilities of pretrained (or teacher) models. Contrary, in the proposed SKD, knowledge is distilled from the current model in the training process, and the knowledge is hidden in the word embedding. During the training process, the word embedding reflects the relationship between words in the vector space. A word close to the target word in the vector space is expected to have similar distribution after softmax, and such information can be used to approximate the soft target probability as in knowledge distillation. We apply our proposed method to two popular NLP tasks: LM and NMT. The experiment results show that our proposed method improves the performance of the tasks. Moreover, SKD reduces overfitting problems which we believe is because SKD uses more information.

The paper is organized as follows. Background is reviewed in Section 2. In Section 3, we describe our proposed method, SKD. Experiment results are presented and analyzed in Section 4, followed by Section 5 with conclusion.

2 Background

In this section, we briefly review the cross-entropy and knowledge distillation. Also, since our proposed method is based on word embedding, the layer right before the softmax operation, word embedding process is summarized.

2.1 Cross Entropy

For classification with C classes, neural networks produce class probabilities p_i, $i \in \{0, 1, ...C\}$ by using a softmax output layer which calculates class probabilities from the logit, z_i considering the other logits as follows.

$$p_i = \frac{\exp{(z_i)}}{\sum_k \exp{(z_k)}}. \tag{1}$$

In most classification problems, the objective function for a single sample is defined by the cross-entropy as follows.

$$J(\theta) \quad = \quad -\sum_k y_k \log p_k, \tag{2}$$

where y_k and p_k are the target and predicted probabilities. The cross-entropy can be simply calculated by

$$J(\theta) \quad = \quad -\log p_t, \tag{3}$$

when the target probability $\boldsymbol{y}$ is a one-hot vector defined as

$$y_k = \begin{cases} 1, & \text{if } k = t(\text{target class}) \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

Note that the cross-entropy objective function says only how likely input samples belong to the corresponding target class, and it does not provide any other information about the input samples.

2.2 Knowledge Distillation

A well trained deep network model contains meaningful information (or knowledge) extracted from training datasets for a specific task. Once a deep model is trained for a task, the trained model can be used to train new smaller (shallower or thinner) networks as shown in (Hinton et al., 2015; Romero et al., 2014). This approach is referred to as *knowledge distillation*.

Basically, knowledge distillation provides more information to new models for training and improves the new model's performance. Thus, when a new model which is usually smaller is trained with the distilled knowledge from the trained deep model, it can achieve a similar (or sometimes even better) performance compared to the pretrained deep model.

In the pretrained model, knowledge lies in the class probabilities produced by softmax of the model as in Eq. (1). All probability values including the target class probability describe relevant information about the input data. Thus, instead of one-hot representation of the target label where only the target class is considered in cross-entropy, all probabilities over the whole classes from the pretrained model can provide more information about the input data in cross-entropy, and can teach new models more efficiently. All probabilities from the pretrained model are considered as *soft target probabilities*.

In a photo tagging task, depending on the other class probabilities, we understand the input image better than just target class. When a class 'mouse' has the highest probability, if 'mascot' has a relatively high probability, then the image would be probably 'mickey mouse'. If 'button' or 'pad' has a high probability, the image would be a mouse as a computer device. The other class probabilities have some extra information and such knowledge in the pretrained model can be transferred to a new model by using a soft target distribution of the training set.

When the target labels are available, the objective function is a weighted sum of the conventional cross-entropy with the correct labels and the cross-entropy with the soft target distribution, given by

$$J(\theta) = -(1-\lambda)\log p_t - \lambda \sum_k q_k \log p_k, \quad (5)$$

where p_k is probability for class k produced by current model with parameter θ, and q_k is the soft target probability from the pretrained model. λ controls the amount of knowledge from the trained model. Note that the conventional knowledge distillation extracts knowledge from a pretrained model, and in this paper, we propose to extract knowledge from the current model itself without any pretrained model.

Furthermore, in a recently proposed paper by (Furlanello et al., 2018), they proved that knowledge distillation can be useful to train a new model which has the same size and the same architecture as the pretrained model. They trained a teacher model first, then they trained a student model with distilled knowledge from the teacher model. Their experiment results show that the student models outperform the teacher model. Also, even though when the teacher model has a less powerful architecture, the knowledge from the trained teacher model can boost student models which have more powerful (or bigger) architectures. It means that even the knowledge is distilled from a relatively weak model, it can be useful to train a bigger model.

2.3 Word Embedding

Word embedding is to convert symbolic representation of words to vector representation with semantic and syntactic meanings, which reflects the relations between words. Including CBOW, Skip-gram (Mikolov et al., 2013), and GloVe (Pennington et al., 2014), various word embedding methods have been proposed to learn a word embedding matrix. The trained embedding matrix can be transferred to other models like LM or NMT (Ahn et al., 2016).

CBOW predicts a word given its neighbor words, and Skip-gram predicts the neighbor words given a word. They use feedforward layers, and the last layer of CBOW includes the word embedding matrix, $\boldsymbol{W}$, as follows.

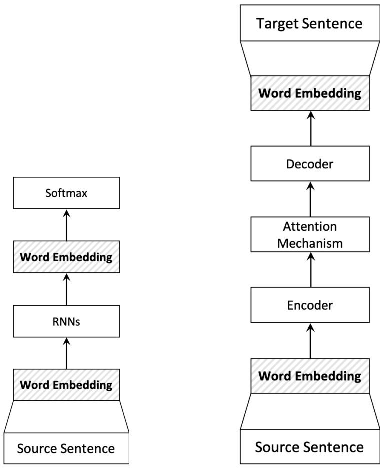
$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{h} + \boldsymbol{b}, \quad (6)$$

where $\boldsymbol{b}$ is a bias, $\boldsymbol{h}$ is hidden layer, and $\boldsymbol{z}$ is logits for the softmax operation.

Words in the embedding space have semantic and syntactic similarities, such that two similar words are close in the space. Thus, when the classification is not correct, the error can be interpreted differently depending on the similarity between the predicted word and the target word. For example, when the target word is 'learning', if the predicted word is 'training', then it is less wrong than other words like 'flower' or 'internet'. In this paper, we utilize such hidden information (or knowledge) in the word embedding space, while training. Fig. 1 shows where the word embedding is located in LM and NMT, respectively.

3 Self-Knowledge Distillation

We propose a new learning method *self-knowledge distillation* (SKD) which distills knowledge from a currently training model, following the conventional knowledge distillation. In this section, we describe an algorithm for SKD and its application to language model and neural machine translation.



(a) Language Model (b) NMT Model

Figure 1: Network architectures of LM and NMT. Word embedding is presented as gray boxes in the models.

3.1 SKD Equations

In order to apply knowledge distillation on a current training model, we need to obtain soft target probabilities as q_k in Eq. (5) for all classes, but they are not available explicitly. However, when the model is trained enough, then the word embedding has such information implicitly. If a word $\boldsymbol{w}_i$ is close to $\boldsymbol{w}_j$ in the embedding space, the probability p_i would be close to p_j for a given input sample.

When t is the target class, we calculate the soft target probabilities q_k based on the word embedding. First, we assume that q_t should be high, and if $\boldsymbol{w}_k$ is close to $\boldsymbol{w}_t$ in the embedding space, q_k should be also high. That is, the Euclidean distance between words is used to estimate the soft target probability. The other class probabilities (or soft target probabilities) q_k can be obtained by

$$q_k = \frac{1}{Z} \exp\{-\sigma \|\boldsymbol{w}_t - \boldsymbol{w}_k\|_2\}, \qquad (7)$$

where $\|\cdot\|_2$ is $l2$-norm, and Z is a normalization term. σ is a scale parameter and its value depends on the average distance to the corresponding nearest neighbors in the word embedding space. However, due to the expensive computational cost, we do not calculate q_k for all classes, and we choose just one of the other classes, which is the predicted class of the current model.

Assuming that the model predicts a class n for a given input sample, only q_t and q_n are used as distilled knowledge. We clip the q_n value with 0.5, meaning that the class n cannot be more correct than the real target t, so Eq. (7) becomes

$$\begin{aligned} q_n &= \min\{\exp\{-\sigma\|\boldsymbol{w}_t - \boldsymbol{w}_n\|_2\}, 0.5\}, \\ q_t &= 1 - q_n, \end{aligned} \qquad (8)$$

where $q_n + q_t = 1$. That is, we consider only two soft target probabilities as shown in Fig. 2. Note that we use Euclidean distance between $\boldsymbol{w}_t$ and $\boldsymbol{w}_n$ to calculate q_n, but other approaches like inner product would be possible.

Now, the objective function of SKD becomes similar to Eq. (5), and is defined by

$$\begin{aligned} J(\theta) &= -(1-\lambda)\log p_t \\ &\quad -\lambda(q_t \log p_t + q_n \log p_n), \end{aligned} \qquad (9)$$

where the second term of Eq. (5) is approximated by $\lambda(q_t \log p_t + q_n \log p_n)$, ignoring the other class probabilities. Eq. (9) can be rewritten simply as follows.

$$J(\theta) = -(1-\lambda q_n)\log p_t - \lambda q_n \log p_n. \qquad (10)$$

Eqs. (9) or (10) can be understood in three cases. First, if the prediction is correct ($n = t$), then Eq. (9) is the same as the conventional cross-entropy objective. Second, if $\boldsymbol{w}_n$ is far from $\boldsymbol{w}_t$ in the word embedding space, then q_n is close to zero and Eq. (9) becomes close to the conventional cross-entropy objective. Finally, if $\boldsymbol{w}_n$ is close to $\boldsymbol{w}_t$ (e.g. q_n = 0.4), it approximates the soft target probability with only two classes t and n, and the model is trained to produce probabilities for class t and n as close as q_t and q_n. This approach trains the model with different targets for different input samples.

Fig. 2 presents how SKD obtains simplified soft target distribution based on the distance of target and estimated vectors in the word embedding space.

3.2 SKD Algorithm

Since SKD distills knowledge from the current training model, at the beginning of the training process, the model does not contain relevant information. That is, we cannot extract any knowledge from the training model at the beginning. Thus, we start training process without knowledge distillation at first and gradually increase the amount of knowledge distillation as the training iteration goes. So, our algorithm starts with the conventional cross-entropy objective function in Eq. (3), and after training the model for a while, it gradually

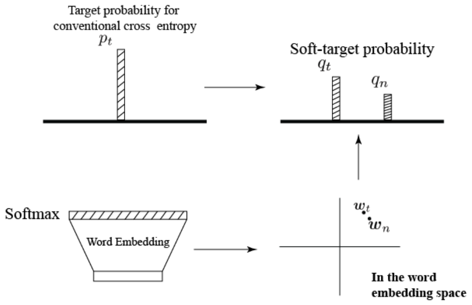

Target probability for conventional cross entropy
p_t
Soft-target probability
q_t
q_n
Softmax
Word Embedding
w_t
w_n
In the word embedding space

Figure 2: Given a target class t, a soft target probabilities are obtained based on the distance in the word embedding space. However, only the target class and the predicted class have soft target probabilities in SKD.

transits to Eq. (10). To implement the transition, another parameter α is introduced to Eq. (10), leading to the final objective function as follows.

$$J(\theta) = -(1 - \alpha\lambda q_n) \log p_t - \alpha\lambda q_n \log p_n, \quad (11)$$

α starts from 0 with which Eq. (11) becomes the conventional cross-entropy. After K iterations, α increases by η per iteration and eventually goes up to 1 with which Eq. (11) becomes the same as Eq. (9). In our experiments, we used a simplified equation as in Eq. (12) without λ so that the objective function relies gradually more on the soft target probabilities as training goes.

$$J(\theta) = -(1 - \alpha q_n) \log p_t - \alpha q_n \log p_n. \quad (12)$$

Table 1 summarizes the proposed SKD algorithm.

Table 1: Self-Knowledge Distillation Algorithm

```
Algorithm 1: SKD Algorithm

Initialize the model parameters $\theta$
Initialize $\alpha = 0$ and $\sigma$
(See the experiments for $\sigma$ values.)
Repeat $K$ times:
    Train the network based on the
    cross-entropy in Eq. (3)
Repeat until convergence:
    Train the network based on
    the SKD objective function in Eq. (12)
    Update $\alpha$ with $\alpha + \eta$
    (See the experiments for $\eta$ values.)
```

3.3 NLP Tasks

SKD is applied to two different NLP tasks: language modeling (LM), and neural machine translation (NMT). Although LM and NMT are actually sentence generation rather than classification, they have classification steps to generate words for the target sentence. Also, the sum of cross-entropies over the words in the sentence is adapted as an objective function for them.

In addition, to check if SKD is robust against errors in the word embedding space, we also evaluate SKD when we add Gaussian noise in the word embedding space for target words in the decoder.

4 Experiments

To evaluate self-knowledge distillation, we compare it to the baseline models for language modeling and neural machine translation.

4.1 Dataset

For language modeling, we use two different datasets: Penn TreeBank (PTB) and WiKi-2. PTB was made by (Marcus et al., 1993), and we use the pre-processed version by (Mikolov et al., 2010). In the PTB dataset, the train, valid and test sets have about 887K, 70K, and 78K tokens, respectively, where the vocabulary size is 10K. The WiKi-2 dataset introduced by (Merity et al., 2016) consists of sentences that are extracted from Wikipedia. It has about 2M, 217K, and 245K tokens for train, valid, and test sets. Its vocabulary size is about 33K. We did not apply additional pre-processing for the PTB dataset. The WiKi-2 dataset is pre-tokenized data, therefore we only added an end-of-sentence token (<EOS>) to every sentence.

For machine translation, we evaluated models on three different translation tasks (En-Fi, Fi-En, and En-De) with the available corpora from WMT'15 [1]. The dictionary size is 10K for En-fi and Fi-En translation task, and 30K for the En-De translation task.

4.2 Language Modeling

Language modeling (LM) has been used in many different NLP tasks like automatic speech recognition (ASR), and machine translation (MT) to capture syntactic and semantic structure of a natural language. The neural network-based language models (NNLM) and recurrent neural network language model (RNNLM) catch the syntactic and semantic regularities of an input language (Bengio et al., 2003; Mikolov et al., 2013). RNNLM is our baseline, which consists of a single LSTM layer and

[1] http://www.statmt.org/wmt15/translation-task.html

single feed forward layer with ReLU (Le et al., 2015).

We evaluate four models: Baseline, Noise (with Gaussian noise on the word embedding), SKD, and Noise+SKD. To show that the information by SKD is more knowledgeable than random noise, we tested a noise injected model which injects only Gaussian noise to the word embedding space. The word dimension is set to 500 and the number of hidden nodes is 400 for all models. We set the σ and η in the SKD algorithm in Table 1 0.1 (both PTB and WiKi-2 dataset) and 0.0002 (PTB), 0.00011 (WiKi-2), respectively. We applied the SKD object function after 500 batches for PTB and 900 batches for WiKi-2. Note that Wiki-2 data is larger than PTB.

The evaluation metric is the negative log-likelihood (NLL) for each sentence (the lower is the better). Table 2 presents NLLs for the test data of two datasets with different models. It shows that our proposed methods (both noise injection and self-distillation knowledge) improve the results in the LM task. Note that SKD provides more knowledgeable information than Gaussian noise.

Table 2: NLLs for LM with different models on PTB and Wiki-2.

Model	PTB	Wiki-2
Baseline	101.40	119.49
+Noise	101.28	118.70
+SKD	99.38	116.85
+Noise+SKD	**97.41**	**116.60**

4.3 Neural Machine Translation

NMT has been widely used in machine translation research, because of its powerful performance and end-to-end training (Sutskever et al., 2014; Bahdanau et al., 2015; Johnson et al., 2017). Attention-based NMT models consist of an encoder, a decoder, and the attention mechanism (Bahdanau et al., 2015), which is our baseline in this paper except for replacing GRU with LSTM and using BPE (Sennrich et al., 2016). The encoder takes the sequence of source words in the word embedding form. The decoder works in a similar way to LM, except the attention mechanism. See (Bahdanau et al., 2015) for NMT and the attention mechanism in detail.

In the experiments, we check how much SKD can improve model's performance using the simple baseline architecture. Since SKD modifies only the objective function, we believe that the improvement by SKD is regardless of model architectures.

Table 3 shows that our proposed method improves NMT performance by around 1 BLEU score. For qualitative comparison, some translation results are presented below. The overall quality of translation of the SKD model looks better than baseline model's. In other words, when the BLEU scores are similar, the sentences translated by the SKD model look better.

- (src) Hallituslähteen mukaan tämä on yksi monista ehdotuksista, joita harkitaan.
  (trg) A governmental source says this is one of the many proposals to be considered.
  (baseline) *According to government revenue, this is one of the many proposals that are being considered to be considered.*
  (SKD) *According to the government, this is one of the many proposals that are being considered.*

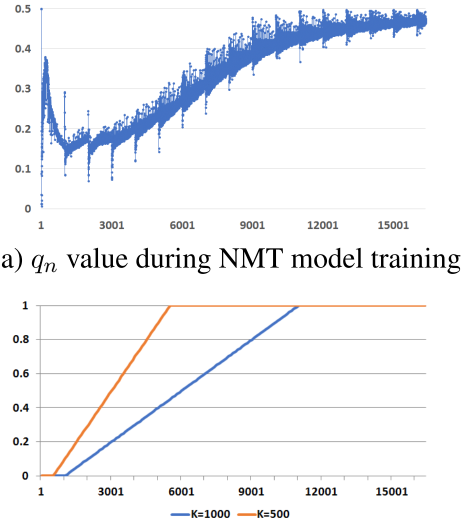
- (src) Meillä on hyvä tunne tuotantoketjunvahvuudesta.
  (trg) We feel very good about supply chain capability.
  (baseline) *We have good knowledge of the strength of the production chain.*
  (SKD) *We have a good sense of the strength of the production chain.*

- (src) En ole oikein tajunnut, että olen näin vanha.
  (trg) I haven't really realized that I'm this old.
  (baseline) *I have not been right to realise that I am so old.*
  (SKD) *I am not quite aware that I am so old.*

- (src) Ne vaikuttavat vasta tulevaisuudessa.
  (trg) They'll have an impact in the future only.
  (baseline) *They will only be affected in the future.*
  (SKD) *They will only affect in the future.*

Fig. 3 shows a trajectory of the q_n values and scheduling of the α value during training the En-Fi NMT model described in Eq. (12), respectively. As expected, the q_n value becomes larger than 0.5 which means that w_n (the predicted word vector) is close enough to the w_t (the target word vector). Fig. 3(b) shows the scheduled value of α in Eq. (12). The α value starts from 0 and increases up to 1 while training. The model is trained with only the cross-entropy for K iterations, and then when the model captures enough knowledge to be distilled, α increases to utilize knowledge from the model.

Also, as in Fig. 4, the SKD models are not (or more slowly) overfitted to the training data. We believe that SKD provides more information distilled by the training model itself to prevent overfitting. Note that there is no significant difference in the improvements by SKD and Noise, but Noise+SKD



(a) q_n value during NMT model training

(b) Scheduling of α value of NMT training

Figure 3: (a) Change of q_n value during NMT model training for En-Fi translation task, and (b) scheduling of α value in Eq. (12) of NMT training for En-Fi translation task. (a) shows that when the model is trained more, the q_n value become more close to the target.

improves further. It implies that SKD provides different kinds of information from noise, while the synergy effect between SKD and noise needs more research.

Table 3: BLEU scores on the test sets for En-Fi, Fi-En and En-De with two different beam widths. The scores on the development sets are in the parentheses.

Model	Beam width	
	1	12
En-Fi		
Baseline	7.29(8.28)	9.01(9.85)
+Noise	7.68(8.50)	9.35(9.53)
+SKD	8.36(**9.43)**	9.87(10.30)
+Noise+SKD	**8.81**(8.95)	**10.13(10.47)**
Fi-En		
Baseline	10.42(11.39)	11.89(12.78)
+Noise	10.74(11.80)	12.39(13.35)
+SKD	10.70(12.52)	12.43(13.82)
+Noise+SKD	**11.87(12.92)**	**13.16(14.13)**
En-De		
Baseline	19.72(19.28)	22.25(20.91)
+Noise	20.69(19.68)	22.40(20.92)
+SKD	20.29(20.41)	22.59(21.75)
+Noise+SKD	**21.16(20.34)**	**23.07(21.64)**

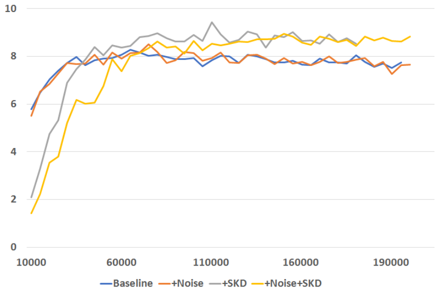


Figure 4: BLEU scores of validation data while training on En-Fi corpus with four different models: Baseline, +Noise, +SKD, and +Noise+SKD. The vertical axis indicates BLEU score and the horizontal axis the number of training iteration.

5 Conclusion

We proposed a new knowledge distillation method, self-knowledge distillation, from the probabilities of the currently training model itself. The method uses only two soft target probabilities that are obtained based on the word embedding space. The experiment results with language modeling and neural machine translation show that our method improves the performance. This method can be straightforwardly applied to other tasks where the cross-entropy is used.

As future works, we want to apply SKD to other applications with different model architectures, to show that SKD does not depend on tasks nor the model architectures. For image classification tasks, if we abuse the term ‘word embedding’ to refer to the layer right before the softmax operation, it may be possible to apply SKD in a similar way, although it is not guaranteed that comparable image classes are closely located in the word embedding space for image related tasks. Also, we can develop an automatic way for the parameters like α in Eq. (12), and generalize the equation for q_n in Eq. (8).

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2017R1D1A1B03033341), and by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. 2018-0-00749, Development of virtual network management technology based on artificial intelligence).

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR* abs/1608.00318:1–10.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. Int'l Conf. on Learning Representations (ICLR)*.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(8):1798–1828.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research* 3:1137–1155.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2017. Context-dependent word representation for neural machine translation. *Computer Speech and Language* 45:149–160.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. pages 1602–1611.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. pages 770–778.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL* 5:339–351.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR* abs/1504.00941.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR* abs/1609.07843.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. Int'l Conf. on Learning Representations (ICLR)*.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*. pages 1045–1048.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. pages 6517–6525.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *CoRR* abs/1412.6550.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *54th Annual Meeting of the Association for Computational Linguistics*. pages 1715–1725.

Anders Søgaard, Yoav Goldberg, and Omer Levy. 2017. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. pages 765–774.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. pages 6000–6010.

From the Paft to the Fiiture: a Fully Automatic NMT and Word Embeddings Method for OCR Post-Correction

Mika Hämäläinen[♠] Simon Hengchen[◇]

[♠] Department of Digital Humanities, University of Helsinki
[◇] COMHIS, University of Helsinki
firstname.lastname@helsinki.fi

Abstract

A great deal of historical corpora suffer from errors introduced by the OCR (optical character recognition) methods used in the digitization process. Correcting these errors manually is a time-consuming process and a great part of the automatic approaches have been relying on rules or supervised machine learning. We present a fully automatic unsupervised way of extracting parallel data for training a character-based sequence-to-sequence NMT (neural machine translation) model to conduct OCR error correction.

1 Introduction

Historical corpora are a key resource to study social phenomena such as language change in a diachronic perspective. Approaching this from a computational point of view is especially challenging as historical data tends to be noisy. The noise can come from OCR (optical character recognition) errors, or from the fact that the spelling conventions have changed as the time has passed, as thoroughly described by Piotrowski (2012).

However, depending on the NLP or DH task being modelled, some methods can cope with the noise in the data. Indeed, Hill and Hengchen (2019) use a subset of an 18th-century corpus, ECCO,[1] and its ground truth version, ECCO-TCP,[2] to compare the output of different common DH methods such as authorship attribution, count-based vector space models, and topic modelling, and report that those analyses produce statistically similar output despite noisiness due to OCR. Their conclusion is similar to Rodriquez et al. (2012) in the case of NER and to Franzini et al. (2018) in the case of authorship attribution, but different from Mutuvi et al. (2018) who, specifically on topic modelling for historical newspapers, confirm the often repeated trope of data too dirty to use. However, reducing the noise of OCRed text by applying a post-correction method makes it possible to gain the full potential of the data without having to re-OCR it and opens up the possibility to process it with the myriad of more precise NLP tools designed for OCR-error free text.

This paper focuses on correcting the OCR errors in ECCO. We present an unsupervised method based on the advances neural machine translation (NMT) in historical text normalization[3]. As NMT requires a parallel dataset of OCR errors and their corresponding correct spellings, we propose a method based on word embeddings, a lemma list, and a modern lemmatizer to automatically extract parallel data for training the NMT model.

2 Related Work

OCR quality for historical texts has recently received a lot of attention from funding bodies and data providers. Indeed, Smith and Cordell (2019) present a (USA-focused) technical report on OCR quality, and aim to spearhead the efforts on setting a research agenda for tackling OCR problems. Other initiatives such as Adesam et al. (2019) set out to analyse the quality of OCR produced by the Swedish language bank Språkbanken, Drobac et al. (2017) correct the OCR of Finnish newspapers using weighted finite-state methods, Tanner et al. (2009) measure mass digitisation in the context of British newspaper archives, while the Euro-

[1] Eighteenth Century Collections Online (ECCO) is a dataset which "contains over 180,000 titles (200,000 volumes) and more than 32 million pages", according to its copyright holder Gale: https://www.gale.com/primary-sources/eighteenth-century-collections-online.

[2] ECCO-TCP (Text Creation Partnership) "is a keyed subset of ECCO, compiled with the support of 35 libraries and made up of 2,231 documents". (Hill and Hengchen, 2019)

[3] Our code https://github.com/mikahama/natas

pean Commission-funded IMPACT project[4] gathers 26 national libraries and commercial providers to "take away the barriers that stand in the way of the mass digitization of the European cultural heritage" by improving OCR technology and advocating for best practices.

Dong and Smith (2018) present an unsupervised method for OCR post-correction. As opposed to our character-level approach, they use a word-level sequence-to-sequence approach. As such a model requires training data, they gather the data automatically by using repeated texts. This means aligning the OCRed text automatically with matched variants of the same text from other corpora or within the OCRed text itself. In contrast, our unsupervised approach does not require any repetition of text, but rather repetition of individual words.

Different machine translation approaches have been used in the past to solve the similar problem of text normalization, which means converting text written in a non-standard form of a language to the standard form in order to facilitate its processing with existing NLP tools. SMT (statistical machine translation) has been used previously, for instance, to normalize historical text (Pettersson et al., 2013) to modern language and to normalize modern Swiss German dialects (Samardzic et al., 2015) into a unified language form. More recently with the rise of the NMT, research has emerged in using NMT to normalize non-standard text, for example work on normalization of medieval German (Korchagina, 2017) and on historical English (Hämäläinen et al., 2018).

All of the normalization work cited above on using machine translation for normalization has been based on character-level machine translation. This means that words are split into characters and the translation model will learn to translate from character to character instead of word to word.

3 Model

As indicated by the related work on text normalization, character-level machine translation is a viable way of normalizing text into a standard variety. Therefore, we will also use character-level NMT in building our sequence-to-sequence OCR post-correction model. However, such a model requires parallel data for training. First, we will present our method of automatically extracting parallel data from our corpus containing OCR errors, then we will present the model designed to carry out the actual error correction.

3.1 Extracting Parallel Data

To extract a parallel corpus of OCR errors and their correctly spelled counterparts out of our corpus, we use a simple procedure consisting of measuring the similarity of the OCR errors with their correct spelling candidates. The similarity is measured in two ways, on the one hand an erroneous form will share a similarity in meaning with the correct spelling as they are realizations of the same word. On the other hand, an erroneous form is bound to share similarity on the level of characters, as noted by Hill and Hengchen (2019) in their study of OCR typically failing on a few characters on the corpus at hand.

In order to capture the semantic similarity, we use Gensim (Řehůřek and Sojka, 2010) to train a Word2Vec (Mikolov et al., 2013) model.[5] As this model is trained on the corpus containing OCR errors, when queried for the most similar words with a correctly spelled word as input, the returned list is expected to contain OCR errors of the correctly spelled word together with real synonyms, the key finding which we will exploit for parallel data extraction.

As an example to illustrate the output of the Word2Vec model, a query with the word *friendship* yields *friendlhip, friendihip, friendflip, friend-, affection, friendthip, gratitude, affetion, friendflhip* and *friendfiip* as the most similar words. In other words, in addition to the OCR errors of the word queried for, other correctly-spelled, semantically similar words (*friend-, affection* and *gratitude*) and even their erroneous forms (*affetion*) are returned. Next, we will describe our method (as shown in Algorithm 1) to reduce noise in this initial set of parallel word forms.

As illustrated by the previous example, we need a way of telling correct and incorrect spellings apart. In addition, we will need to know which incorrect spelling corresponds to which correct spelling (*affetion* should be grouped with *affection* instead of *friendship*).

For determining whether a word is a correctly spelled English word, we compare it to the lem-

[4] http://www.impact-project.eu

[5] Parameters: CBOW architecture, window size of 5, frequency threshold of 100, 5 epochs. Tokens were lowercased and no stopwords were removed.

mas of the Oxford English Dictionary (OED).[6] If the word exists in the OED, it is spelled correctly. However, as we are comparing to the OED lemmas, inflectional forms would be considered as errors, therefore, we lemmatize the word with spaCy[7] (Honnibal and Montani, 2017). If neither the word nor its lemma appear in the OED, we consider it as an OCR error.

For a given correct spelling, we get the most similar words from the Word2Vec model. We then group these words into two categories: correct English words and OCR errors. For each OCR error, we group it with the most similar correct word on the list. This similarity is measured by using Levenshtein edit distance (Levenshtein, 1966). The edit distances of the OCR errors to the correct words they were grouped with are then computed. If the distance is higher than 3 – a simple heuristic, based on ad-hoc testing –, we remove the OCR error from the list. Finally, we have extracted a small set of parallel data of correct English words and their different erroneous forms produced by the OCR process.

Algorithm 1: Extraction of parallel data

```
Draw $words\ w$ from the input word list;
for $w$ do
    Draw synonyms $s_w$ in the word embedding model
    for synonym $s_w$ do
        if $s_w$ is correctly spelled then
            Add $s_w$ to correct forms $forms_c$
        end
        else
            Add $s_w$ to error forms $forms_e$
        end
    end
    for error $e$ in $forms_e$ do
        group $e$ with the correct form in $forms_c$ by $Lev_{min}$
        if $Lev_{(e,c)} > 3$ then
            remove($e$)
        end
    end
end
```

We use the extraction algorithm to extract the parallel data by using several different word lists. First, we list all the words in the vocabulary of the Word2Vec model and list the words that are correctly spelled. We use this list of correctly spelled words in the model to do the extraction. However, as this list introduces noise to the parallel data, we combat this noise by producing another list of correctly spelled words that have occurred over 100,000 times in ECCO. For these two word lists, one containing all the correct words in the model and the other filtered with word frequencies, we produce parallel datasets consisting of words longer or equal to 1, 2, 3, 4 and 5. The idea behind these different datasets is that longer words are more likely to be matched correctly with their OCR error forms, and also frequent words will have more erroneous forms than less frequent ones.

source	all	>=2	>=3	>=4	>=5
W2V all	29013	28910	27299	20732	12843
W2V freq >100,000	11730	11627	10373	7881	5758
BNC	7692	7491	6681	5926	4925

Table 1: Sizes of the extracted parallel datasets

In addition, we use the frequencies from the British National Corpus (The BNC Consortium, 2007) to produce one more dataset of words occurring in the BNC over 1000 times to test whether the results can be improved with frequencies obtained from a non-noisy corpus. This BNC dataset is also used to produce multiple datasets based on the length of the word. The sizes of these automatically extracted parallel datasets are shown in Table 1.

3.2 The NMT Model

We use the automatically extracted parallel datasets to train a character level NMT model for each dataset. For this task, we use OpenNMT[8] (Klein et al., 2017) with the default parameters except for the encoder where we use a BRNN (bi-directional recurrent neural network) instead of the default RNN (recurrent neural network) as BRNN has been shown to provide a performance gain in character-level text normalization (Hämäläinen et al., 2019). We use the default of two layers for both the encoder and the decoder and the default attention model, which is the general global attention presented by Luong et al. (2015). The models are trained for the default number of 100,000 training steps with the

[6] http://www.oed.com.

[7] Using the en_core_web_md model.

[8] Version 0.2.1 of opennmt-py

source	all			>=2			>=3			>=4			>=5		
	Correct	False positive	No output	Correct	False positive	No output	Correct	False positive	No output	Correct	False positive	No output	Correct	False positive	No output
W2V all	0,510	0,350	0,140	0,500	0,375	0,125	0,520	0,325	0,155	0,490	0,390	0,120	0,525	0,390	0,085
W2V freq >100,000	0,515	0,305	0,180	0,540	0,310	0,150	0,510	0,340	0,150	0,540	0,315	0,145	0,515	0,330	0,155
BNC	**0,580**	0,285	0,135	0,555	0,300	0,145	0,570	**0,245**	0,185	0,550	0,310	0,140	0,550	0,315	0,135

Table 2: Results of the NMT models trained on different datasets

same seed value.

We use the trained models to do a character level translation on the erroneous words. We output the top 10 candidates produced by the model, go through them one by one and check whether the candidate word form is a correct English word (as explained in section 3.1). The first candidate that is also a correct English word is considered as the corrected form produced by the system. If none of the top 10 candidates is a word in English, we consider that the model failed to produce a corrected form. The use of looking at the top 10 candidates instead of the topmost candidates is motivated by the findings by Hämäläinen et al. (2019) in historical text normalization with a character-level NMT.

4 Evaluation

For evaluation, we prepare by hand a gold standard containing 200 words with OCR errors from the ECCO and their correct spelling. The performance of our models calculated as a percentage of how many erroneous words they were able to fix correctly. As opposed to the other common metrics such as character error rate and word error rate, we are measuring the absolute performance in predicting the correct word for a given erroneous input word.

Table 2 shows the results for each dataset. The highest accuracy of 58% is achieved by training the model with all of the frequent words in the BNC, and the lowest number of false positives (i.e. words that do exist in English but are not the right correction for the OCR error) is achieved by the model trained with the BNC words that are at least 3 characters long. The *No output* column shows the number of words the models didn't output any word for that would have been correct English.

If, instead of using NMT, we use the Word2Vec extraction method presented in section 3.1 to conduct the error correction by finding the semantically similar word with the lowest edit distance under 4 for an erroneous form, the accuracy of such a method is only 26%. This shows that training an NMT model is a meaningful part in the correction process.

In the spirit of Hämäläinen et al. (2018), whose results indicate that combining different methods in normalization can be beneficial, we can indeed get a minor boost for the results of the highest accuracy NMT model if we first try to correct with the above described Word2Vec method and then with NMT, we can increase the overall accuracy to 59.5%. However, there is no increase if we invert the order and try to first correct with the NMT and after that with the Word2Vec model.

5 Conclusion and Future Work

In this paper we have proposed an unsupervised method for correcting OCR errors. Apart from the lemma list and the lemmatizer, which can also be replaced by a morphological FST (finite-state transducer) analyzer or a list of word forms, this method is not language specific and can be used even in scenarios with less NLP resources than what English has. Although not a requirement, having the additional information about word frequencies from another OCR error-free corpus can boost the results.

A limitation of our approach is that it cannot do word segmentation in the case where multiple words have been merged together as a result of the OCR process. However, this problem is complex enough on its own right to deserve an entire publication of its own and is thus not in the scope of our paper. Indeed, previous research has been conducted focusing solely on the segmentation problem (Nastase and Hitschler, 2018; Soni et al., 2019) of historical text and in the future such methods can be incorporated as a preprocessing step for our proposed method.

It is in the interest of the authors to extend the approach presented in this paper on historical data written in Finnish and in Swedish in the immediate near future. The source code and the best working NMT model discussed in this paper has be made freely available on GitHub as a part of the natas Python library[9].

[9] https://github.com/mikahama/natas

Acknowledgements

We would like to thank the COMHIS group[10] for their support, as well as GALE for providing the group with ECCO data.

References

Yvonne Adesam, Dana Dannélls, and Nina Tahmasebi. 2019. Exploring the quality of the digital historical newspaper archive kubhist. *Proceedings of DHN*.

Rui Dong and David Smith. 2018. Multi-input attention for unsupervised OCR correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2363–2372.

Senka Drobac, Pekka Sakari Kauppinen, Bo Krister Johan Linden, et al. 2017. OCR and post-correction of historical finnish texts. In *Proceedings of the 21st Nordic Conference on Computational Linguistics, NoDaLiDa, 22-24 May 2017, Gothenburg, Sweden*. Linköping University Electronic Press.

Greta Franzini, Mike Kestemont, Gabriela Rotari, Melina Jander, Jeremi K Ochab, Emily Franzini, Joanna Byszuk, and Jan Rybicki. 2018. Attributing authorship in the noisy digitized correspondence of Jacob and Wilhelm Grimm. *Frontiers in Digital Humanities*, 5:4.

Mika Hämäläinen, Tanja Säily, Jack Rueter, Jörg Tiedemann, and Eetu Mäkelä. 2018. Normalizing early English letters to present-day English spelling. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 87–96, Santa Fe, New Mexico. Association for Computational Linguistics.

Mika Hämäläinen, Tanja Säily, Jack Rueter, Jörg Tiedemann, and Eetu Mäkelä. 2019. Revisiting NMT for normalization of early English letters. In *Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 71–75, Minneapolis, USA. Association for Computational Linguistics.

Mark J. Hill and Simon Hengchen. 2019. Quantifying the impact of dirty OCR on historical text analysis: Eighteenth Century Collections Online as a case study. *Digital Scholarship in the Humanities: DSH*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. *To appear*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proc. ACL*.

Natalia Korchagina. 2017. Normalizing medieval german texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 12–17.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, 8, pages 707–710.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Stephen Mutuvi, Antoine Doucet, Moses Odeo, and Adam Jatowt. 2018. Evaluating the impact of OCR errors on topic modeling. In *International Conference on Asian Digital Libraries*, pages 3–14. Springer.

Vivi Nastase and Julian Hitschler. 2018. Correction of OCR word segmentation errors in articles from the ACL collection through neural machine translation methods. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013. An SMT approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013; May 22-24; 2013; Oslo; Norway. NEALT Proceedings Series 18*, 087, pages 54–69. Linköping University Electronic Press.

Michael Piotrowski. 2012. Natural language processing for historical texts. *Synthesis lectures on human language technologies*, 5(2):1–157.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Kepa Joseba Rodriquez, Mike Bryant, Tobias Blanke, and Magdalena Luszczynska. 2012. Comparison of named entity recognition tools for raw OCR text. In *KONVENS*, pages 410–414.

Tanja Samardzic, Yves Scherrer, and Elvira Glaser. 2015. Normalising orthographic and dialectal variants for the automatic processing of Swiss German. In *Proceedings of the 7th Language and Technology Conference*.

[10] `https://www.helsinki.fi/en/researchgroups/computational-history`

David A. Smith and Ryan Cordell. 2019. A research agenda for historical and multilingual optical character recognition. Technical report, Northeastern University.

Sandeep Soni, Lauren Klein, and Jacob Eisenstein. 2019. Correcting whitespace errors in digitized historical texts. In *Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 98–103, Minneapolis, USA. Association for Computational Linguistics.

Simon Tanner, Trevor Muñoz, and Pich Hemy Ros. 2009. Measuring mass text digitization quality and usefulness. *D-lib Magazine*, 15(7/8):1082–9873.

The BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition). Distributed by Bodleian Libraries, University of Oxford, on behalf of the BNC Consortium. Http://www.natcorp.ox.ac.uk/.

Investigating Terminology Translation in Statistical and Neural Machine Translation: A Case Study on English-to-Hindi and Hindi-to-English

Rejwanul Haque,[†] Mohammed Hasanuzzaman[‡] and Andy Way[†]
ADAPT Centre
[†]School of Computing, Dublin City University, Dublin, Ireland
[‡]Department of Computer Science, Cork Institute of Technology, Cork, Ireland
firstname.lastname@adaptcentre.ie

Abstract

Terminology translation plays a critical role in domain-specific machine translation (MT). In this paper, we conduct a comparative qualitative evaluation on terminology translation in phrase-based statistical MT (PB-SMT) and neural MT (NMT) in two translation directions: English-to-Hindi and Hindi-to-English. For this, we select a test set from a legal domain corpus and create a gold standard for evaluating terminology translation in MT. We also propose an error typology taking the terminology translation errors into consideration. We evaluate the MT systems' performance on terminology translation, and demonstrate our findings, unraveling strengths, weaknesses, and similarities of PB-SMT and NMT in the area of term translation.

1 Introduction

Over the last five years, there has been incremental progress in the field of NMT (Bahdanau et al., 2015; Vaswani et al., 2017) to the point where some researchers are claiming parity with human translation (Hassan et al., 2018). Nowadays, NMT is regarded as a preferred alternative to PB-SMT (Koehn et al., 2003) and represents a new state-of-the-art in MT research. The rise of NMT has resulted in a swathe of research in the field of MT, unraveling the strengths, weaknesses, impacts and commercialisation aspects of the classical (i.e. PB-SMT) and emerging (i.e. NMT) methods (e.g. (Bentivogli et al., 2016; Toral and Way, 2018)). In brief, the NMT systems are often able to produce better translations than the PB-SMT systems. Interestingly, terminology translation, a crucial factor in industrial translation workflows (TWs), is one of the less explored areas in MT research. In this context, a few studies (Burchardt et al., 2017; Macketanz et al., 2017; Specia et al., 2017), with their focus on high-level evaluation, have indicated that NMT lacks effectiveness in translating domain terms compared to PB-SMT. In this work, we aim to compare PB-SMT and NMT in relation to terminology translation, by carrying out a thorough manual evaluation. For this, we select a test set from legal domain data (i.e. judicial proceedings), and create a gold standard evaluation test set following a semi-automatic terminology annotation strategy. We inspected the patterns of the term translation-related errors in MT. From our observations we make a high-level classification of the terminology translation-related errors and propose an error typology. We discuss various aspects of terminology translation in MT considering each of the types from the proposed terminology translation typology, and dig into the extent of the term translation problems in PB-SMT and NMT with statistical measures as well as linguistic analysis. For experimentation, we select a less examined and low-resource language pair, English–Hindi.

2 MT Systems

To build our PB-SMT systems we used the Moses toolkit (Koehn et al., 2007). For LM training we combine a large monolingual corpus with the target-side of the parallel training corpus. Additionally, we trained a neural LM with the NPLM toolkit (Vaswani et al., 2013) on the target side of the parallel training corpus alone. We considered the standard PB-SMT log-linear features for training. We call the English-to-Hindi and Hindi-to-English PB-SMT systems EHPS and HEPS, respectively. Our NMT systems are Google Transformer models (Vaswani et al., 2017). In our experiments we followed the recommended best set-up from Vaswani et al. (2017). We call our the English-to-Hindi and Hindi-to-English NMT systems EHNS and HENS, respectively.

For experimentation we used the IIT Bombay English-Hindi parallel corpus (Kunchukuttan et al., 2017). For building additional LMs for Hindi and English we use the HindEnCorp monolingual corpus (Bojar et al., 2014) and monolingual data from the OPUS project (Tiedemann, 2012), respectively. Corpus statistics are shown in Table 1. We selected 2,000 sentences (test set) for the evaluation of the MT systems and 996 sentences (development set) for validation from the Judicial parallel corpus (cf. Table 1) which is a juridical domain corpus (i.e. proceedings of legal judgments). The MT systems were built with the training set shown in Table 1 that includes the remaining sentences of the Judicial parallel corpus.

Table 1: Corpus Statistics.

English–Hindi parallel corpus			
	Sentences	Words (En)	Words (Hi)
Training set	1,243,024	17,485,320	18,744,496
(Vocabulary)		180,807	309,879
Judicial	7,374	179,503	193,729
Development set	996	19,868	20,634
Test set	2,000	39,627	41,249
Monolingual Corpus		Sentences	Words
Used for PB-SMT Language Model			
English		11M	222M
Hindi		10.4M	199M
Used for NMT Back Translation			
English		1M	20.2M
Hindi		903K	14.2M

We present the comparative performance of the PB-SMT and NMT systems in terms of BLEU score (Papineni et al., 2002) in Table 2. Additionally, we performed statistical significance tests using bootstrap resampling methods (Koehn, 2004). The confidence level (%) of the improvement obtained by one MT system with respect to the another MT system is reported. As can be seen from Table 2, EHPS and EHNS produce reasonable BLEU scores (28.8 BLEU and 36.6 BLEU) on the test set given the difficulty of the translation pair. These BLEU scores, in fact, underestimate the translation quality, given the relatively free word order in Hindi, as we have just a single reference translation set for evaluation. As far as the Hindi-to-English translation task is concerned, HEPS and HENS produce moderate BLEU scores (34.1 BLEU and 39.9 BLEU) on the test set. As expected, translation quality in the morphologically-rich to morphologically-poor language improves.

Table 2: Performance of MT systems on BLEU.

System	BLEU	System	BLEU
EHPS	28.8	HEPS	34.1
EHNS	**36.6** (99.9%)	HENS	**39.9** (99.9%)

3 Creating Gold Standard Evaluation Set

To evaluate terminology translation with our MT systems, we manually annotated the test set by marking term-pairs on the source- and target-sides of the test set (cf. Table 1) with a view to creating a gold standard evaluation set. The annotation process is performed using our own bilingual term annotation tool, *TermMarker*. If there is a source term present in the source sentence, its translation equivalent (i.e. target term) is found in the target sentence, and the source–target term-pair is marked. The annotators are native Hindi evaluators with excellent English skills. They were instructed to mark those words as terms that belong to legal or judicial domains. The annotators were also instructed to mark those sentence-pairs from the test set that contain errors (e.g. mistranslations, spelling mistakes) in either source or target sentences. The annotators reported 75 erroneous sentence-pairs which we discarded from the test set. In addition, 655 sentence-pairs of the test set did not contain any terms. We call the remaining 1,270 sentence-pairs our *gold-testset*. Each sentence-pair of gold-testset contains at least one aligned source-target term-pair. We have made the gold-testset publicly available to the research community.[1]

Annotation Suggestions from Bilingual Terminology While manually annotating bilingual terms in the judicial domain test set, we took support from a bilingual terminology that was automatically created from the Judicial corpus (cf. Table 1). For automatic bilingual term extraction we followed the approach of Haque et al. (2018). We found 3,064 English terms and their target equivalents (3,064 Hindi terms) in the source- and target-sides of gold-testset, respectively.

Variations of Term A term may have more than one domain-specific translation equivalent. The number of translation equivalents for a source term could vary from language to language depending on the morphological nature of the target language. For example, translation of the En-

[1] https://www.computing.dcu.ie/~rhaque/termdata/terminology-testset.zip

glish word ‘affidavit’ has multiple target equivalents (LIVs (lexical and inflectional variations)) in Hindi even if the translation domain is legal or juridical: ‘shapath patr’, ‘halaphanaama’, ‘halaphanaame’, or ‘halaphanaamo’. The term ‘shapath patr’ is the lexical variation of Hindi term ‘halaphanaama’. The base form ‘halaphanaama’ could have many inflectional variations (e.g. ‘halaphanaame’, ‘halaphanaamo’) given the sentence’s syntactic and morphological profile (e.g. gender, case).

For each term we check whether the term has any additional LIVs pertaining to the juridical domain and relevant to the context of the sentence. If this is the case, we include the relevant variations as legitimate alternatives term.

We again exploit the method of Haque et al. (2018) for obtaining variation suggestions for a term. The automatically extracted bilingual terminology of Haque et al. (2018) comes with the four highest-weighted target terms for a source term. If the annotator accepts an annotation suggestion (source–target term-pair) from the bilingual terminology, the remaining three target terms are considered as alternative suggestions of the target term.

Two annotators took part in the annotation task, and two sets of annotated data were obtained. The term-pairs of gold-testset are finalised on the basis of the annotation agreement by the two annotators, i.e. we keep those source–target term-pairs in gold-testset for which both annotators agree that the source and target entities are terms and aligned. On completion of the annotation process, inter-annotator agreement was computed using Cohen’s kappa (Cohen, 1960) at word-level. For each word we count an agreement whenever both annotators agree that it is a term (or part of term) or non-term entity. We found the kappa coefficient to be very high (i.e. 0.95) for the annotation task. This indicates that our terminology annotation is of excellent quality.

The final LIV list for a term is the union of the LIV lists created by the annotators. This helps make the resulting LIV lists exhaustive.

4 Terminology Translation Typology

In order to annotate errors in (automatic) translations, MT users often exploit the MQM (Multidimensional Quality Metric) error annotation framework (Lommel et al., 2014). One of the error types in the MQM toolkit is terminology (i.e. *inconsistent with termbase, inconsistent use of terminology*) which is an oversimplified attribute and does not consider various nuances of term translation errors. We propose an error typology taking terminology translation into consideration. First, we translated the test set sentences with our MT systems, and sampled 300 translations from the whole translation set. Then, the terminology translations were manually inspected, noting the patterns of the term translation-related errors. From our observations we found that the terminology translation-related errors can be classified into eight primary categories. As far as the term translation quality of an MT system is concerned, our proposed typology could provide a better perspective as to how the MT system lacks quality in translating domain terms. The categories are as follows: (i) reorder error (RE): the translation of a source term forms the wrong word order in the target, (ii) inflectional error (IE): the translation of a source term inflicts a morphological error, (iii) partial error (PE): the MT system correctly translates part of a source term into the target and commits an error for the remainder of the source term, (iv) incorrect lexical selection (ILS): the translation of a source term is an incorrect lexical choice, (v) term drop (TD): the MT system omits the source term in translation, (vi) source term copied (STC): a source term or part of it is copied verbatim to target, (vii) disambiguation issue in target (DIT): although the MT system makes a potentially correct lexical choice for a source term, its translation-equivalent does not carry the meaning of the source term, and (viii) other error (OE): there is an error in relation to the translation of a source term, whose category, however, is beyond all remaining error categories. The proposed terminology translation error typology is illustrated in Figure 1 (cf. Appendix A).

Apart from the above error categories, we have a class for a source term being correctly translated into the target, i.e. the MT system produces a correct translation (CT) for a source term. As pointed out in Section 3, we wanted to see how diverse an MT model can be in translating domain terms, and how close the translation of a source term can be to the reference terms or its LIVs or to what extent (e.g. syntactically and morphologically) it differs from them. For this reason, we divide the CT class into seven sub-classes, and define them below: (i) CT given the reference term (CTR): the translation of a source term is the reference term, (ii) CT given one of the LIVs (CTV): the translation of a source is one of the LIVs of the reference term, (iii) variation missing (VM): a source term is correctly translated into the target, but the translation is neither the reference term nor any of its LIVs, (iv) correct inflected form (CIF): a source term is correctly translated into the target, but the translation is neither the reference term nor any of its LIVs. However, the base form of the translation of the source term is identical to the base form of either the reference term or one of the LIVs of the reference term, (v) correct reorder form (CRF): a source term is correctly translated into the target, and the translation includes those words that either the reference term or one of the LIVs has, but the word order of the translation is different to that of the the reference term or one of the LIVs, (vi) correct reorder and inflected form (CRIF): this class is a combination of both CIF and CRF, and (vii) other correct (OC): a source term is correctly translated into the target, whose category, however, is beyond the all remaining correct categories.

5 Manual Evaluation Plan

This section presents our manual evaluation plan. Translations of the source terms of gold-testset were manually validated and classified in accordance with the set of fine-grained errors and correct categories described above. This was accomplished by the human evaluator. The manual evaluation was carried out with a GUI that randomly displays a source sentence and its reference translation from gold-testset, and the automatic translation by one of the MT systems. For each source term the GUI highlights the source term and the corresponding reference term from the source and reference sentences, respectively, and displays the LIVs of the reference term, if any. The GUI lists the error and correct categories described in Section 4. The evaluator, a native Hindi speaker with the excellent English and Hindi skills, was instructed to follow the following criteria for evaluating the translation of a source term: (a) judge correctness / incorrectness of the translation of the source term in hypothesis and label it with an appropriate category listed in the GUI, (b) do not need to judge the whole translation, but instead look at the local context to which both source term and its translation belong, and (c) take the syntactic and morphological properties of the source term and its translation into account.

The manual classification process was completed for all MT system types. We measure agreement in manual classification of terminology translation. For this, we randomly selected an additional 100 segments from gold-testset and hired another evaluator having the similar skills. We considered the correct and incorrect categories for the calculation, i.e. we count an agreement whenever both evaluators agree that it is a correct (or incorrect) term translation, with agreement by chance = 1/2. We found that the kappa coefficient for this ranges from 0.97 to 1.0. Thus, our manual term translation classification quality can be labeled as excellent.

6 Terminology Translation Evaluation in PB-SMT and NMT

This section provides a comparative evaluation of the ability of PB-SMT and NMT to translate terminology accurately. In Table 5, we report the statistics of terminology translations from the English-to-Hindi MT task. We see that EHPS and EHNS incorrectly translate 303 and 253 English terms (out of total 3,064 terms) (cf. last row of Table 5), respectively, into Hindi, resulting in 9.9% and 8.3% terminology translation errors, respectively. We use approximate randomization (Yeh, 2000) to test the statistical significance of the difference between two systems, and report the significance-level (*p*-value) in the last column of Table 5. We found that the difference between the error rates is statistically significant. In Table 6, we report the statistics of terminology translations for the Hindi-to-English MT task. We see that HEPS and HENS incorrectly translate 396 and 353 Hindi terms (cf. last row of Table 6), respectively, into English, resulting in 12.9% and 11.5% terminology translation errors, respectively. As can be seen from Table 6, the difference between the error rates is statistically significant. When we compare these scores with those from Table 5, we see that these scores are slightly higher compared to those for the English-to-Hindi task. Surprisingly, the terminology translation quality from the morphologically-rich to the morphologically-poor language deteriorates compared to the overall MT quality (cf. Section 2).

6.1 Comparison with Fine-Grained Category

This section discusses the numbers and highlights phenomena for the fine-grained categories, starting with those that involve correct terminology translations.

CTV & VM We see from Tables 5 and 6 that the numbers under the CTV (correct term given one of the LIVs class are much higher in the English-to-Hindi task (695 and 662) compared to those in the Hindi-to-English task (241 and 245). CTV is measured as the count of instances where a source term is (i) correctly translated into the target translation and (ii) the translation-equivalent of that term is one of the LIVs of the reference term. As can be seen from Table 1, the training set vocabulary size is much higher in Hindi compared to that in English since the former is a morphologically-rich and highly inflected language, which is probably the reason why these numbers are much higher in the English-to-Hindi task.

In a few cases, the human evaluator found that the source terms are correctly translated into the target, but the translations are neither the reference terms nor any of its LIVs. The manual evaluator marked those instances with VM (variation missing) (cf. Tables 5 and 6). These can be viewed as annotation mistakes since the annotator omitted to add relevant LIVs for the reference term into gold-testset. In future, we aim to make gold-testset as exhaustive as possible by adding missing LIVs for the respective reference terms.

CRF, CIF, CRIF & OC We start this section by highlighting the problem of word order in term translation, via a translation example from gold-testset. The Hindi-to-English NMT system correctly translates a Hindi source term 'khand nyaay peeth ke nirnay' (English reference term: 'division bench judgment') into the following target translation (English): "it shall also be relevant to refer to article 45 - 48 of the *judgment of the division bench*". The manual evalautor marks this term translation as CRF (correct reorder form) since the term *'judgment of the division bench'* was not in the LIV list for the reference term, 'division bench judgment'.

We show another example from the Hindi-to-English translation task. This time, we highlight the issue of inflection in term translation. As an example, we consider a source Hindi term 'abhikathan' from gold-testset. Its reference term is 'allegation', and the LIV list of the reference term includes two lexical variations for 'allegation': 'accusation' and 'complaint'. A portion of the reference translation is 'an allegation made by the respondent ...'. A portion of the translation produced by the Hindi-to-English NMT system is 'it was *alleged* by the respondent ...'. In this translation, we see the Hindi term 'abhikathan' is translated into 'alleged' which is a correct translation of the Hindi legal term 'abhikathan' as per the syntax of the target translation. As above, the manual evalautor marked these term translations as CIF (correct inflected form) since the translation-equivalent of this term is not found in the LIV list of the reference term.

As stated in Section 4, CRIF (correct reorder and inflected form) is the combination of the above two types: CRF and CIF. As an example, consider a portion of the source Hindi sentence '*vivaadagrast vaseeyat* hindee mein taip kee gaee hai ...' and the English reference translation 'the *will in dispute* is typed in hindi ...' from gold-testset. Here, 'vivaadagrast vaseeyat' is a Hindi term and its English equivalent is 'will in dispute'. The translation of the source sentence by the Hindi-to-English NMT system is 'the *disputed will* have been typed in hindi ...'. We see that the translation of the source term ('vivaadagrast vaseeyat') is 'disputed will' which is correct. We also see that its word order is different to that of the reference term ('will in dispute'); and the morphological form of (part of) the translation is not identical to that of (part of) the reference term. As is the case with CRF and CIF, the manual evaluator marks such term translations as CRIF.

When translation of a source term is correct but its category is beyond the all remaining correct categories, the manual evaluator marks that term translation as OC (other correct). In our manual evaluation task, we encountered various such phenomena, and detail some of those below. (1) term transliteration: the translation-equivalent of a source term is the transliteration of the source term itself. We observed this happening only when the target language is Hindi. In practice, many English terms (transliterated form) are often used in Hindi text (e.g. 'decree' as 'dikre', 'tariff orders' as 'tarif ordars'), (2) terminology translation coreferred: translation-equivalent of a source term is not found in the hypothesis, however, it is correctly coreferred in target translation, and (3) semantically coherent terminology translation: the translation-equivalent of a source term is not seen in the hypothesis, but its meaning is correctly transferred into the target. As an example, consider the source Hindi sentence "sabhee apeelakartaon ne *aparaadh sveekaar nahin* kiya aur muqadama chalaaye jaane kee maang kee", and reference English sentence "all the appellants pleaded not guilty to the charge and claimed to be tried" from gold-testset.[2] Here, 'aparaadh sveekaar nahin' is a Hindi term and its English translation is 'pleaded not guilty'. The Hindi-to-English NMT system produces the following English translation "all the appellants did not accept the crime and sought to run the suit" for the source sentence. In this example, we see the meaning of

[2] In this example, the reference English sentence is the literal translation of the source Hindi sentence.

the source term 'aparaadh sveekaar nahin' is preserved in the target translation.

Table 3: CIF, CRF, CRIF and OC in PB-SMT and NMT.

	PB-SMT	NMT
English-to-Hindi	122 (4%)	98 (3.2%)
Hindi-to-English	90 (2.9%)	138 (4.5%)

We recall the rule that we defined while forming the LIV list for a reference term from Section 3. Our annotators considered only those inflectional variations for a reference term that would be grammatically relevant to the context of the reference translation in which they would appear. In practice, translation of a source sentence can be generated in numerous ways. It is possible that a particular inflectional variation of a reference term could be grammatically relevant to the context of the target translation, which, when it replaces the reference term in the reference translation, may (syntactically) misfit the context of the reference translation. As far as CRF and CRIF are concerned, a similar story might be applicable to the translation of a multiword term. A multiword term may be translated into the target in various ways (as shown above, 'division bench judgment' as 'judgment of the division bench', and 'disputed will' as 'will in dispute'). In reality, it would be an impossible task for the human annotator to consider all possible such variations for a multiword reference term. Additionally, as above, we saw more diverse translations with the domain terms under the OC category. In Table 3, we report the combined numbers under the above categories (CRF, CRIF, CIF and OC), with their percentage with respect to the total number of terms. We see that translations of a notable portion of source terms in each translation task are diverse. Therefore, investigating the automation of the terminology translation evaluation process (Haque et al., 2019), these phenomena have to be taken into consideration.

RE Now, we turn our focus to the error classes, starting with RE (reordering error). We compare the results under RE from Tables 5 and 6, and we see that NMT commits many fewer terminology translation-related reordering errors than PB-SMT. 15 REs are caught in the English-to-Hindi PB-SMT task compared to 5 in the English-to-Hindi NMT task. The same trend is observed with the reverse direction, with 18 reordering errors seen in the Hindi-to-English PB-SMT task compared to 5 in the Hindi-to-English NMT task. As can be seen from the last columns of Tables 5 and 6, the differences in these numbers in PB-SMT and NMT are statistically significant.

IE As far as the inflectional error type is concerned, the Hindi-to-English PB-SMT system makes nearly twice as many mistakes as the Hindi-to-English NMT system (118 vs 76) (cf. Tables 5 and 6), which is statistically significant. We see a different picture in the English-to-Hindi direction, i.e. the numbers of morphological errors are nearly the same, both in PB-SMT and NMT (77 vs 79). We found no statistically significant difference between them.

PE The numbers (cf. Tables 5 and 6) of partial term translation errors in PB-SMT and NMT are almost the same regardless of the translation directions. We found that the differences in these numbers are not statistically significant.

ILS PB-SMT appears to be more error-prone than NMT as far as a term's lexical selection is concerned. EHPS commits 77 incorrect lexical choices which is 35 more than EHNS. The same trend is observed with the Hindi-to-English direction. HEPS and HENS commit 139 and 90 incorrect lexical choices, respectively. We found that the differences in these numbers in PB-SMT and NMT are statistically significant.

TD Comparing the numbers of the term drop category from Tables 5 and 6, we see that the numbers of term omission by the PB-SMT and NMT systems are almost the same (53 versus 56) in the English-to-Hindi translation task. We found no statistically significant difference in these numbers. In contrast, in the Hindi-to-English translation task, HENS drops terms more than twice as often as HEPS (86 versus 38). This time, we found that the difference in these numbers is statistically significant.

STC & OE Now we focus on discussing various aspects with the STC (source term copied) and OE (other error) classes, starting with the English-to-Hindi task. We counted the number of source terms of gold-testset that are not found in the source-side of the training corpus (cf. Table 1). We see that 88 source terms (out of a total of 3,064 terms) are not found in the training data, with almost all being multiword terms. Nevertheless, only 5 unique words (i.e. adjudicary, halsbury, presuit, decretal, adj) that are either single-word terms or words of multiword terms are not found in the training data. In other words, these are out-of-vocabulary (OOV) items.

Table 4: STC in English-to-Hindi PB-SMT and NMT.

STC (PB-SMT)	translation (NMT)	class (NMT)
adjudicatory role	nyaay - nirnay keea koee bhoomika	PE
decretal		TD
halsbury 's laws	halbury ke kaanoonon	PE
presuit	poorva vaad	RE
adjudicatory		TD
learned **adj**	vidvat edeeje	CTR
learned **adj**	kaabil edeeje	CTV
mrtp act	mrtp adhiniyam	CTR
testatrix	testrex	OE
concealments	rahasyon	OE
res **judicata**	nyaayik roop	OE
subjudice	vichaaraadheen	CTV

We recall Table 5 where we see that the manual evaluator has marked 12 term translations with STC since in those cases the PB-SMT system copied source terms (or a part of source terms) verbatim into the target. In Table 4, we show those source terms in the PB-SMT task that belong to the STC class. The first column of the table shows source terms with the term itself or part of it in bold, which means those words are copied verbatim into target. We see from the table that the OOV terms (i.e. adjudicary, halsbury, presuit, decretal, adj), in most cases, are responsible for the term translations being marked with the STC tag. In one instance we found that a part of the English term ('mrtp') (cf. row 8 of Table 4) itself was present in the target-side of the training corpus. This could be the possible reason why 'mrtp' is seen in the target translation. Each of the remaining source terms (last 4 rows of Table 4) include words that are copied directly into the target translation despite the fact that they are not OOVs. This is a well-known problem in PB-SMT and rarely happens with the low frequency words of the training corpus. In short, these source terms (last 4 entries of Table 4) either alone or with the adjacent words of the test set sentences (i.e. as a part of phrase) are not found in the source-side of the PB-SMT phrase table.

Now we see how NMT performed with the 12 source terms above; their translations with EHNS and the corresponding manual class are shown in the second and third columns of Table 4, respectively. We see that out of 12 translations EHNS made a mistake on 8 occasions and correctly translated on 4 occasions. The errors are spread over different categories (e.g. TD, OE, PE). Unsurprisingly, we see NMT is capable of correctly translating rare and even unknown words, by exploiting the strength of the open-vocabulary translation technique (Sennrich et al., 2016). However, this method also has down-sides. For example, some of the term translations under the OE category in the NMT task are non-existent wordforms of the target language, for which the open-vocabulary translation technique is responsible. This phenomenon is also corroborated by Farajian et al. (2017) while translating technical domain terms. We discuss the OE class further below.

We see from Table 5 that the human evaluator has marked 24 term translations with OE in NMT. In this category we observed that the translations of the source terms are usually either strange words that have no relation to the meaning of the source term, repetitions of other translated words or terms, entities that are non-existent wordforms of the target language, or words with typographical errors. As far as PB-SMT is concerned, we see from Table 5 that the evaluator also tagged 12 term translations with OE, most of which are related to typographical errors.

Now we turn our focus on the Hindi-to-English task. We counted the number of those source terms from gold-testset that are not found in the source-side (Hindi) of the training corpus (cf. Table 1). We see that 160 source terms (out of a total of 3,064 terms) are not found in the training data, most of which are, in fact, multiword terms. However, only 18 unique Hindi words that are either single-word terms or words within multiword terms are not found in the training data. As in English-to-Hindi translation task, in this task we found that the OOV items are largely responsible for the term translations being marked as STC. We also examined how the Hindi-to-English NMT system performed with those 17 source terms that were marked as STC. We see that HENS makes a mistake on 13 occasions and correctly translates on 4 occasions. The error types are spread over different categories: TD (2), OE (6), PE (1) and ILS (4). We observed that 3 out of 4 source terms of the STC category for which the Hindi-to-English NMT system produces correct translations are OOV items. Here, we again see the strength of the open-vocabulary translation technique for the translation of novel terms. In the Hindi-to-English translation task, we found that the terminology translations under the OE category, as in English-to-Hindi translation, are roughly related to odd translations, non-existent wordforms of the target language, typological mistakes and repetition of other translated words or terms.

DIT We see from Table 5 and Table 6 that the manual evaluator marked 3 and 1 term translations as DIT (disambiguation issue in target) in English-to-Hindi and Hindi-to-English PB-SMT tasks, respectively. We found that the MT systems made correct lexical choices for the source terms, although the meanings of their target-equivalents in the respective translations are different to those of the source terms. This can be viewed as a cross-lingual disambiguation problem. For example, one of the three source terms from English-to-Hindi translation task is 'victim' (reference translation 'shikaar') and the English-to-Hindi PB-SMT system makes a correct lexical choice ('shikaar') for 'victim', although the meaning of 'shikaar' is completely different in the target translation, i.e. here, its meaning is equivalent to English 'hunt'.

Pairwise Overlap We report the numbers of pairwise overlaps, i.e. the number of instances in which NMT and PB-SMT have identical classification outcomes. We recall Table 5 & 6 whose fourth columns show the numbers of pairwise overlap for categories. The small number of overlapping instances in each category indicates that term translation errors from the PB-SMT system are quite different from those from the NMT system. As can be seen from the last row of Table 5 & 6, the numbers of overlaps in the combination of all error classes are 86 and 115, respectively, which are nearly one third or fourth of the number of errors committed by the NMT and PB-SMT systems alone, indicating that the majority of the errors in PB-SMT are complementary with those in NMT. This finding on terminology translation is corroborated by Popović (2017), who finds complementarity with the various issues relating to the translations of NMT and PB-SMT.

7 Conclusion and Future Work

In this paper, we investigated domain term translation in PB-SMT and NMT with two morphologically divergent languages, English and Hindi. Due to the unavailability of a gold standard for term translation evaluation, we adopted a technique that semi-automatically creates a gold standard test set from an English–Hindi judicial domain parallel corpus. The gold standard that we have developed will serve as an important resource for the evaluation of term translation in MT. We also propose a terminology translation typology focused on term translation errors in MT. From our evaluation results, we found that the NMT systems commit fewer lexical, reordering and morphological errors than the PB-SMT systems. The differences in error rates of the former (lexical selection and reordering errors) types are statistically significant in both MT tasks, and the difference of the morphological error rates is statistically significant in the Hindi-to-English task. The morphological errors are seen relatively more often in PB-SMT than in NMT when translation is performed from a morphologically-rich language (Hindi) to the a morphologically-poor language (English). The opposite picture is observed in the case of term omission in translation, with NMT omitting more terms in translation than PB-SMT. We found that the difference in term omission-related error rates in PB-SMT and NMT are statistically significant in the Hindi-to-English task, i.e. again from the morphologically-rich language to the morphologically-poor language. Another important finding from our analysis is that NMT is able to correctly translate unknown terms, by exploiting the strength of the open-vocabulary translation technique, which, as expected, are copied verbatim into the target in PB-SMT. We also found that the majority of the errors made by the PB-SMT system are complementary to those made by the NMT system. In NMT, we observed that translations of source terms are occasionally found to be strange words that have no relation to the source term, non-existent wordforms of the target language, and/or repetition of other translated words. This study also shows that a notable portion of the term translations by the MT systems are diverse, which needs to be taken into consideration while investigating the automation of the terminology translation evaluation process.

As far as future work is concerned, we plan to test terminology translation with different language pairs and domains.

Acknowledgments

The ADAPT Centre for Digital Content Technology is funded under the Science Foundation Ireland (SFI) Research Centres Programme (Grant No. 13/RC/2106) and is co-funded under the European Regional Development Fund. This project has partially received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713567, and the publication has emanated from research supported in part by a research grant from SFI under Grant Number 13/RC/2077.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267, Austin, Texas.

Bojar, O., Diatka, V., Rychlý, P., Straňák, P., Suchomel, V., Tamchyna, A., and Zeman, D. (2014). HindEnCorp – Hindi-English and Hindi-only corpus for machine translation. In *Proceedings of the Ninth International Language Resources and Evaluation Conference (LREC'14)*, pages 3550–3555, Reykjavik, Iceland.

Burchardt, A., Macketanz, V., Dehdari, J., Heigold, G., Peter, J.-T., and Williams, P. (2017). A linguistic evaluation of rule-based, phrase-based, and neural mt engines. *The Prague Bulletin of Mathematical Linguistics*, 108(1):159–170.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Farajian, M. A., Turchi, M., Negri, M., Bertoldi, N., and Federico, M. (2017). Neural vs. phrase-based machine translation in a multi-domain scenario. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 280–284, Valencia, Spain.

Haque, R., Hasanuzzaman, M., and Way, A. (2019). TermEval: An automatic metric for evaluating terminology translation in MT. In *Proceedings of CICLing 2019, the 20th International Conference on Computational Linguistics and Intelligent Text Processing*, La Rochelle, France.

Haque, R., Penkale, S., and Way, A. (2018). TermFinder: log-likelihood comparison and phrase-based statistical machine translation models for bilingual terminology extraction. *Language Resources and Evaluation*, 52(2):365–400.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., Liu, S., Liu, T., Luo, R., Menezes, A., Qin, T., Seide, F., Tan, X., Tian, F., Wu, L., Wu, S., Xia, Y., Zhang, D., Zhang, Z., and Zhou, M. (2018). Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In Lin, D. and Wu, D., editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., College, W., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *HLT-NAACL 2003: conference combining Human Language Technology conference series and the North American Chapter of the Association for Computational Linguistics conference series*, pages 48–54, Edmonton, AB.

Kunchukuttan, A., Mehta, P., and Bhattacharyya, P. (2017). The IIT Bombay English–Hindi parallel corpus. *CoRR*, 1710.02855.

Lommel, A. R., Uszkoreit, H., and Burchardt, A. (2014). Multidimensional Quality Metrics (MQM): A framework for declaring and describing translation quality metrics. *Tradumàtica: tecnologies de la traducció*, (12):455–463.

Macketanz, V., Avramidis, E., Burchardt, A., Helcl, J., and Srivastava, A. (2017). Machine translation: Phrase-based, rule-based and neural approaches with linguistic evaluation. *Cybernetics and Information Technologies*, 17(2):28–43.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL-2002: 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA. ACL.

Popović, M. (2017). Comparing language related issues for nmt and pbmt between German and English. *The Prague Bulletin of Mathematical Linguistics*, 108(1):209–220.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Specia, L., Harris, K., Blain, F., Burchardt, A., Macketanz, V., Skadiņa, I., Negri, M., and Turchi, M. (2017). Translation quality and productivity: A study on rich morphology languages. In *Proceedings of MT Summit XVI, the 16th Machine Translation Summit*, pages 55–71, Nagoya, Japan.

Tiedemann, J. (2012). Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'2012)*, pages 2214–2218, Istanbul, Turkey.

Toral, A. and Way, A. (2018). What level of quality can neural machine translation attain on literary text? In *Translation Quality Assessment*, pages 263–287. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics - Volume 2, COLING 2000*, pages 947–953, Saarbrücken, Germany.

A Supplementary Material

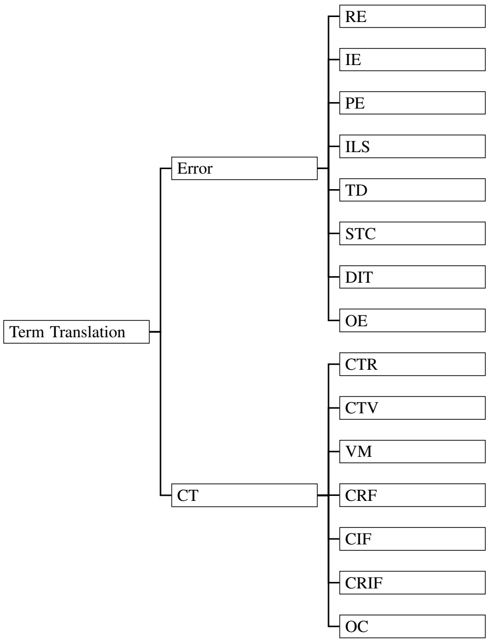
Table 5: PB-SMT vs NMT: English-to-Hindi.

	PB-SMT	NMT	∩	p-value
CTR	1,907	2,015	1662	
CTV	695	662	466	
VM	35	36	10	
CRF	4	7	4	
CIF	112	87	31	
CRIF				
OC	8	4		
CT	**2,761**	**2,811**	**2614**	
RE	15	5		0.044
IE	79	77	30	0.91
PE	52	47	19	0.61
ILS	77	44	9	0.001
TD	53	56	9	0.83
STC	12			
OE	12	24	2	
DIT	3			
ERROR	**303**	**253**	**86**	**0.011**

Table 6: PB-SMT vs NMT: Hindi-to-English.

	PB-SMT	NMT	∩	p-value
CTR	2,313	2,295	2,075	
CTV	241	245	147	
VM	24	33	5	
CRF	13	11	4	
CIF	75	107	48	
CRIF		2		
OC	2	18		
CT	**2,668**	**2,711**	**2,483**	
RE	18	5	1	0.008
IE	118	76	21	0.0009
PE	65	73	31	0.42
ILS	139	90	35	0.0001
TD	38	86	6	0.0001
STC	17			
OE		23		
DIT	1			
ERROR	**396**	**353**	**115**	**0.04**

Figure 1: Terminology Translation Typology.



Beyond English-Only Reading Comprehension: Experiments in Zero-Shot Multilingual Transfer for Bulgarian

Momchil Hardalov[1] **Ivan Koychev**[1] **Preslav Nakov**[2]
[1]Sofia University “St. Kliment Ohridski”, Bulgaria,
[2]Qatar Computing Research Institute, HBKU, Qatar,
{hardalov, koychev}@fmi.uni-sofia.bg
pnakov@hbku.edu.qa

Abstract

Recently, reading comprehension models achieved near-human performance on large-scale datasets such as SQuAD, CoQA, MS Macro, RACE, etc. This is largely due to the release of pre-trained contextualized representations such as BERT and ELMo, which can be fine-tuned for the target task. Despite those advances and the creation of more challenging datasets, most of the work is still done for English. Here, we study the effectiveness of multilingual BERT fine-tuned on large-scale English datasets for reading comprehension (e.g., for RACE), and we apply it to Bulgarian multiple-choice reading comprehension. We propose a new dataset containing 2,221 questions from matriculation exams for twelfth grade in various subjects —history, biology, geography and philosophy—, and 412 additional questions from online quizzes in history. While the quiz authors gave no relevant context, we incorporate knowledge from Wikipedia, retrieving documents matching the combination of question + each answer option. Moreover, we experiment with different indexing and pre-training strategies. The evaluation results show accuracy of 42.23%, which is well above the baseline of 24.89%.

1 Introduction

The ability to answer questions is natural to humans, independently of their native language, and, once learned, it can be easily transferred to another language. After understanding the question, we typically depend on our background knowledge, and on relevant information from external sources. Machines do not have the reasoning ability of humans, but they are still able to learn concepts. The growing interest in teaching machines to answer questions posed in natural language has led to the introduction of various new datasets for different tasks such as reading comprehension, both extractive, e.g., span-based (Nguyen et al., 2016; Trischler et al., 2017; Joshi et al., 2017; Rajpurkar et al., 2018; Reddy et al., 2019), and non-extractive, e.g., multiple-choice questions (Richardson et al., 2013; Lai et al., 2017; Clark et al., 2018; Mihaylov et al., 2018; Sun et al., 2019a). Recent advances in neural network architectures, especially the raise of the Transformer (Vaswani et al., 2017), and better contextualization of language models (Peters et al., 2018; Devlin et al., 2019; Radford et al., 2018; Grave et al., 2018; Howard and Ruder, 2018; Radford et al., 2019; Yang et al., 2019b; Dai et al., 2019) offered new opportunities to advance the field.

Here, we investigate skill transfer from a high-resource language, i.e., English, to a low-resource one, i.e., Bulgarian, for the task of multiple-choice reading comprehension. Most previous work (Pan et al., 2018; Radford et al., 2018; Tay et al., 2018; Sun et al., 2019b) was monolingual, and a relevant context for each question was available a priori. We take the task a step further by exploring the capability of a neural comprehension model in a multilingual setting using external commonsense knowledge. Our approach is based on the multilingual cased BERT (Devlin et al., 2019) fine-tuned on the RACE dataset (Lai et al., 2017), which contains over 87,000 English multiple-choice school-level science questions. For evaluation, we build a novel dataset for Bulgarian. We further experiment with pre-training the model over stratified Slavic corpora in Bulgarian, Czech, and Polish Wikipedia articles, and Russian news, as well as with various document retrieval strategies.

Finally, we address the resource scarceness in low-resource languages and the absence of question contexts in our dataset by extracting relevant passages from Wikipedia articles.

Our contributions are as follows:

- We introduce a new dataset for reading comprehension in a low-resource language such as Bulgarian. The dataset contains a total of 2,636 multiple-choice questions without contexts from matriculation exams and online quizzes. These questions cover a large variety of science topics in biology, philosophy, geography, and history.

- We study the effectiveness of zero-shot transfer from English to Bulgarian for the task of multiple-choice reading comprehension, using Multilingual and Slavic BERT (Devlin et al., 2019), fine-tuned on large corpora, such as RACE (Lai et al., 2017).

- We design a general-purpose pipeline[1] for extracting relevant contexts from an external corpus of unstructured documents using information retrieval.

The rest of this paper is organized as follows: The next section presents related work. Section 3 describes our approach. Details about the newly-proposed multiple-choice Bulgarian dataset are given in Section 4. All experiments are described in Section 5. Finally, Section 6 concludes and points to possible directions for future work.

2 Related Work

2.1 Machine Reading Comprehension

The growing interest in machine reading comprehension (MRC) has led to the release of various datasets for both extractive (Nguyen et al., 2016; Trischler et al., 2017; Joshi et al., 2017; Rajpurkar et al., 2018; Reddy et al., 2019) and non-extractive (Richardson et al., 2013; Peñas et al., 2014; Lai et al., 2017; Clark et al., 2018; Mihaylov et al., 2018; Sun et al., 2019a) comprehension. Our work primarily focuses on the non-extractive multiple-choice type, designed by educational experts, since their task is very close to our newly-proposed dataset, and are expected to be well-structured and error-free (Sun et al., 2019a).

These datasets brought a variety of models and approaches. The usage of external knowledge has been an interesting topic, e.g., Chen et al. (2017a) used Wikipedia knowledge for answering open-domain questions, Pan et al. (2018) applied entity discovery and linking as a source of prior knowledge. Sun et al. (2019b) explored different reading strategies such as back and forth reading, highlighting, and self-assessment. Ni et al. (2019) focused on finding essential terms and removing distraction words, followed by reformulation of the question, in order to find better evidence before sending a query to the MRC system. A simpler approach was presented by Clark et al. (2016), who leveraged information retrieval, corpus statistics, and simple inference over a semi-automatically constructed knowledge base for answering fourth-grade science questions.

Current state-of-the-art approaches in machine reading comprehension are grounded on transfer learning and fine-tuning of language models (Peters et al., 2018; Conneau et al., 2018; Devlin et al., 2019). Yang et al. (2019a) presented an open-domain extractive reader based on BERT (Devlin et al., 2019). Radford et al. (2018) used generative pre-training of a Transformer (Vaswani et al., 2017) as a language model, transferring it to downstream tasks such as natural language understanding, reading comprehension, etc.

Finally, there has been a Bulgarian MRC dataset (Peñas et al., 2012). It was used by Simov et al. (2012), who converted the question-answer pairs to declarative sentences, and measured their similarity to the context, transforming both to a bag of linguistic units: lemmata, POS tags, and dependency relations.

2.2 (Zero-Shot) Multilingual Models

Multilingual embeddings helped researchers to achieve new state-of-the-art results on many NLP tasks. While many pre-trained model (Grave et al., 2018; Devlin et al., 2019; Lample and Conneau, 2019) are available, the need for task-specific data in the target language still remains. Learning such models is language-independent, and representations for common words remain close in the latent vector space for a single language, albeit unrelated for different languages. A possible approach to overcome this effect is to learn an alignment function between spaces (Artetxe and Schwenk, 2018; Joty et al., 2017).

[1] The dataset and the source code are available at `http://github.com/mhardalov/bg-reason-BERT`

Moreover, zero-shot application of fine-tuned multilingual language models (Devlin et al., 2019; Lample and Conneau, 2019) on XNLI (Conneau et al., 2018), a corpus containing sentence pairs annotated with textual entailment and translated into 14 languages, has shown very close results to such by a language-specific model.

Zero-shot transfer and multilingual models had been a hot topic in (neural) machine translation (MT) in the past several years. Johnson et al. (2017) introduced a simple tweak to a standard sequence-to-sequence (Sutskever et al., 2014) model by adding a special token to the encoder's input, denoting the target language, allowing a zero-shot learning for new language pairs. Recent work in zero-resource translation outlined different strategies for learning to translate without having a parallel corpus between the two target languages. First, a many-to-one approach was adopted by Firat et al. (2016) based on building a corpus from a single language paired with many others, allowing simultaneous training of multiple models, with a shared attention layer. A many-to-many relationship between languages was later used by Aharoni et al. (2019), in an attempt to train a single Transformer (Vaswani et al., 2017) model.

Pivot-language approaches can also be used to overcome the lack of parallel corpora for the source–target language pair. Chen et al. (2017b) used a student-teacher framework to train an NMT model, using a third language as a pivot. A similar idea was applied to MRC by Asai et al. (2018), who translated each question to a pivot language, and then found the correct answer in the target language using soft-alignment attention scores.

3 Model

Our model has three components: (*i*) a context retrieval module, which tries to find good explanatory passages for each question-answer pair, from a corpus of non-English documents, as described in Section 3.1, (*ii*) a multiple-choice reading comprehension module pre-trained on English data and then applied to the target language in a zero-shot fashion, i.e., without further training or additional fine-tuning, to a target (non-English) language, as described in Section 3.2, and (*iii*) a voting mechanism, described in Section 3.3, which combines multiple passages from (*i*) and their scores from (*ii*) in order to obtain a single (most probable) answer for the target question.

3.1 Context Retriever

Most public datasets for reading comprehension (Richardson et al., 2013; Lai et al., 2017; Sun et al., 2019a; Rajpurkar et al., 2018; Reddy et al., 2019; Mihaylov et al., 2018) contain not only questions with possible answers, but also an evidence passage for each question. This limits the task to question answering over a piece of text, while an open-domain scenario is much more challenging and much more realistic. Moreover, a context in which the answer can be found is not easy to retrieve, sometimes even for a domain expert. Finally, data scarceness in low-resource languages poses further challenges for finding resources and annotators.

In order to enable search for appropriate passages for non-English questions, we created an inverted index from Wikipedia articles using Elasticsearch.[2] We used the original dumps for the entire Wikipage,[3] and we preprocessed the data leaving only plain textual content, e.g., removing links, HTML tags, tables, etc. Moreover, we split the article's body using two strategies: a sliding window and a paragraph-based approach. Each text piece with its corresponding article title was processed by applying word-based tokenization, lowercasing, stop-words removal, stemming (Nakov, 2003; Savoy, 2007), and n-gram extraction. Finally, the matching between a question and a passage was done using cosine similarity and BM25 (Robertson and Zaragoza, 2009).

3.2 BERT for Multiple-Choice RC

The recently-proposed BERT (Devlin et al., 2019) framework is applicable to a vast number of NLP tasks. A shared characteristic between all of them is the form of the input sequences: a single sentence or a pair of sentences separated by the [SEP] special token, and a classification token ([CLS]) added at the beginning of each example. In contrast, the input for multiple-choice reading comprehension questions is assembled by three sentence pieces, i.e., context passage, question, and possible answer(s). Our model follows a simple strategy of concatenating the option (candidate answer) at the end of a question. Following the notation of Devlin et al. (2019), the input sequence can be written as follows:

[CLS] Passage [SEP] Question + Option [SEP]

[2] http://www.elastic.co/

[3] http://dumps.wikimedia.org/

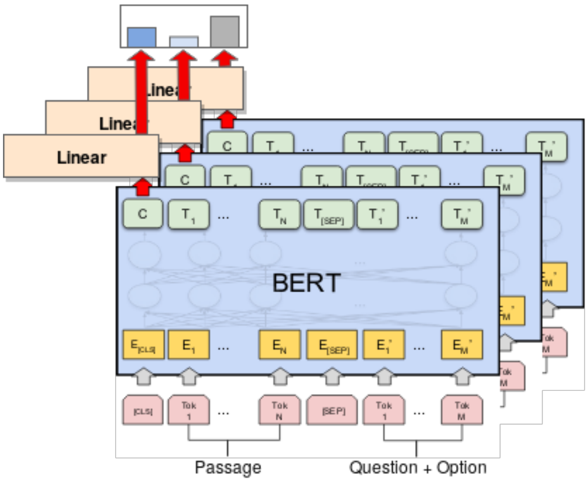


Figure 1: BERT for multiple-choice reasoning.

As recommended by Devlin et al. (2019), we introduce a new task-specific parameter vector L, $L \in \mathbb{R}^H$, where H is the hidden size of the model. In order to obtain a score for each passage-question-answer triplet, we take the dot product between L and the final hidden vector for the classification token ([CLS]), thus ending up with N unbounded numbers: one for each option. Finally, we normalize the scores by adding a softmax layer, as shown in Figure 1. During fine-tuning, we optimize the model's parameters by maximizing the log-probability of the correct answer.

3.3 Passage Selection Strategies

Finding evidence passages that contain information about the correct answer is crucial for reading comprehension systems. The context retriever may be extremely sensitive to the formulation of a question. The latter can be very general, or can contain insignificant rare words, which can bias the search. Thus, instead of using only the first-hit document, we should also evaluate lower-ranked ones. Moreover, knowing the answer candidates can enrich the search query, resulting in improved, more answer-oriented passages. This approach leaves us with a set of contexts that need to be evaluated by the MRC model in order to choose a single correct answer. Prior work suggests several different strategies: Chen et al. (2017a) used the raw predicted probability from a recurrent neural network (RNN), Yang et al. (2019a) tuned a hyper-parameter to balance between the retriever score and the reading model's output, while Pan et al. (2018) and Ni et al. (2019) concatenated the results from sentence-based retrieval into a single contextual passage.

In our experiments below, we adopt a simple summing strategy. We evaluate each result from the context retriever against the question and the possible options (see Section 3.2 for more details), thus obtaining a list of raw probabilities. We found empirically that explanatory contexts assign higher probability to the related answer, while general or uninformative passages lead to stratification of the probability distribution over the answer options. We formulate this as follows:

$$Pr(a_j|p;q) = \frac{exp(BERT(p, q + a_j))}{\sum_{j'} exp(BERT(p, q + a_{j'}))}, \quad (1)$$

where p is a passage, q is a question, A is the set of answer candidates, and $a_j \in A$.

We select the final answer as follows:

$$Ans = \arg\max_{a \in A} \sum_{p \in P} Pr(A|p;q) \quad (2)$$

4 Data

Our goal is to build a task for a low-resource language, such as Bulgarian, as close as possible to the multiple-choice reading comprehension setup for high-resource languages such as English. This will allow us to evaluate the limitations of transfer learning in a multilingual setting. One of the largest datasets for this task is RACE (Lai et al., 2017), with a total of 87,866 training questions with four answer candidates for each. Moreover, there are 25,137 contexts mapped to the questions and their correct answers.

While there exist many datasets for reading comprehension, most of them are in English, and there are a very limited number in other languages (Peñas et al., 2012, 2014). Hereby, we collect our own dataset for Bulgarian, resulting in 2,633 multiple-choice questions, without contexts, from different subjects: biology (16.6%), philosophy (23.93%), geography (23.24%), and history (36.23%). Table 2 shows an example question with candidate answers chosen to represent best each category. We use green to mark the correct answer, and bold for the question category. For convenience all the examples are translated to English.

Table 1 shows the distribution of questions per subject category, the length (in words) for both the questions and the options (candidate answers), and the vocabulary richness, measured in terms of unique words. The first part of the table presents statistics about our dataset, while the second part is a comparison to RACE (Lai et al., 2017).

Domain	#QA-pairs	#Choices	Len Question	Len Options	Vocabulary Size
12th Grade Matriculation Exam					
Biology	437	4	10.4	2.6	2,414 (12,922)
Philosophy	630	4	8.9	2.9	3,636 (20,392)
Geography	612	4	12.8	2.5	3,239 (17,668)
History	542	4	23.7	3.6	5,466 (20,456)
Online History Quizzes					
Bulgarian History	229	4	14.0	2.8	2,287 (10,620)
PzHistory	183	3	38.9	2.4	1,261 (7,518)
Overall	2,633	3.9	15.7	2.9	13,329 (56,104)
RACE Train - Mid and High School					
RACE-M	25,421	4	9.0	3.9	32,811
RACE-H	62,445	4	10.4	5.8	125,120
Overall	87,866	4	10.0	5.3	136,629

Table 1: Statistics about our Bulgaria dataset compared to the RACE dataset.

(**Biology**) The thick coat of mammals in winter is an example of:
A. physiological adaptation
B. behavioral adaptation
C. genetic adaptation
D. morphological adaptation

(**Philosophy**) According to relativism in ethics:
A. there is only one moral law that is valid for all
B. there is no absolute good and evil
C. people are evil by nature
D. there is only good, and the evil is seeming

(**Geography**) Which of the assertions about the economic specialization of the Southwest region is true?
A. The ratio between industrial and agricultural production is 15:75
B. Lakes of glacial origin in Rila and Pirin are a resource for the development of tourism
C. Agricultural specialization is related to the cultivation of grain and ethereal-oil crops
D. The rail transport is of major importance for intra-regional connections

(**History**) Point out the concept that is missed in the text of the Turnovo Constitution: „Art. 54 All born in Bulgaria, also those born elsewhere by parents Bulgarian ______, count as ______ of the Bulgarian Principality. Art. 78 Initial teaching is free and obligatory for all ______ of the Bulgarian Principality."
A. residents
B. citizents
C. electors
D. voters

(**History Quiz**) Sofroniy Vrachanski started a family that plays a big role in the history of the Bulgarian National Revival. What is its name?
A. Georgievi
B. Tapchileshtovi
C. Bogoridi
D. Palauzovi

Table 2: Example questions, one per subject, from our Bulgarian dataset. The correct answer is marked in green.

We divided the Bulgarian questions into two groups based on the question's source. The first group (*12th Grade Matriculation Exam*) was collected from twelfth grade matriculation exams created by the Ministry of Education of Bulgaria in the period 2008–2019. Each exam contains thirty multiple-choice questions with four possible answers per question. The second set of questions (*Online History Quizzes*) are history-related and are collected from online quizzes. While they are not created by educators, the questions are still challenging and well formulated. Furthermore, we manually filtered out questions with non-textual content (i.e., pictures, paintings, drawings, etc.), ordering questions (i.e., order the historical events), and questions involving calculations (i.e., how much X we need to add to Y to arrive at Z).

Table 1 shows that history questions in general contain more words (14.0–38.9 on average), compared to other subjects (8.9–12.8 on average). A tangible difference in length compared to other subjects is seen for *12th grade History* and *PzHistory*, due to the large number of quotes, and document pieces contained in questions from these two groups. Also, the average question length is 15.7, which is longer compared to the RACE dataset with 10.0. On the other hand, the option lengths per subject category in our dataset follow a narrower distribution. They fall in the interval between 2.5 and 2.9 words on average, expect for *12th grade History*, with 3.6 words. Here, we note a significant difference compared to the option lengths in RACE, which tend to be 2.4 words longer on average – 5.3 for RACE vs. 2.9 for ours.

Finally, we examine the vocabulary richness of the two datasets. The total number of unique words is shown in the last column of Table 1 (Vocab Size). For our dataset, there are two numbers per row: the first one shows statistics based on the question–answer pairs only, while the second one, enclosed in parentheses, measures the vocabulary size including the extracted passages by the Context Retriever. The latter number is a magnitude estimate rather then a concrete number, since its upper limit is the number of words in Wikipedia, and it can vary for different retrieval strategies.

5 Experiments and Evaluation

5.1 BERT Fine-Tuning

We divide the fine-tuning into two groups of models (*i*) Multilingual BERT, and (*ii*) Slavic BERT. Table 3 below presents the results in the multiple-choice comprehension task on the dev dataset from RACE (Lai et al., 2017).

#Epoch	RACE-M	RACE-H	Overall
BERT 1	64.21	53.66	56.73
BERT 2	68.80	57.58	60.84
BERT 3	69.15	58.43	61.55
Slavic 2	53.55	44.48	47.12
Slavic 3	57.38	46.88	49.94

Table 3: Accuracy measured on the dev RACE dataset after each training epoch.

Multilingual BERT As our initial model, we use BERT$_{\text{base}}$, Multilingual Cased which is pretrained on 104 languages, and has 12-layers, 768-hidden units per layer, 12-heads, and a total of 110M parameters. We further fine-tune the model on RACE (Lai et al., 2017) for 3 epochs saving a checkpoint after each epoch. We use a batch size of 8, a max sequence size of 320, and a learning rate of 1e-5.

Slavic BERT The Slavic model[4] was built using transfer learning from the Multilingual BERT model to four Slavic languages: Bulgarian, Czech, Polish, and Russian. In particular, the Multilingual BERT model was fine-tuned on a stratified dataset of Russian news and Wikipedia articles for the other languages. We use this pre-trained Slavic BERT model, and we apply the same learning procedure as for *Multilingual BERT*.

5.2 Wikipedia Retrieval and Indexing

Here, we discuss the retrieval setup (see Section 3.1 for more details). We use the Bulgarian dump of Wikipedia from 2019-04-20, with a total of 251,507 articles. We index each article title and body in plain text, which we call a *passage*. We further apply additional processing for each field:

- *ngram*: word-based 1–3 grams;
- *bg*: lowercased, stop-words removed (from Lucene), and stemmed (Savoy, 2007);
- *none*: bag-of-words index.

We ended up using a subset of four fields from all the possible analyzer-field combinations, namely *title.bg, passage, passage.bg,* and *passage.ngram*. We applied Bulgarian analysis on the *title* field only as it tends to be short and descriptive, and thus very sensitive to noise from stop-words, which is in contrast to questions that are formed mostly of stop-words, e.g., *what*, *where*, *when*, *how*.

For indexing the Wikipedia articles, we adopt two strategies: sliding window and paragraph. In the window-based strategy, we define two types of splits: small, containing 80-100 words, and large, of around 300 words. In order to obtain indexing chunks, we define a window of size K, and a stride equal to one forth of K. Hence, each $\frac{K}{4}$ characters, which is the size of the stride, are contained into four different documents. The paragraph-based strategy divides the article by splitting it using one or more successive newline characters ([\n]+) as a delimiter. We avoid indexing entire documents due to their extensive length, which can be far beyond the maximum length that BERT can take as an input, i.e., 320 word pieces (see Section 5.1 for the more details). Note that extra steps are needed in order to extract a proper passage from the text. Moreover, the amount of facts in the Wikipedia articles that are unrelated to our questions give rise to false positives since the question is short and term-unspecific.

Finally, we use a list of top-N hits for each candidate answer. Thus, we have to execute an additional query for each question + option combination, which may result in duplicated passages, thus introducing an implicit bias towards the candidates they support. In order to mitigate this effect, during the answer selection phase (see Section 3.3), we remove all duplicate entries, keeping a single instance.

[4] http://github.com/deepmipt/Slavic-BERT-NER

Setting	Accuracy
Random	24.89
Train for 3 epochs	–
+ window & title.bg & pass.ngram	29.62
+ passage.bg & passage	39.35
– title.bg	39.69
+ passage.bg^2	40.26
+ title.bg^2	40.30
+ bigger window	36.54
+ paragraph split	42.23
+ Slavic pre-training	33.27
Train for 1 epoch best	40.26
Train for 2 epochs best	41.89

Table 4: Accuracy on the Bulgarian testset: ablation study when sequentially adding/removing different model components.

5.3 Experimental Results

Here, we discuss the accuracy of each model on the original English MRC task, followed by experiments in zero-shot transfer to Bulgarian.

English Pre-training for MCRC. Table 3 presents the change in accuracy on the original English comprehension task, depending on the number of training epochs. In the table, "BERT" refers to the Multilingual BERT model, while "Slavic" stands for BERT with Slavic pre-training. We further fine-tune the models on the RACE dataset. Next, we report their performance in terms of accuracy, following the notation from (Lai et al., 2017). Note that the questions in RACE-H are more complex than those in RACE-M. The latter has more word matching questions and fewer reasoning questions. The final column in the table, *Overall*, shows the accuracy calculated over all questions in the RACE testset. We train both setups for three epochs and we report their performance after each epoch. We can see a positive correlation between the number of epochs and the model's accuracy. We further see that the Slavic BERT performs far worse on both RACE-M and RACE-H, which suggests that the change of weights of the model towards Slavic languages has led to catastrophic forgetting of the learned English syntax and semantics. Thus, it should be expected that the adaptation to Slavic languages would yield decrease in performance for English. What matters though is whether this helps when testing on Bulgarian, which we explore next.

Zero-Shot Transfer. Here, we assess the performance of our model when applied to Bulgarian multiple-choice reading comprehension. Table 4 presents an ablation study for various components. Each line denotes the type of the model, and the addition (+) or the removal (–) of a characteristic from the setup in the previous line. The first line shows the performance of a baseline model that chooses an option uniformly at random from the list of candidate answers for the target question. The following rows show the results for experiments conducted with a model trained for three epochs on RACE (Lai et al., 2017).

Our basic model uses the following setup: Wikipedia pages indexed using a small sliding window (400 characters, and stride of 100 characters), and context retrieval over two fields: Bulgarian analyzed title (*text.bg*), and word n-grams over the passage (*passage.ngram*). This setup yields 29.62% accuracy, and it improves over the random baseline by 4.73% absolute. We can think of it as a non-random baseline for further experiments. Next, we add two more fields to the IR query: passage represented as a bag of words (named *passage*), and Bulgarian analyzed (*passage.bg*), which improves the accuracy by additional 10%, arriving at 39.35%. The following experiment shows that removing the *title.bg* field does not change the overall accuracy, which makes it an insignificant field for searching. Further, we add double weight on *passage.bg*, (shown as ^2), which yields 1% absolute improvement.

From the experiments described above, we found the best combination of query fields to be *title.bulgarian*^2, *passage.ngram, passage, passage.bulgarian*^2, where the *title* has a minor contribution, and can be sacrificed for ease of computations and storage. Fixing the best query fields, allowed us to evaluate other indexing strategies, i.e., bigger window (size 1,600, stride 400) with accuracy 36.54%, and paragraph splitting, with which we achieved our highest accuracy of 42.23%. This is an improvement of almost 2.0% absolute over the small sliding window, and 5.7% over the large one.

Next, we examined the impact of the Slavic BERT. Surprisingly, it yielded 9% absolute drop in accuracy compared to the multi-lingual BERT. This suggests that the latter already has enough knowledge about Bulgarian, and thus it does not need further adaptation to Slavic languages.

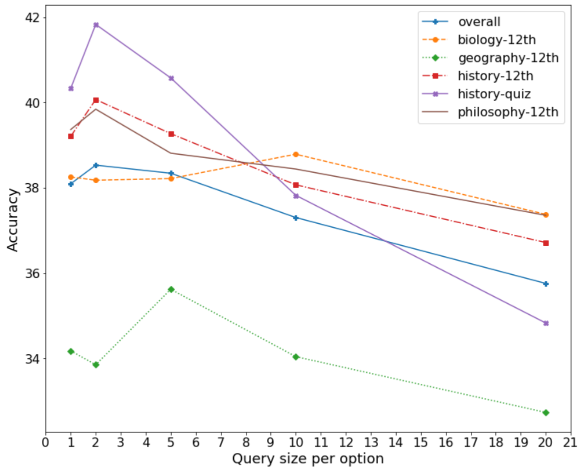


Figure 2: Accuracy per question category based on the number of query results per answer option.

Next, we study the impact of the number of fine-tuning epochs on the model's performance. We observe an increase in accuracy as the number of epochs grows, which is in line with previously reported results for English tasks. While this correlation is not as strong as for the original RACE task (see Table 3 for comparison), we still observe 1.6% and 0.34% absolute increase in accuracy for epochs 2 and 3, respectively, compared to epoch 1. Note that we do not go beyond three epochs, as previous work has suggested that 2-3 fine-tuning epochs are enough (Devlin et al., 2019), and after that, there is a risk of catastrophic forgetting of what was learned at pre-training time (note that we have already seen such forgetting with the Slavic BERT above).

We further study the impact of the size of the results list returned by the retriever on the accuracy for the different categories. Figure 2 shows the average accuracy for a given query size S_q over all performed experiments, where $S_q \in \{1, 2, 5, 10, 20\}$. We can see in Figure 2 that longer query result lists (i.e., containing more than 10 results) per answer option worsen the accuracy for all categories, except for *biology*, where we see a small peak at length 10, while still the best overall results for this category is achieved for a result list of length 5. A single well-formed maximum at length 2 is visible for *history* and *philosophy*. With these two categories being the biggest ones, the cap at the same number of queries for the overall accuracy is not a surprise. The per-category results for the experiments are discussed in more detail in Appendix A.

We can see that the highest accuracy is observed for *history*, particularly for online quizzes, which are not designed by educators and are more of a word-matching nature rather then a reasoning one (see Table 2). Finally, *geography* appears to be the hardest category with only 38.73% accuracy: 3.5% absolute difference compared to the second-worst category. The performance for this subject is also affected differently by changes in query result length: the peak is at lengths 5 and 10, while there is a drop for length 2. A further study of the model's behavior can be found in Appendix B.

6 Conclusion and Future Work

We studied the task of multiple-choice reading comprehension for low-resource languages, using a newly collected Bulgarian corpus with 2,633 questions from matriculation exams for twelfth grade in history and biology, and online exams in history without explanatory contexts. In particular, we designed an end-to-end approach, on top of a multilingual BERT model (Devlin et al., 2019), which we fine-tuned on large-scale English reading comprehension corpora, and open-domain commonsense knowledge sources (Wikipedia). Our main experiments evaluated the model when applied to Bulgarian in a zero-shot fashion. The experimental results found additional pre-training on the English RACE corpus to be very helpful, while pre-training on Slavic languages to be harmful, possibly due to catastrophic forgetting. Paragraph splitting, n-grams, stop-word removal, and stemming further helped the context retriever to find better evidence passages, and the overall model to achieve accuracy of up to 42.23%, which is well above the baselines of 24.89% and 29.62%.

In future work, we plan to make use of reading strategies (Sun et al., 2019b), linked entities (Pan et al., 2018), concatenation and reformulation of passages and questions (Simov et al., 2012; Clark et al., 2016; Ni et al., 2019), as well as re-ranking of documents (Nogueira and Cho, 2019).

Acknowledgments

We want to thank Desislava Tsvetkova and Anton Petkov for the useful discussions and for their help with some of the experiments.

This research is partially supported by Project UNITe BG05M2OP001-1.001-0004 funded by the OP "Science and Education for Smart Growth" and co-funded by the EU through the ESI Funds.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *Proceedings of the Conference of the North American Chapter of ACL*. Minneapolis, MN, USA, NAACL-HLT '19, pages 3874–3884.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464* .

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. *arXiv preprint arXiv:1809.03275* .

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to answer open-domain questions. In *Proceedings of the Meeting of the Association for Computational Linguistics*. Vancouver, Canada, ACL '17, pages 1870–1879.

Yun Chen, Yang Liu, Yong Cheng, and Victor O.K. Li. 2017b. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, ACL '17, pages 1925–1935.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457* .

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence*. Phoenix, AZ, USA, AAAI '16, pages 2580–2586.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 2475–2485.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the Meeting of the Association for Computational Linguistics*. Florence, Italy, ACL '19, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, NAACL-HLT '19, pages 4171–4186.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Austin, TX, USA, EMNLP '16, pages 268–277.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Conference on Language Resources and Evaluation*. Miyazaki, Japan, LREC '18.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Melbourne, Australia, ACL '18, pages 328–339.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics* 5:339–351.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada, ACL '17, pages 1601–1611.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proc. of the Conference on Computational Natural Language Learning*. Vancouver, Canada, CoNLL '17, pages 226–237.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, EMNLP '17, pages 785–794.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291* .

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 2381–2391.

Preslav Nakov. 2003. Building an inflectional stemmer for Bulgarian. In *Proceedings of the 4th International Conference Conference on Computer Systems and Technologies: E-Learning*. Rousse, Bulgaria, CompSysTech '03, pages 419–424.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*. Barcelona, Spain, CoCo@NIPS '16.

Jianmo Ni, Chenguang Zhu, Weizhu Chen, and Julian McAuley. 2019. Learning to attend on essential terms: An enhanced retriever-reader model for open-domain question answering. In *Proceedings of the Conference of the North American Chapter of ACL*. Minneapolis, MN, USA, NAACL-HLT '19, pages 335–344.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085* .

Xiaoman Pan, Kai Sun, Dian Yu, Heng Ji, and Dong Yu. 2018. Improving question answering with external knowledge. *arXiv preprint:1902.00993* .

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, Corina Forascu, Yassine Benajiba, and Petya Osenova. 2012. Overview of QA4MRE at CLEF 2012: Question answering for machine reading evaluation. In *CLEF Working Note Papers*. Rome, Italy, pages 1–24.

Anselmo Peñas, Christina Unger, and Axel-Cyrille Ngonga Ngomo. 2014. Overview of CLEF question answering track 2014. In *Information Access Evaluation. Multilinguality, Multimodality, and Interaction*. pages 300–306.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training .

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners .

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the Meeting of the Association for Computational Linguistics*. Melbourne, Australia, ACL '18, pages 784–789.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics* 7:249–266.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, WA, USA, EMNLP '13, pages 193–203.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.* 3(4):333–389.

Jacques Savoy. 2007. Searching strategies for the Bulgarian language. *Inform. Retrieval* 10(6):509–529.

Kiril Ivanov Simov, Petya Osenova, Georgi Georgiev, Valentin Zhikov, and Laura Tolosi. 2012. Bulgarian question answering for machine reading. In *CLEF Working Note Papers*. Rome, Italy.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019a. DREAM: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics* 7:217–231.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2019b. Improving machine reading comprehension with general reading strategies. In *Proceedings of the North American Chapter of ACL*. Minneapolis, MN, USA, NAACL-HLT '19, pages 2633–2643.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. Montreal, Canada, NIPS '14, pages 3104–3112.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-range reasoning for machine comprehension. *arXiv preprint arXiv:1803.09074* .

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. Vancouver, Canada, RepL4NLP '19, pages 191–200.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. Long Beach, CA, USA, NIPS '17, pages 5998–6008.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with BERTserini. In *Proceedings of the Conference of the North American Chapter of ACL*. Minneapolis, MN, USA, NAACL-HLT '19, pages 72–77.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019b. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* .

Appendix

A Per-Category Results

Table 5 gives an overview, including per-category breakdown, of our parameter tuning experiments. We present the results for some interesting experiments rather then for a full grid search. The first row shows a random baseline for each category. In the following rows, we compare different types of indexing: first, we show the results for a small sliding window (400-character window, and 100-character stride), followed by a big window (1,600-character window, and 400-character stride), and finally for paragraph indexing. We use the same notation as in Section 5. The last group in the table (*Paragraph*) shows the best-performing model, where we mark in bold the highest accuracy for each category. For completeness, we also show the accuracy when using the *Slavic BERT* model for prediction, which yields a 10% drop on average compared to using the *Multilingual BERT*, for each of the categories.

B Case Study

In Table 6, we present the retrieved evidence passages for the example questions in Table 2: we omit the answers, and we only show the questions and the contexts. Each example is separated by a double horizontal line, where the first row is the question starting with "*Q:*", and the following rows contain passages returned by the retriever. For each context, we normalize the raw scores from the comprehension model using Eq. 1 to obtain a probability distribution. We then select an answer using $\arg\max$, according to Eq. 2. In the table, we indicate the correctness of each predicted answer using one of the following symbols before the question:

- ✓ The question is answered correctly.
- ✗ An incorrect answer has the highest score.
- ? Two or more answers have the highest score.

We show the top retrieved result in order to illustrate the model scores over different evidence passages and the quality of the articles. The queries are formed by concatenating the question with an answer option, even though this can lead to duplicate results since some answers can be quite similar or the question's terms could dominate the similarity score.

The questions in Table 6 are from five different categories: biology, philosophy, geography, history, and online quizzes. Each of them has its own specifics and gives us an opportunity to illustrate a different model behavior.

The first question is from the biology domain, and we can see that the text is very general, and so is the retrieved context. The latter talks about *hair* rather than *coat*, and the correct answer (D) *morphological adaptation* is not present in the retrieved text. On the other hand, all the terms are only connected to it, and hence the model assigns high probability to this answer option.

For the second question, from the philosophy domain, there are two related contexts found. The first one is quite short, noisy, and it does not give much information in general. The second paragraph manages to extract the definition of *relativism* and to give good supporting evidence for the correct answer, namely that *there is no absolute good and evil* (B). As a result, this option is assigned high probability. Nevertheless, the incorrect answer *here is only one moral law that is valid for all* (A) is assigned an even higher probability and it wins the voting.

In the third example, from the domain of geography, we see a large number of possible contexts, due to the long and descriptive answers. We can make two key observations: (*i*) the query is drawn in very different directions by the answers, and (*ii*) there is no context for *Southwestern region*, and thus, in the second option, the result is for Russia, not for Bulgaria. The latter passage pushes the probability mass to an option that talks about transportation (D), which is incorrect. Fortunately, the forth context has an almost full term overlap with the correct answer (B), and thus gets very high probability assigned to it: 72%.

The fourth question, from the history domain, asks to point out a missing concept, but the query is dominated by the question, and especially by underscores, leading to a single hit, counting only symbols, without any words. As expected, the model assigned uniform probability to all classes.

The last question, a history quiz, is a factoid one, and it lacks a reasoning component, unlike the previous examples. The query returned a single direct match. The retrieved passage contains the correct answer exactly: option *Bogoridi* (C). Thereby, the comprehension model assigns to it a very high probability of 68%.

#docs	Overall	biology-12th	philosophy-12th	geography-12th	history-12th	history-quiz
Random						
0	24.89	26.09	24.44	24.18	25.87	24.03
Window Small						
title.bulgarian, passage.bulgarian						
1	39.95	40.27	40.63	34.97	42.99	41.99
2	40.22	40.27	40.63	35.95	42.62	42.72
5	40.22	38.90	40.63	38.07	41.51	42.48
10	38.66	40.50	39.84	35.46	39.30	38.83
20	36.84	37.53	39.05	33.82	38.75	34.71
title.bulgarian, passage.ngram						
1	28.94	29.06	32.06	27.29	27.49	28.40
2	29.09	29.06	33.33	25.00	28.78	29.13
5	29.05	27.46	32.06	26.63	30.63	27.67
10	29.62	29.06	32.54	26.96	30.07	29.13
20	29.43	31.81	32.70	26.63	28.60	27.18
title.bulgarian, passage.ngram, passage, passage.bulgarian						
1	38.32	38.22	40.00	34.48	39.48	40.05
2	39.08	37.07	40.32	34.48	40.59	44.17
5	39.35	40.96	39.84	34.64	41.33	41.26
10	38.63	40.50	40.63	33.50	40.41	38.83
20	36.54	38.67	37.94	31.37	37.45	38.59
passage.ngram, passage, passage.bulgarian^2						
1	39.69	40.27	40.63	35.13	42.07	41.26
2	40.26	39.82	40.95	35.95	42.62	42.96
5	39.57	39.59	39.37	37.25	40.96	41.50
10	38.70	41.19	39.52	35.78	39.30	38.35
20	37.14	39.36	37.78	35.29	38.38	34.95
title.bulgarian^2, passage.ngram, passage, passage.bulgarian^2						
1	39.84	40.27	40.79	35.13	42.25	41.75
2	40.30	40.27	40.63	36.11	42.80	42.72
5	40.26	39.13	40.63	38.40	41.14	42.48
10	38.74	40.50	39.68	35.62	39.48	39.08
20	37.07	37.76	39.05	34.64	38.56	34.95
Window Big						
title.bulgarian^2, passage.ngram, passage, passage.bulgarian^2						
1	31.22	28.38	33.97	29.41	30.81	33.25
2	33.12	31.58	37.46	31.21	33.95	29.85
5	36.04	35.70	38.10	33.82	37.82	34.22
10	36.54	37.30	36.03	33.99	39.30	36.65
20	35.62	34.55	39.68	31.05	38.38	33.74
Paragraph						
title.bulgarian^2, passage.ngram, passage, passage.bulgarian^2						
1	41.82	41.42	42.06	38.07	40.96	48.54
2	**42.23**	42.56	**43.17**	35.62	**42.99**	**49.27**
5	41.59	**43.25**	40.32	**38.73**	40.04	48.06
10	39.46	40.96	38.41	36.93	39.85	42.72
20	37.52	39.13	37.62	34.64	38.56	38.59
Slavic BERT						
1	33.19	30.89	33.17	28.76	32.29	43.45
2	33.27	31.58	31.90	31.21	35.24	37.62
5	31.14	30.21	30.16	29.25	31.00	36.65
10	30.42	29.29	29.68	29.74	31.92	31.80
20	29.66	28.60	29.37	28.43	32.10	29.85

Table 5: Evaluation results for the Bulgarian multiple-choice reading comprehension task: comparison of various indexing and query strategies.

Context	Pr_A	Pr_B	Pr_C	Pr_D
✓ Q: The thick coat of mammals in winter is an example of:				
1) The hair cover is a rare and rough bristle. In winter, soft and dense hair develops between them. Color ranges from dark brown to gray, individually and geographically diverse	0.19	0.19	0.15	0.47
✗ Q: According to relativism in ethics:				
1) Moral relativism	0.45	0.24	0.10	0.21
2) In ethics, relativism is opposed to absolutism. Whilst absolutism asserts the belief that there are universal ethical standards that are inflexible and absolute, relativism claims that ethical norms vary and differ from age to age and in different cultures and situations. It can also be called epistemological relativism - a denial of absolute standards of truth evaluation.	0.28	0.41	0.09	0.22
✓ Q: Which of the assertions about the economic specialization of the Southwest region is true?				
1) Geographic and soil-climatic conditions are blessed for the development and cultivation of oil-bearing rose and other essential oil crops.	0.12	0.52	0.28	0.08
2) Kirov has an airport of regional importance. Kirov is connected with rail transport with the cities of the Transsiberian highway (Moscow and Vladivostok).	0.14	0.27	0.06	0.53
3) Dulovo has always been and remains the center of an agricultural area, famous for its grain production. The industrial sectors that still find their way into the city's economy are primarily related to the primary processing of agricultural produce. There is also the seamless production that evolved into small businesses with relatively limited economic significance.	0.25	0.05	0.67	0.03
4) In the glacial valleys and cirques and around the lakes in the highlands of Rila and Pirin, there are marshes and narrow-range glaciers (overlaps).	0.10	0.72	0.08	0.10
? Q: Point out the concept that is missed in the text of the Turnovo Constitution: …				
1) ____________________________	0.26	0.26	0.26	0.22
✓ Q: Sofroniy Vrachanski sets up a genre that plays a big role in the history of the Bulgarian Revival. What is his name?				
1) Bogoridi is a Bulgarian Chorbadji genus from Kotel. Its founder is Bishop Sofronius Vrachanski (1739-1813). His descendants are:	0.06	0.16	0.68	0.10

Table 6: Retrieved unique top-1 contexts for the example questions in Table 2. The passages are retrieved using queries formed by concatenating a question with an answer option.

Tweaks and Tricks for Word Embedding Disruptions

Amir Hazem[1] **Nicolas Hernandez**[1]
[1] LS2N - UMR CNRS 6004, Université de Nantes, France
{Amir.Hazem,Nicolas.Hernandez}@univ-nantes.fr

Abstract

Word embeddings are established as very effective models used in several NLP applications. If they differ in their architecture and training process, they often exhibit similar properties and remain vector space models with continuously-valued dimensions describing the observed data. The complexity resides in the developed strategies for learning the values within each dimensional space. In this paper, we introduce the concept of disruption which we define as a side effect of the training process of embedding models. Disruptions are viewed as a set of embedding values that are more likely to be noise than effective descriptive features. We show that dealing with disruption phenomenon is of a great benefit to bottom-up sentence embedding representation. By contrasting several in-domain and pre-trained embedding models, we propose two simple but very effective tweaking techniques that yield strong empirical improvements on textual similarity task.

1 Introduction

Word embedding models are now a standard in many NLP applications. If the choice of the most appropriate model is not always straightforward, context word representation is at the core of each model and the performance is closely related to how well the context is exploited. In this paper, we introduce the notion of disruption, a phenomenon caused by the training process of word embedding models. Disruptions are a set of extreme embedding values that are more likely to be noise than reliable features. We consider this phenomenon as a negative side effect closely related to the data and to the training optimization decisions. If we observe the Gaussian distribution of word embedding dimensional-values, we notice a set of high positive and negative values of which the percentage varies from one embedding model to another. In the context of vector-space models where each word is represented as a point in the N-dimensional Euclidean space, disruptions may drastically affect word's position and so create unbalanced embedding values. If normalization tends to smooth this effect, our experiments reveal that detecting disruptions and adjusting them is far more efficient than a standard normalization. We show that dealing with disruptions is of a substantial benefit to bottom-up sentence embedding representation.

A bottom-up sentence representation is a weighted sum of the embedding vectors of its constituent words. This simple approach turned out to be very competitive in many NLP applications (Wieting et al., 2016; Arora et al., 2017) and outperformed several advanced RNNs and LSTM-based models of which the performance heavily depends on the quality and the large size of the training data set (Socher et al., 2011; Le and Mikolov, 2014; Kiros et al., 2015; Pagliardini et al., 2018). By contrast to sophisticated approaches, bottom-up sentence embedding models are less constrained and more easy to acquire. The core of the bottom-up model is the word embedding unit. An efficient sentence representation is then closely related to the quality of the used word embedding model. We state that the additive process of bottom-up sentence representation amplifies disruption's negative impact and propose to manage this phenomenon by introducing two tweaking techniques. Our approaches take into account and reduce the effect of disruptions in order to improve bottom-up sentence embedding representation. We evaluate bottom-up sen-

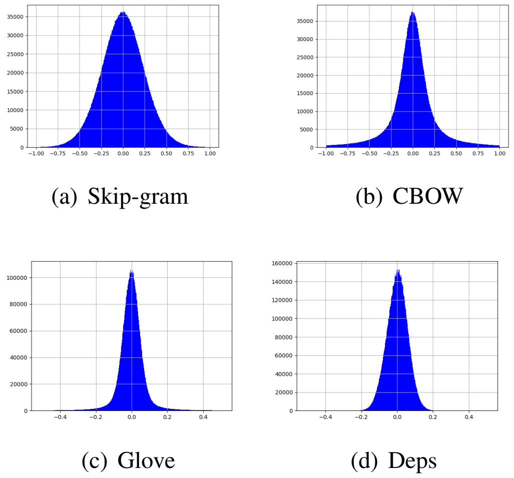
Proceedings of Recent Advances in Natural Language Processing, pages 460–464,
Varna, Bulgaria, Sep 2–4 2019.

tence embeddings using StackExchange Philosophy data set over several in-domain [1] and pre-trained embedding models on question to question similarity task and show significant improvements. The used pre-trained models are skipgram (Sg) (Mikolov et al., 2013), Glove (Pennington et al., 2014), dependency relation (Deps) (Levy and Goldberg, 2014), and the character Skipgram (ChSg) and character CBOW (ChC)[2] (Bojanowski et al., 2016).

2 Word Embedding Disruptions

Word embedding models are vector-spaces in which words are represented as points in an N-dimensional Euclidean space. Regardless of the complexity of the embedding models, the main concern remains weights estimation. At the end of the training process, the obtained model is expected to map and to efficiently represent the training data set. This supposes that each dimension has a degree of representativeness of a given word. Which means that each single dimensional value can potentially affect the word's position in the N-dimensional space. This also means that extreme values that we call disruptions might have a bigger impact on the word's position. This phenomenon is amplified by the mathematical properties and the additive process of bottom-up representation. If we consider for instance a 3-dimensional space in which one dimensional value is drastically high, the position in the 3-D space will be attracted by this dimension. This is not problematic on its own if the 3-D model well maps the data. However, if it is not the case, this might weaken the quality of the word's embedding vector.

Multiple reasons lend support to the idea that disruptions are more likely to be side effects of the training process rather than being discriminative values. The first reason comes from the characteristics of the training data set. Regardless of the size which in most cases greatly affects the quality of the embedding models, not all words are equally distributed and even using down-sampling and other sophisticated techniques to reduce the size impact, infrequent words will always be under-characterized at least for embedding models not involving character n-gram modeling. The second reason comes from the architecture and the training procedure of word embeddings (Nematzadeh et al., 2017). CBOW model for instance, predicts the target word based on a mean average weights of its context words. While, skip-gram maximizes the average log-probability of each word's context (Mikolov et al., 2013). Also, the process of weights computation is often based on batches which leads to a mean error minimization instead of a specific attention to each single training example. This optimization process might lead to some computation decisions that create margin values, potentially not relevant and so creates dimension disruptions. Even without using batches, and in order to allow efficient training, evaluating the normalization factor of the Softmax (Mikolov et al., 2013) for instance, introduces approximations. We don't claim that one of the above cited reasons is the main cause of disruptions, however we hypothesize that several parameters may lead to training side effects that lead to disruption values. If it is difficult to remove disruptions, we propose two tweaking techniques to reduce their effects.



(a) Skip-gram

(b) CBOW

(c) Glove

(d) Deps

Figure 1: 300 dimensional word embedding distributions on the StackExchange Philosophy data set.

Figure 1 illustrates the distribution values of some well-known state-of-art embedding models. We first observe that all the embedding models follow a Gaussian distribution with a mean around zero. The main differences concern the standard deviation, minimum and maximum values and the density. Also, Table 1 reports the statistics of in-domain and pre-trained embeddings. We observe that each model shows different characteris-

[1] In-domain embedding models are embeddings trained on the in-domain philosophy corpus.

[2] ChC is not available, and is only used as in-domain.

	In-domain embedding models					Pre-trained embedding models						
	Sg	CBOW	Glove	ChC	ChSG	W2V	Glove6B	Glove42B	Bow5C	Bow5W	Deps	ChSG
μ	-0.01	-0.003	-0.0008	0.001	-0.011	-0.003	-0.003	0.005	-0.0002	-0.0007	0.0004	0.007
σ	0.35	0.47	0.12	0.77	0.17	0.13	0.38	0.29 2	0.05	0.05	0.05	0.295
min	-2.07	-5.51	-3.06	-8.01	-2.55	-4.06	-3.06	-5.09	-0.31	-0.358	-0.34	-10.1
max	2.20	7.57	2.57	9.98	2.26	4.18	3.52	3.25	0.32	0.354	0.29	13.6
disrupt (%)	7.56	17.3	7.11	15.8	22.1	8.1	18.3	17.1	36.6	37.9	38.4	21.5
Cove (%)	100	100	100	100	100	52.1	55.2	66.1	52.0	52.0	51.9	56.4

Table 1: Statistics over several in-domain and pre-trained embedding models on the Philosophy data set. The mean value over the entire set of embeddings (μ), its corresponding standard deviation (σ), minimum (min) and maximum (max) values, the percentage of disruptions ($disrupt$) and vocabulary coverage of the embedding models on the Philosophy data set ($Cove$).

tics which can be comparable in some cases (Sg and ChSg) or totally different in other cases such as Sg and CBOW. Also, we see that substantial dimensional values that we define as disruptions are beyond the standard deviation.

3 Tweaks Approach

In order to reduce disruption impact on bottom-up sentence representation, we introduce the mean correction trick. A tweaking process that adjusts extreme values and center them around the mean embedding value. Let's consider N the number of embedding dimensions, V the corpus vocabulary size and S the set of disruptions with $S \in N \times V$. All the S values are replaced by the mean.

Algorithm 1 Mean Tweak approach

```
Require: $Emb = SG, CBOW, Glove, ...$
Require: $\mu \leftarrow Mean(Emb)$
Require: $\sigma \leftarrow StandardDeviation(Emb)$
Require: $\alpha \in ]min, \mu - \sigma]$
Require: $\beta \in [\mu + \sigma, max[$

function TWEAK($Emb_w$,$\mu$,$\alpha$, $\beta$)
    for $i \in Dim(Emb_w)$ do
        if $Emb_w[i] \notin [\alpha, \beta]$ then
            $Emb_w[i] \leftarrow \mu$
        end if
    end for
    return $Emb_w$
end function
```

Algorithm 1 represents the mean tweak approach where the mean value is computed over the entire set of embedding models. By contrast, per dimension approach computes one mean value per dimension. In our experiments α and β were computed empirically on a development set, however, their values are often around the $\mu + \sigma$ for max and μ - σ for min intervals. In the dimensional space this procedure tends to center the words position in the space and to reduce the impact of disruptions. From the bottom-up sentence representation side, this can be interpreted as ignoring some dimensions in the additive process. As can be seen in Table 1, the mean is often around zero and this value will not have any effect on the position of the sentence in the space when we sum-up the words of a given sentence. We consider two ways of mean computation. The first is computed over the entire set of word embeddings and the second one is the computation of a mean per dimension. We contrast both techniques in our experiments.

4 Data Description

The data set was extracted from the philosophy community question answering forum StackExchange. Basically, each post is composed of a pair of questions and one or several answers and comments. Our data set contains 5.7k posts and 1.1M tokens. We took 10% of questions for dev and 10% for test set (575 questions).

Philosophy question pair example:

Q1: were there any pre-enlightenment philosophical consideration of the naturalistic fallacy or relate concept?

Q2: naturalistic fallacy was described and named by g.e. Moore at the beginning of the 20th century. but have there been pre-enlightenment philosopher who have treat the concept?

5 Experiments and Results

Similarly to (Arora et al., 2017), we evaluate the performance of the bottom-up approach on the questions similarity task. This consists in, first, computing for each question, an average sum of its embedding words and then compute the cosine similarity on the entire set of questions to

	In-domain embedding models					Pre-trained embedding models						
Approach	Sg	CBOW	Glove	ChC	ChSG	W2V(Sg)	Glove6B	Glove42B	Bow5C(CBOW)	Bow5W(Sg)	Deps	ChSG
Baseline	63.6	40.8	46.5	36.8	49.6	49.6	51.0	50.1	53.1	52.0	50.3	54.5
+Norm	60.3	52.2	54.0	48.0	50.9	50.2	51.9	51.9	52.6	51.8	49.8	55.3
+MeanAll	**63.9**	56.1	58.7	56.4	**59.7**	54.1	55.2	57.3	**59.2**	55.8	56.4	57.8
+MeanDim	63.8	**63.1**	**61.5**	**59.0**	59.0	**55.0**	58.1	**59.1**	58.4	57.8	**56.9**	59.4
+MeanALL+Norm	60.7	53.4	54.6	55.3	**59.7**	54.3	55.8	56.7	59.0	55.7	56.8	57.9
+MeanDim+Norm	60.9	60.1	59.2	58.5	58.3	54.7	**58.4**	59.0	57.6	**58.0**	56.3	**59.5**

Table 2: Results (MAP%) of bottom-up sentence representation on the test question-to-question similarity task using the Philosophy data set, 5 in-domain 300 dimensions embedding models (CBOW and Skipgram (Mikolov et al., 2013)), Glove (Pennington et al., 2014), Character n-gram CBOW (ChC) and character Skip-gram (ChSG) (Bojanowski et al., 2016) and 7 pre-trained 300 dimensions models (W2V(sg) trained on googlenews (Mikolov et al., 2013), Glove6B trained on wikipedia and Gigaword, Glove42B trained on Common Crawl (Pennington et al., 2014), Bow5C(CBOW), Bow5W(Sg) and Deps trained on wikipedia (Levy and Goldberg, 2014) and Character skipgram (ChSG) trained on wikipedia (Bojanowski et al., 2016)).

rank the target candidates. The mean average precision (MAP) is used for evaluation. We compare the raw bottom-up approach (*baseline*) to the tweaks that we introduced. As a preprocessing step, we apply the L2 norm ($+Norm$) and our two tweaking techniques that is: $+MeanAll$ for a mean computation over the entire corpus and $+MeanDim$, for a per dimension mean adjustment. We also contrast the use of the L2 norm on the top of our tweaking techniques that we refer to as $+MeanAll + Norm$ and $+MeanDim + Norm$.

Table 2 reports the evaluation of in-domain and pre-trained embeddings of the bottom-up approach. Overall, we see that whether using in-domain or pre-trained embeddings, the baseline bottom-up approach gains significant improvements while using the proposed tweaking techniques. The gain depends on the model but the margin is in most cases very important especially for in-domain CBOW and ChC models where we notice a rise of about 20 points of Map score. We also observe improvements in all pre-trained embeddings while the gain is not as important as the one observed in the in-domain sets. This can be explained by the vocabulary coverage of pre-trained embeddings which is often between 50 and 60% as shown in Table 1. If in most cases a per dimension tweak ($MeanDim$) shows better results than $MeanAll$, we observe some margin improvements using L2 norm on the top of our tweaking techniques. If L2 norm on its own improves the performance of the baseline, the results clearly show that managing disruptions is more efficient than L2 Norm.

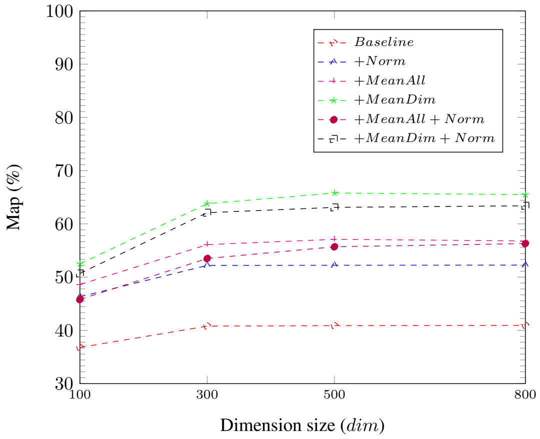


Figure 2: Contrasting several embedding dimension size of CBOW on Philosophy data set.

Figure 2 contrasts different embedding dimension size of the CBOW model[3]. The figure shows the same tendency of improvements regardless of the change in dimension size. Also, it shows that our tweaking techniques are very effective to improve question similarity using a bottom-up model.

6 Conclusion

We introduced disruption phenomenon, a negative side effect of word embedding training process. We consider the resulting set of extreme positive and negative dimensional-values as noise and as not reliable descriptive features. To reduce the effect of disruptions on bottom-up sentence representation, we proposed two tweaking techniques. Our procedure aims at adjusting the disrupted values by smoothing them around the mean value

[3] Other models are not presented for a matter of space. Except the skipgram in-domain model, all the 10 other embedding models show the same tendency as CBOW.

of embedding dimensions. Our results over in-domain and pre-trained models showed significant improvements for question similarity task.

7 Acknowledgments

The current work was supported by the ANR 2016 PASTEL (ANR-16-CE33-0007) project[4].

References

Sanjeev Arora, Liang Yingyu, and Ma Tengyu. 2017. A simple but tough to beat baseline for sentence embeddings. In *Proceedings of the 17th International Conference on Learning Representations (ICLR'17)*. pages 1–11.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726* .

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR* abs/1405.4053.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. pages 302–308. http://aclweb.org/anthology/P/P14/P14-2050.pdf.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Aida Nematzadeh, Stephan C. Meylan, and Thomas L. Griffiths. 2017. Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In *CogSci*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *Internationa Conference on Learning Representations, CoRR* abs/1511.08198.

[4] `http://www.agence-nationale-recherche.fr/?Projet=ANR-16-CE33-0007`.

Meta-Embedding Sentence Representation for Textual Similarity

Amir Hazem[1] **Nicolas Hernandez**[1]
[1] LS2N - UMR CNRS 6004, Université de Nantes, France
{Amir.Hazem,Nicolas.Hernandez}@univ-nantes.fr

Abstract

Word embedding models are now widely used in most NLP applications. Despite their effectiveness, there is no clear evidence about the choice of the most appropriate model. It often depends on the nature of the task and on the quality and size of the used data sets. This remains true for bottom-up sentence embedding models. However, no straightforward investigation has been conducted so far. In this paper, we propose a systematic study of the impact of the main word embedding models on sentence representation. By contrasting in-domain and pre-trained embedding models, we show under which conditions they can be jointly used for bottom-up sentence embeddings. Finally, we propose the first bottom-up meta-embedding representation at the sentence level for textual similarity. Significant improvements are observed in several tasks including question-to-question similarity, paraphrasing and next utterance ranking.

1 Introduction

According to Enkvist (1987): *"a model is a simplified representation of reality. It is simplified because it aims at reproducing a selection of relevant elements of reality rather than all of reality at once."*. If several word embedding models (Mikolov et al., 2013a; Pennington et al., 2014; Yin and Schütze, 2016; Arora et al., 2017) capture a selection of relevant features, different embedding sets can cover different characteristics which can also be complementary (Yin and Schütze, 2016). In order to capture a wide range of features, it is useful to perform models combination (ensemble models). The representation of longer pieces of texts such as sentences, by an element-wise sum of their word embeddings has recently shown promising results and outperformed sophisticated models in several textual similarity tasks (Mikolov et al., 2013a; Arora et al., 2017). This representation, also known as bottom-up sentence embeddings, is greatly affected by the choice of word embedding models. In this paper, we propose a systematic study of the impact of word embedding models on bottom-up sentence representation for textual similarity. We report the results of the main individual pre-trained embedding models that are publicly available as well as embedding models trained on in-domain data sets. Finally, we contrast multiple ensemble models and propose the first bottom-up meta-embedding sentence representation for textual similarity. We evaluate the different approaches on four tasks that is: question-to-question similarity (SemEval 2016/2017), textual entailment (SemEval 2014), paraphrasing (Sick) and next utterance ranking (NUR) and show under which conditions meta-embeddings can be beneficial to bottom-up sentence-based approaches.

2 Related Work

Embedding models at the word level representations have been widely explored in many applications (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2016). Naturally, they have been extended to sentence, paragraph and document level representations (Socher et al., 2011; Mikolov et al., 2013a; Le and Mikolov, 2014; Kalchbrenner et al., 2014; Kiros et al., 2015b; Wieting et al., 2016; Arora et al., 2017), thanks to the continuous advances of deep neural embedding methods such as Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM) and Convolutional Neural Networks (CNN). Some sentence embedding representations can be seen as a direct inspiration from word embedding models. For instance, while the skip-gram model (Mikolov et al., 2013a) predicts the surrounding words given a source word, in the same way, *SkipThought* (Kiros et al., 2015a) and *FastSent* (Hill et al., 2016) models predict surrounding sentences given a source sentence. Also, the paragraph *DBOW* model (Le and Mikolov, 2014) learns representations for variable length pieces of texts and learns to predict the surrounding words based on contexts sampled from paragraphs. Recently, Pagliardini et al. (2018) introduced *Sent2Vec*, an approach based on word vectors along with n-gram embeddings simultaneously to represent sentences.

Another type of sentence embedding representation, also called bottom-up approach, represents sentences by a weighted sum of the embedding vectors of their individual words. This naive approach turned out to be competitive and outperformed sophisticated approaches based on RNNs and LSTMs in many natural language processing applications (Mikolov et al., 2013a; Wieting et al., 2016; Arora et al., 2017; Hazem and Morin, 2017). Mikolov et al. (2013a) for instance demonstrated the effectiveness of their model on the phrase analogy task. They used the hierarchical softmax and subsampling using large amount of data. Wieting et al. (2016) have shown that a simple but supervised word averaging model of sentence embeddings leads to better performance on the paraphrase pairs data set (PPDB). However, the performance of their approach is closely related to the supervision from the date set, while without supervision, their approach did not perform well on textual similarity tasks. More recently, Arora et al. (2017) proposed a new sentence embedding method where they first compute a weighted average sum of the word embedding vectors of sentences, and then, remove the projections of the average vectors on their first principal components. Like Mikolov et al. (2013a) and Wieting et al. (2016), their approach is based on word embedding sum, but the difference is remarkable on the weighted schema and on the use of principal component analysis (PCA) method to remove the correlation of sentence vectors dimensions. They significantly achieved better performance than the unweighted average on a variety of textual similarity tasks. A noticeable remark is that their approach outperformed sophisticated supervised methods such as RNN's and LSTM's. Finally, some approaches are supervised and need labelled data, such as *DictRep* (Hill et al., 2015) which uses structured data to map dictionary definitions of words with their pre-trained embeddings. With the encouraging results and simplicity of bottom-up approaches, we focus in this paper on this type of approaches and show their potential while used jointly with meta-embeddings.

3 Sentence Meta-Embedding Representation

To deal with textual similarity, we propose a new approach that we refer to as meta-embedding sentence representation (MetaSentEmb). In the next sections we first recall the principle of ensemble approach from which we drawn our inspiration, then we give the details of our approach.

3.1 Ensemble Approach

The principle of the ensemble approach is to combine different models in order to catch the strength of each individual model. The main combination techniques that have shown their effectiveness are: vector addition (Garten et al., 2015) and vector concatenation (Garten et al., 2015; Yin and Schütze, 2016). For vector addition, given two embedding models, the procedure consists of applying a simple dimension-wise vector addition[1]. For vector concatenation, given two embedding models of dimensions $dim1$ and $dim2$, the resulting concatenated embedding vector will be of size $dim1 + dim2$. The vectors have to be normalized before concatenation. Usually L2 norm is performed[2]. Yin and Schütze (2016) performed a weighted concatenation of 5 embedding models. They also experienced the SVD on top of weighted concatenation vectors of dimension 950. This resulted in a reduced model of 200 dimensions.

3.2 Proposed Approach

The bottom-up sentence embedding representation consists of representing each given sentence (or piece of text of any length) by an embedding vector which is the sum of the vector embedding of each word of the sentence (Mikolov et al.,

[1] This technique can not be applied when embeddings are not of the same dimension size.

[2] L2 norm can be performed either at dimension level (as suggested by Glove authors) or at vector length level.

2013b; Wieting et al., 2016; Arora et al., 2017). This representation is illustrated in the following equation:

$$Sent_i = \sum_{j=1}^{n}(Embedding(w_j)) \qquad (1)$$

with $Sent_i$ a given sentence i and n the number of words in $Sent_i$. $Embedding(w_j)$ corresponds to the embedding model used to represent each word of the sentence $Sent_i$. We refer to this baseline approach as $SentEmb$ for sentence embedding representation. A variant of this representation is the use of a weighted sum as presented in (Wieting et al., 2016) for instance.

In this work, we extend the baseline representation ($SentEmb$) and propose $MetaSentEmb$, a meta-embedding sentence representation. We aim at improving sentence representation based on the sum of its word embeddings. As we mainly operate at the word level representation, we study different word meta-embedding techniques for sentence representation. We basically use an ensemble approach to represent each word, which means that each word has its own meta-embedding. Then, we sum each meta-embedding word of a given sentence to obtain a meta-embedding sentence representation (equation 2).

$$Sent_i = \sum_{j=1}^{n}(Ensemble(w_j)) \qquad (2)$$

with $Sent_i$ a given sentence i and n the number of words in $Sent_i$. $Ensemble(w_j)$ corresponds to the ensemble technique used to represent word meta-embeddings. $Ensemble(w_j)$ can be the additive or the concatenation technique. We refer to our proposed approach as $MetaSentEmb$ for sentence meta-embedding representation. Each sentence is pre-processed (Tokenization, part-of-speech tagging and lemmatization). Depending on the targeted task, stop-words can be removed and part of speech filtering can be applied (keeping only nouns, verbs and adjectives for instance).

4 Data and Tasks Description

In this section, we briefly outline the different textual resources used for our experiments, namely: (i) the Qatar Living corpus used in SemEval 2016/2017 for question similarity task, (ii) the Sick corpus used in SemEval 2014 for textual entailment and relatedness, (iii) the Microsoft Research Paraphrase Corpus (MSRPC) used for paraphrase detection and (iv) the Ubuntu Dialogue Corpus used for Next Utterance Ranking (NUR).

4.1 Embedding Models

To study the impact of external data and context representation, we chose different embedding models. In addition to the word2vec model trained on Google News (Mikolov et al., 2013a), we used the two Glove models respectively trained on Wikipedia+GigaWord ($Glove6B$) and on Common Crawl ($Glove42B$) (Pennington et al., 2014). We also used three Wikipedia pre-trained models (Levy and Goldberg, 2014), that is, two linear bag of word contexts and one dependency-based context. The bag of word models use a context size of 5 ($Bow5_C$ corresponds to CBow and $Bow5_W$ to skipgram). The dependency-based model used syntactic relations ($Deps$). Finally, we experienced the recent proposed character n-gram model (Bojanowski et al., 2016) by using the character Skip-gram model trained on Wikipedia ($ChSG$). A summary of the pre-trained out-of-domain embedding sets is presented in Table 1. We also trained embedding models (CBOW, Skipgram, Glove and character n-gram models) on in-domain data sets (Qatar Living and Sick corpus of SemEval, MSPR for paraphrasing and Ubuntu for NUR). We respectively noted in domain trained embeddings as $CBow$, $SkipGram$, $Glove$, $CharSG$ and $CharCBOW$.

4.2 Data Sets

4.2.1 Qatar Living Corpus

The Qatar Living corpus is a community question answering data set made of original and related questions and their n corresponding answers. The training and development data sets consist of 317 original questions and 3,169 related questions[3]. The test sets of 2016 and 2017 respectively consist of 70 original/700 related questions and 88 original/880 related questions. The SemEval (2016/2017) question-to-question similarity shared task (Task3, SubtaskB) consists of identifying for each original question, its corresponding related questions over 10 candidates (Nakov et al., 2016, 2017). The question-to-question similarity task of SemEval offers an appropriate and interesting framework for evaluating our meta-

[3] http://alt.qcri.org/semeval2016/task3/index.php?id=data-and-tools

Model	Vocab	Dim	Training Data
word2vec	93k	300	Google News (Mikolov et al., 2013a)
Glove6B	400k	50-300	Wikipedia-Gigaword (Pennington et al., 2014)
Glove42B	1.9M	300	Common Crawl (Pennington et al., 2014)
Bow, Deps	175K	300	Wikipedia (Levy and Goldberg, 2014)
CharSG	175K	300	Wikipedia (Bojanowski et al., 2016)

Table 1: Pre-trained embedding sets (Dim: dimension size).

embedding approach since an evaluation of multiple approaches including sentence embeddings have been already performed.

4.2.2 Sick Corpus

The sick data set consists of 10,000 English sentence pairs annotated for relatedness in meaning (a score form 1 to 5) and for entailment (Neutral, Entailment or Contradiction). The SemEval 2014 shared task (Task1) consists of predicting whether two given sentences are entailed, contradictions or neutral. Using sentence embeddings as well as meta-embeddings for entailment prediction is an appropriate textual similarity task for evaluation, however, dealing with contradictions and neutral sentences is more difficult than a binary classification which consists of predicting whether sentences are entailed or not. In any case and for the sake of comparison, we perform the same evaluation as the state of the art approaches by keeping the three classes (Neutral, Entailment or Contradiction) instead of two classes (Entailment or Not Entailment). As this work is mainly dedicated to the evaluation of sentence representations in sentence similarity, we only focus on the entailment part and don't consider the relatedness (we only report the results of the accuracy).

4.2.3 Microsoft Research Paraphrase Corpus

The Microsoft Research Paraphrase (MSRP) Corpus (Dolan et al., 2004) is composed of 5,801 news paraphrase sentence pairs extracted from the web. Each sentence pair has been annotated by humans as being in paraphrase relationship (label=1) or not (label=0). 67% of sentence pairs are positive examples (in paraphrase relationship) and 33% are negative examples which make the corpus unbalanced. The corpus has been divided into 4,076 training pairs and 1,725 test pairs. The paraphrasing task consists of identifying if a paraphrase relation exists between two given sentences. By contrast to the question similarity and entailment prediction tasks, sentence embedding similarity for paraphrasing might not be appropriate for evaluation since the MSRP corpus includes many sentence pairs which are not paraphrases but contain many similar words. Sentence similarity based approaches should fail in this case to detect paraphrases. However, showing the behaviour and the performance of sentence similarity approaches including meta-embeddings on such a task may offer some clues and may constitute a baseline for more sophisticated approaches.

4.2.4 Ubuntu Dialogue Corpus

The Ubuntu Dialogue Corpus is a large freely available multi-turn dialogue data set (Lowe et al., 2015) constructed from the Ubuntu chat logs[4]. The corpus (Human-Human chat) consists of approximately 930,000 two person dialogues, 7,100,000 utterances[5] and 100,000,000 words. The task of NUR consists of retrieving the most probable utterance among a database of existing human productions given a similar context. This task offers a key challenge for sentence similarity approaches since the relations between dialogue utterances are more generic. Here also, evaluating sentence similarity based approaches on a different task, should give some insights about their behaviour and to what extent it might help utterance prediction.

[4] The first version can be found in `http://irclogs.ubuntu.com/`. A newer version has been recently released in `https://github.com/rkadlec/ubuntu-ranking-dataset-creator`

[5] All the replies and initial questions are referred to as utterances

Input		Tasks					
Training data	Models	SemEval16	SemEval17	MSRP	SICK	NUR	
		Map (%)	Map (%)	Accuracy	Accuracy	1 in 10 R@1	
1*Wiki/GigaWord	Glove6B	74.63 (8)	43.12 (5)	69.7 (5)	**64.3** (2)	63.6 (4)	(4)
1*Common Crawl	Glove42B	74.93 (7)	41.68 (9)	66.9 (8)	61.5 (8)	59.4 (11)	(10)
1*Google News	word2vec	74.42 (9)	42.38 (8)	67.5 (7)	63.5 (5)	62.0 (7)	(8)
4*wikipidia	Bow5C	74.08 (10)	42.93 (6)	66.4 (9)	47.1 (12)	58.8 (12)	(11)
	Bow5W	**75.64** (2)	**45.69** (2)	70.1 (4)	**63.9** (3)	62.6 (5)	(3)
	Deps	75.02 (6)	44.33 (4)	67.9 (6)	63.1 (7)	60.2 (8)	(6)
	CharSG	75.08 (5)	42.91 (7)	**71.8** (2)	**64.4** (1)	60.1 (9)	(7)
5* In-domain	Glove	73.04 (11)	41.57 (10)	64.9 (11)	54.4 (11)	62.3 (6)	(12)
	Cbow	72.78 (12)	40.12 (11)	65.1 (10)	60.1 (10)	**66.1** (2)	(12)
	SkipGram	**76.16** (1)	**45.58** (3)	**70.3** (3)	**63.9** (3)	**68.5** (1)	(1)
	CharCBOW	75.13 (4)	45.23 (4)	63.4 (12)	61.1 (9)	59.8 (10)	(9)
	CharSG	**75.21** (3)	**46.75** (1)	**72.1** (1)	63.3 (6)	**64.2** (3)	(2)

Table 2: Results of SentEmb for five distinct textual relation detection tasks (question-to-question with SemEval16 and SemEval17, paraphrase with MSRP, entailment with SICK and Next Utterance Ranking with UDC) using different pre-trained out-of-domain and in-domain embedding models. The numbers in brackets refer to the model rank in the given task, except for the last column which ranks the models regardless of the task. The score of the three best models for each task are in bold.

5 Results and Discussion

We conducted two sets of experiments. The first one aims at providing insights about the behaviour of pre-trained and in-domain embeddings used individually. The second one aims at studying the contribution of ensemble models.

Table 2 shows that, regardless of the task, the skipgram models (in-domain SkipGram, in-domain CharSG, out-of-domain Wikipedia Bow5W) outperform the other models. In addition, the two best models are in-domain and the two following are out-of-domain. The fourth position is hold by the Wikipedia/GigaWord Glove6B model. Among the worst models, we observe two CBOW models (in-domain Cbow out-of-domain Bow5C) as well as the in-domain Glove and the out-of-domain Glove42B models. Having said that, even if the differences between the extremities are notable, the coefficients of variability between two successive ranked scores are often very low. A closer look at the results shows that the out-of-domain and in-domain character skipgram models (CharSG) performed best for the paraphrase prediction task (MSRP). The entailment detection task (SICK) is the only one for which best models are largely out-of-domain. This can be explained by the very small size of the in-domain training data set. Surprisingly, the in-domain CBow which is globally one of the two worst models achieves the second best position for the next utterance ranking task (NUR). This can be explained by the large size of the in-domain Ubuntu data set while compared to other in-domain data sets.

Table 3 reports the results of MetaSentEmb approach for the four tasks using several pre-trained embedding combinations. Overall, we observe that pre-trained embedding combination is useful in the majority of tasks (except the SICK task where no significant improvements were observed). That said, not all the combinations are efficient. It depends on the tasks and on the nature of the training data sets. For instance, in the question-to-question similarity task (Semeval) the best meta-embedding models combination were Glove6B with Glove42B (76.2% using addition and 76.4% using concatenation on 2016 edition) and CharSG with Glove42B (76.5% using addition and 76.2% using concatenation on 2016 edition), while for 2017 edition the best models were Glove6B with word2vec (47.3% using concatenation) and Deps combined with Glove42B (47.4% using addition). It is to note that Deps concatenated to Bow5W obtained similar results with

Tasks	Pre-trained embedding models																	
	Glove6B						ChSG					w2v				Deps		
	Gl42	w2c	B5C	B5W	Deps	ChSG	Gl42	w2v	B5C	B5W	Deps	Gl42	B5C	B5W	Deps	Gl42	B5C	B5W
$SE16_A$	**76.2**	75.2	75.5	75.6	74.8	74.6	**76.5**	73.9	74.8	75.8	73.57	75.0	74.1	75.4	74.8	75.4	74.2	75.1
$SE16_C$	**76.4**	76.2	75.6	76.1	74.4	75.0	76.2	74.8	74.7	75.8	74.7	75.6	74.6	76.1	74.2	75.3	74.8	75.3
$SE17_A$	44.1	44.5	43.4	46.4	45.3	45.0	43.8	42.4	44.1	45.8	45.4	44.6	43.8	46.5	46.1	**47.41**	46.1	46.5
$SE17_C$	45.9	**47.3**	45.2	46.1	45.8	45.3	45.0	43.7	44.4	46.3	46.1	45.3	43.4	46.1	45.7	45.8	45.4	**47.1**
$MSPR_A$	69.2	68.4	68.6	68.1	70.0	68.1	**72.6**	68.9	68.0	69.1	70.3	70.7	69.2	67.8	71.5	70.6	69.1	70.8
$MSPR_C$	69.4	67.9	68.6	67.0	70.4	68.0	**72.7**	69.2	69.1	68.0	71.5	71.5	69.2	68.0	71.0	70.3	69.2	70.3
$SICK_A$	64.8	64.4	64.2	64.2	64.3	64.4	64.2	63.6	62.2	63.7	63.9	63.3	62.7	63.4	63.8	64.1	63.4	63.9
$SICK_C$	64.3	64.3	64.0	63.9	64.3	64.4	64.3	63.5	62.4	63.6	64.1	63.5	63.1	63.5	63.9	64.1	63.5	63.9
NUR_A	**65.1**	65.0	61.3	63.2	62.0	64.9	62.9	61.6	57.8	60.0	58.4	64.2	59.8	61.5	60.2	60.2	60.9	62.1
NUR_C	**65.6**	65.3	61.1	63.6	62.1	65.1	64.4	61.5	57.8	60.2	58.5	64.4	59.6	62.0	60.3	62.2	61.1	62.2

Table 3: Results of the meta-embedding SentEmb, using addition ($_A$) and concatenation ($_C$) along with pre-trained embeddings. $SE16$ and $SE17$ stand respectively for SemEval16 and SemEval17. Models names were also digested to match the page setup: Glove42B (Gl42), word2vec (w2v), CharSG (ChSG), Bow5C (B5C) and Bow5W (B5W).

Tasks	In domain embedding models									
	SkipGram				CharSG			Cbow		Glove
	Cbow	Glove	CharCBOW	CharSG	Cbow	Glove	CharCBOW	Glove	CharCBOW	CharCBOW
$SE16_A$	73.5	74.2	**75.4**	**75.5**	72.7	74.0	75.3	74.5	74.7	73.6
$SE16_C$	71.4	74.1	71.4	73.5	73.7	71.0	75.1	73.5	74.2	71.0
$SE17_A$	40.9	41.6	45.1	45.4	41.3	40.4	**46.0**	41.9	41.6	40.2
$SE17_C$	42.9	43.4	44.1	43.1	42.4	41.2	44.0	42.6	40.6	42.1
$MSPR_A$	64.7	62.2	67.5	**70.7**	63.6	62.7	68.1	61.1	63.4	62.4
$MSPR_C$	63.5	62.4	66.3	69.4	61.2	63.4	67.7	60.0	64.6	61.2
$SICK_A$	62.2	62.2	62.8	**63.9**	61.7	61.3	62.3	60.0	61.4	61.4
$SICK_C$	62.3	63.4	61.3	63.1	61.2	63.1	62.1	61.1	61.6	60.4
NUR_A	**66.5**	63.9	62.5	65.6	66.2	63.4	64.7	64.8	65.8	63.9
NUR_C	**66.5**	64.2	63.1	66.1	66.4	64.4	62.7	**66.7**	65.9	63.7

Table 4: Meta-Embedding results using addition ($_A$) and concatenation ($_C$) along with in-domain embedding models. $SE16$ and $SE17$ stand respectively for SemEval16 and SemEval17.

47.1%.

For the paraphrasing task (MSPR), the best meta-embedding model was CharSG with Glove42B (72.6% using addition and 72.7% using concatenation). Other interesting combinations can be observed such as CharSG with Deps (71.5% using concatenation) and Deps with word2vec (71.5% using addition), etc. Concerning the NUR task, the best meta-embedding model is Glove6B combined with Glove42B (65.1% using addition and 65.6% using concatenation) closely followed by Glove6B combined with word2vec (65.0% using addition and 65.3% using concatenation) and CharSG (64.9% using addition and 65.1% using concatenation). Surprisingly, the majority of other models failed to improve the performance of SentEmb. One particular remark is that the best meta-embeddings always involve the Glove models. Finally, no significant improvements were observed for the entailment task (SICK). This may be due to the task itself which consists of recognizing not only the entailment relation but also the opposite and the neutral relations. In this study, no particular attention was given to opposite and neutral labels. If the combination of different embedding models is useful for 3 out of 4 tasks, the nature of the data sets also plays an important role. Embedding models trained on different data sets may provide complementary information. This can be observed for instance when combining Glove6B (trained on Wikipedia and Gigaword) and Glove42B (trained on Common Crawl). An important observation regarding Tables 2 and 3 is that best individual models are not necessary the most appropriate for combination. For instance,

Data	Tasks				
Models	SemEval16 Map (%)	SemEval17 Map (%)	MSRP Accuracy	SICK Accuracy	NUR 1 in 10 R@1
Best@1	**76.7** (1)	47.2 (3)	**80.4**	**84.6**	55.2
Best@2	76.0	46.9	77.4	83.6	48.8
Best@3	75.8	46.6	76.8	83.1	37.9
SentEmb	76.16	46.75	72.2	64.4	**68.5**
MetaSentEmbADD (In)	75.5	46.0	70.7	63.9	66.5
MetaSentEmbConcat (In)	74.1	44.0	69.4	63.1	66.7
MetaSentEmbADD (Out)	76.5 (2)	**47.4** (1)	72.6	64.8	65.1
MetaSentEmbConcat (Out)	76.4 (3)	47.3 (2)	72.7	64.3	65.6

Table 5: Results obtained by the 3 best state of the art models proposed during the official competitions of each task. Results obtained by the SentEmb baseline and with our proposed MetaSentEmb using addition and concatenation techniques over pre-trained and in-domain embeddings as well as their combinations.

Bow5W (rank 3 overall the four tasks) was less efficient while combined to other models, on the contrary, Glove42b which performed poorly individually (rank 10 overall the four tasks), turned out to be very efficient while combined to other models.

To study the impact of in-domain embeddings, we report in Table 4 the results of MetaSentEmb while using embeddings trained on the in-domain data set of each task. According to the results, we observe the same tendency as for pre-trained embeddings. However, the improvements seem to be task dependent. For instance, the best obtained results were 76.5% for pre-trained versus 75.5% (Semeval 2016) and 47.4% for pre-trained versus 46.0% (Semeval 2017) while for NUR task, the in-domain embedding obtained better results with 66.5% versus 65.6% for the pre-trained models. That said for the NUR results, the different is not significant. Generally speaking, the results of Tables 3 and 4 confirm the usefulness of using external data in addition to various embedding models and also put forward the possibility to combine embeddings trained on both in-domain and external data sets.

Table 5 reports the 3 best state of the art results obtained during the official competition of each task. Also, it contrasts the SentEmb baseline with our proposed MetaSentEmb using addition and concatenation techniques over pre-trained (Out) and in-domain (In) embeddings as well as their combinations. Below the header, the first horizontal frame reports the state of the art results. The second frame depicts results for similarity measures and the last frame contains results of classification-based approaches.

Globally, except for the NUR task, the meta-embedding configurations using pre-trained models are slightly better than the ones using in-domain models and enhance the performance of the SentEmb model. In particular, they outperform the SentEmb baseline and are ranked among the three best models for the SemEval tasks. Concerning the NUR task, the SentEmb baseline and all the meta-embedding models outperform the three best state of the art models. For this specific task, the combination of in-domain models give better results than out-of-domain models. Concerning the MSRP and the SICK tasks, while meta-embedding models build on out-of-domain corpora achieve better results than in-domain models, none of them succeeded in beating the state of the art models.

If additional efforts are certainly needed to understand the weak results on the entailment task and the different errors over all the evaluations, our observations through an error analysis showed different findings, depending on the task of course but also on the proposed method itself which is quite naive, especially for tasks like paraphrasing or entailment. First, MetaSentEmb performed well on the question-to-question similarity task and was competitive with regards to the best SemEval systems. This is certainly due to the adequacy of the task with our way of measuring sentence similarity. The questions in the Qatar Living corpus contain few ambiguities and the main errors were due to the specific forum vocabulary and mistakes that can be done by users. Also, one notable remark is the size of the original and related questions which is very important. Our way to deal with that was to filter stop-words and keep only nouns, verbs and adjectives to limit the impact of long sentences. Using POS-tagging also provided tagging errors that introduced some errors of our system. Second, MetaSentEmb did not compete with the three best systems on the paraphrasing task (MSPR), however, if we compare the results of MetaSentEmb with state of art sentence embedding representations such as FastSent (72.2%) and Skipthough (73.0%) (Hill et al., 2016), our approach obtained similar results (72.7%) with much simpler training. This finding is encouraging while no particular attention was given to the characteristics of paraphrase. Concerning textual entailment, MetaSentEmb failed to improve the performance of SentEmb. Here also the particularities of the small data set as well as the prediction of three classes including contradiction and neutral sentences may explain the low results. Finally, for the NUR task, our approach turned out to be very efficient. Utterance characteristics, at least for the Ubuntu corpus exhibit strong similarities which are certainly better captured by our meta-embedding approach.

6 Conclusion

In this paper we introduced the first bottom-up meta-embedding sentence representation for textual similarity. We have explored a variety of pre-trained and in-domain embedding models and there impact on question-to-question similarity, paraphrasing, textual entailment and next utterance ranking tasks. We have also proposed meta-embedding sentence representations based on vectors addition and concatenation and have shown under which conditions they can be jointly used for better performance. If further investigations are needed, the preliminary results lend support the idea that using meta-embeddings improve the performance of bottom-up sentence-based embedding approaches and offer an appropriate way to deal with textual similarity. One notable advantage of our approach is its simplicity, especially when using pre-trained embeddings since no computational cost is incurred.

7 Acknowledgments

The current work was supported by the ANR 2016 PASTEL (ANR-16-CE33-0007) project[6].

References

Sanjeev Arora, Liang Yingyu, and Ma Tengyu. 2017. A simple but tough to beat baseline for sentence embeddings. In *Proceedings of the 17th International Conference on Learning Representations (ICLR'17)*. pages 1–11.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JOURNAL OF MACHINE LEARNING RESEARCH* 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR* abs/1607.04606. http://arxiv.org/abs/1607.04606.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning, ICML*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '04.

N Enkvist. 1987. *Text linguistics for the applier: An orientation.*. In U. Connor R. Kaplan, Writing across languages: Analysis of L2 Text (pp. 23-44), Reading, MA: Addison-Wesley.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining distributed vector representations for words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, CO, USA, pages 95–101. http://www.aclweb.org/anthology/W15-1513.

Amir Hazem and Emmanuel Morin. 2017. Bilingual word embeddings for bilingual terminology extraction from specialized comparable corpora. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, pages 685–693. http://aclweb.org/anthology/I17-1069.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *CoRR* abs/1602.03483. http://arxiv.org/abs/1602.03483.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *CoRR* abs/1504.00548. http://arxiv.org/abs/1504.00548.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *CoRR* abs/1404.2188.

[6] http://www.agence-nationale-recherche.fr/?Projet=ANR-16-CE33-0007.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015a. Skip-thought vectors. *arXiv preprint arXiv:1506.06726* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015b. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 3294–3302.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR* abs/1405.4053.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. pages 302–308. http://aclweb.org/anthology/P/P14/P14-2050.pdf.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Tomas Mikolov, Scott Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, San Diego, California, SemEval '16.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *Internationa Conference on Learning Representations, CoRR* abs/1511.08198.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *ACL (1)*. The Association for Computer Linguistics. http://dblp.uni-trier.de/db/conf/acl/acl2016-1.htmlYinS16.

Emoji Powered Capsule Network to Detect Type and Target of Offensive Posts in Social Media

Hansi Hettiarachchi[1], **Tharindu Ranasinghe**[2]
[1]Department of Computer Science and Engineering,, University of Moratuwa, Sri Lanka
[2]Research Group in Computational Linguistics, University of Wolverhampton, UK
hansi.11@cse.mrt.ac.lk
t.d.ranasinghehettiarachchige@wlv.ac.uk

Abstract

This paper describes a novel research approach to detect type and target of offensive posts in social media using a capsule network. The input to the network was character embeddings combined with emoji embeddings. The approach was evaluated on all the subtasks in SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. The evaluation also showed that even though the capsule networks have not been used commonly in NLP tasks, they can outperform existing state of the art solutions for offensive language detection in social media.

1 Introduction

Social media has become a normal medium of communication for people these days as it provides the convenience of sending messages fast from a variety of devices. Unfortunately, social networks also provide the means for distributing abusive and aggressive content. Given the amount of information generated every day on social media, it is not possible for humans to identify and remove such messages manually, instead it is necessary to employ automatic methods. Recently, many shared tasks have been introduced to encourage the development of methods capable of classifying messages from social media as offensive. As an example, the First Workshop on Trolling, Aggression and Cyberbullying has organised the First Shared Task on Aggression Identification to classify messages from Facebook and Twitter into three categories Overtly Aggressive (OAG), Covertly Aggressive (CAG) and Non-aggressive (NAG) (Kumar et al., 2018). The task was organised for English and Hindi.

Recently, more complete dataset covering different aspects of offensive identification was released for the shared task in SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. The task was not only to identify offensive messages in social media. The participants had to categorize the offensive language and also had to identify the targeted audience (Zampieri et al., 2019). We used this dataset to experiment our novel architecture since it covers more aspects in offensive language detection in social media. More details about the tasks and the dataset will be discussed in Section 2.

People from all over the world uses social media. Therefore, a random sample of social media messages can be written in several languages. As a result, systems that detect offensive posts without relying too much on language dependent features would be valuable in real life scenarios. In the offensive language identification shared tasks too, most researches have worked on systems that rely on word/character embeddings rather than linguistics features. As an example Galery and Charitos (2018) has taken an approach to feed fasttext (Mikolov et al., 2018) character embeddings to a Gated Recurrent Neural Network architecture (Chung et al., 2014). As the system doesn't rely on linguistic features, it is easily portable between Hindi and English. These type of architectures can be easily implemented for other languages once the data for training is available.

Most approaches in shared tasks are based on word/character embeddings feeding to a neural network (Kumar et al., 2018). Most of these architectures use max pooling or successive convolutional layers that reduce spacial size of the data flowing through the network and therefore increase the view of higher layers neurons, allowing them to detect higher order features in a larger region of the input embeddings. However recent introduction of capsule networks shows that while pooling works better in most of the scenarios, it nonetheless is losing valuable information (Hinton et al., 2018). The solution that has been brought forward is Capsule Networks. How it overcomes the weaknesses in max pooling layer will be discussed in Section 3.

Since the Capsule Networks are very new to the field, they have not been used much in NLP tasks. However, their good performance in image classification tasks motivated us to use them in offensive language detection tasks too. To the best of our knowledge, no prior work has been explored in offensive language identification with Capsule Networks. Also, it might be important to explore how the Capsule Networks performs in NLP domain. Additionally we analyzed that most of the social media posts contain not only text but emojis too, which can be a contributing factor for offense. Therefore we propose a method to incorporate emoji knowledge to the Capsule Network architecture. Generally, this paper proposes a Capsule Network architecture with emoji information to detect offensive posts in social media. The rest of the paper is organised as follow. Section 2 would briefly describe the tasks and the dataset. Section 3 would describe the capsule network architecture we used and how we integrated emoji information to the architecture. After that we evaluate the system comparing with the architectures provided in Zampieri et al. (2019). Finally, the conclusions are presented.

2 Dataset and Task Description

Dataset that we used was released for the Task 6 in SemEval-2019 : OffensEval: Identifying and Categorizing Offensive Language in Social Media. It has been collected from Twitter using its API and searching for keywords and constructions that are often included in offensive messages, such as she is, to:BreitBartNews, gun control etc (Zampieri et al., 2019).

There were three tasks associated with the shared task.

- *Subtask A: Offensive language Detection* : Goal of the task was to discriminate between the following types of tweets:
  - **Not Offensive (NOT)** : Posts that do not contain offense
  - **Offensive (OFF)**: Posts containing any form of non-acceptable language. These posts can include insults, threats swear words etc. (Zampieri et al., 2019)
- *Subtask B: Categorization of Offensive Language :* Task's goal was to categorize the type of offense.
  - **Targeted Insult (TIN)** : Posts that contain targeted insults and threats.
  - **Untargeted (UNT)** : Posts containing non targeted insults or threats.
- *Subtask C: Offensive Language Target Identification :* Goal of the task was to categorize the targets of insults/threats.
  - **Individual (IND)** : Insults that target individuals.
  - **Group (GRP)** : Insults that target a group of people.
  - **Other (OTH)** : The target does not belong to any category mentioned above.

Few examples from the training set is shown in Table 1.

As you can see, the nature of the three tasks are different and it would interesting to explore how one architecture can be used to capitalise all of the three tasks.

3 Research Approach

We first describe the existing approaches mentioned in Zampieri et al. (2019). Then we will describe the proposed capsule network architecture.

3.1 Existing Approaches

There are three approaches considered in Zampieri et al. (2019) which will be described in the following list. We describe them briefly in this sub section before introducing the capsule network architecture.

1. **SVM** - A linear SVM trained on word unigrams. SVMs have achieved state-of-the-art results for many text classification tasks (Zampieri et al., 2018).

2. **BiLSTM** - The model consists of (i) an input embedding layer,(ii) a bidirectional LSTM layer (Schuster and Paliwal, 1997), and (iii) an average pooling layer of input features. The concatenation of the LSTM layer and

id	tweet	a	b	c
44209	@USER @USER what a baby! URL	NOT	NULL	NULL
97670	@USER Liberals are all Kookoo !!!	OFF	TIN	OTH
74831	@USER Trump kicks dem butt - its so fun.	OFF	TIN	IND
17259	IM FREEEEE!!!! WORST EXPERIENCE OF MY FUCKING LIFE	OFF	UNT	NULL

Table 1: Example rows from the dataset

the average pooling layer is further passed through a dense layer, whose output is ultimately passed through a softmax to produce the final prediction. The model is adapted from a pre-existing model for sentiment analysis (Rasooli et al., 2017).

3. **CNN** - A convolutional neural network based on the model proposed in Kim (2014). It consists of an (i) an input embedding layer, (ii) a convolutional layer (Collobert et al., 2011) and (iii) a max pooling layer (Collobert et al., 2011) of input features. The output of the max pooling layer is further passed through a dropout (Srivastava et al., 2014) and softmax output.

Both BiLSTM and CNN architectures above have pooling layers which is a very primitive type of routing mechanism. The most active features in a local pool is routed to the higher layer and the higher-level detectors don't have an impact in the routing. However in the Capsule Network, only those features that agree with high-level detectors are routed. It has a superior dynamic routing mechanism. With this advantage we propose a novel capsule network architecture for aggression detection which will be described in the next section.

3.2 Proposed Architecture

The proposed architecture is depicted in Figure 1. The architecture consists of four layers.

1. *Embedding Layer* - We represent every text x_i , as a sequence of one-hot encoding of its words, $x_i = (w_1, w_2, ...w_n)$ of length n, which is the maximum length of the all of the texts in the training set, with zero padding. Such a sequence becomes the input to the embedding layer. Most of the words exist in social media texts are not proper words. If we used word embeddings to initialize the embedding matrix in the embedding layer, most of the words that are fed will be out-of-vocabulary words. Therefore we used character embeddings (Mikolov et al., 2018) as it provides embeddings for misspelling words and new words. Also character embeddings handle infrequent words better than word2vec embedding as later one suffers from lack of enough training opportunity for those rare words. We used fasttext embeddings pre trained on Common Crawl (Mikolov et al., 2018). Using the model we represented each word as a vector with a size of 300 values. The embedding layer is improved more with emoji information, which will be described in Section 3.3.

2. *Feature Extraction Layer* - We used this layer to extract long term temporal dependencies within the text. We experimented both LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Chung et al., 2014) for this layer. Due to the fact that we had a small number of training examples, GRUs performed better than LSTMs, capitalising on GRU's ability to exhibit better performance on smaller datasets. For the final architecture we used a bi directional GRU layer with 50 time steps, each getting initialised with glorot normal initialiser.

3. *Capsule Layer* - The Capsule layer we used is primarily composed of two sub-layers *Primary Capsule Layer* and *Convolutional Capsule Layer*.

   (a) *Primary Capsule Layer* - The primary capsule layer is supposed to capture the instantiated parameters of the inputs, for example, in case of texts, local order of words and their semantic representation is captured with the primary capsule layer.

   (b) *Convolutional Capsule Layer* - The convolutional capsule layer outputs a lo-

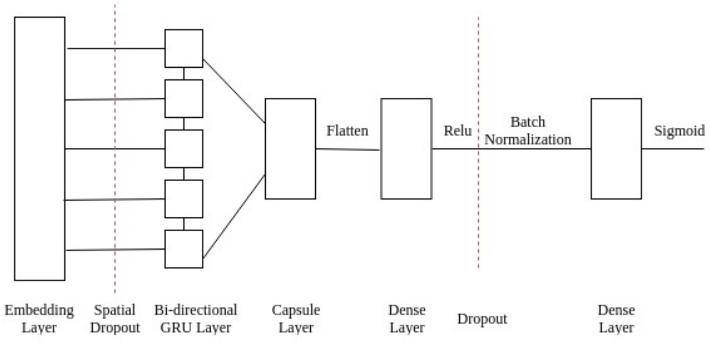


Figure 1: Capsule Network

cal grid of vectors to capsules in earlier layers using different transformation matrices for each capsule. This is trained using dynamic routing algorithm described in Sabour et al. (2017) that overlooks words that are not important or unrelated in the text, like stopwords and name mentions which are common in social media texts.

4. *Dense Layers* - Output of the Capsule layer is flattened and then fed in to two dense layers. First dense layer had 100 units and was activated with relu function. After applying batch normalization to the output of first dense layer, it was fed in to the second dense layer with 1 unit and and sigmoid activation.

Apart from the major sections in the architecture described above, we used a spatial dropout (Tompson et al., 2015) between the embedding layer and the feature extraction layer and a dropout (Srivastava et al., 2014) between the two dense layers to minimize over fitting of the network. The implementation was done using Keras (Chollet et al., 2015) and Python[1].

The next section would describe how we integrated emoji knowledge to this architecture.

3.3 Integrating Emojis

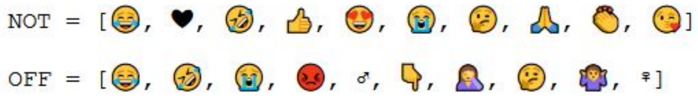
Emojis are ideograms which are used with text to visually complement its meaning. In present, emojis are widely used by social media. A global analysis done on Twitter has been found that 19.6% of tweets contain emojis. Further it stated, emojis are used by 37.6% of users (Ljubesic and Fiser, 2016). A research conducted by (Barbieri et al., 2017) has been showed that there is an unique and important relation between sequences of words and emojis. When analyze the top 10 emojis belong to both categories; Offensive and Not Offensive, in the selected dataset, it also shows a clear distinction of emojis corresponding to its category as shown in Figure 2. Due to the extensive usage of emojis in social media and the relationship lie between emojis and text, integration of emojis can be used to improve the social media offensive language detection.

Since the proposed architecture is based on embeddings, we decided to integrate emojis also using the embeddings. But most of the available pre-trained word embedding sets include few or no emoji representations. Therefore in addition to the character embeddings, separate embedding set; emoji2vec (Eisner et al., 2016) was chosen for emojis. Emoji2vec consists of pre-trained embeddings for all Unicode emojis using their descriptions in the Unicode emoji standard. This maps emojis into 300-dimensional space similar to other available word embeddings; word2vec, glove, etc. to make the integration easy with word vectors. Emoji2vec embeddings were evaluated based on sentiment analysis on tweets and it showed word2vec with emoji embeddings advances the classification accuracy while proving that the emoji2vec embeddings are useful in social natural language processing tasks.

Following (Eisner et al., 2016), there were two pre-trained emoji2vec models[2]. One model is based on the sum of vectors corresponds to the words found in phrases which describe the emojis. As an extended version of it, other model feeds the actual word embeddings to an LSTM layer. We

[1] The code is available on ”https://github.com/TharinduDR/Aggression-Identification”

[2] https://github.com/uclmr/emoji2vec



```
NOT = [😂, ♥, 🤣, 👍, 😍, 😭, 🤔, 🙏, 👏, 😘]
OFF = [😂, 🤣, 😭, 😡, ♂, 👇, 🤦, 🤔, 🤷, ♀]
```

Figure 2: Top 10 emojis belong to Not Offensive (NOT) and Offensive (OFF) posts, Task 6 Dataset, SemEval-2019

Model	NOT			OFF			Weighted Average			
	P	R	F1	P	R	F1	P	R	F1	F1 Macro
SVM	0.80	0.92	0.86	0.66	0.43	0.52	0.76	0.78	0.76	0.69
BiLSTM	0.83	0.95	0.89	0.81	0.48	0.60	0.82	0.82	0.81	0.75
CNN	0.87	0.93	0.90	0.78	0.63	0.70	0.82	0.82	0.81	0.80
CapsuleNet †	0.88	0.93	0.91	0.82	0.64	0.71	0.83	0.82	0.82	**0.81**
All NOT	-	0.00	0.00	0.72	1.00	0.84	0.52	0.72	0.	0.42
All OFF	0.28	1.00	0.44	-	0.00	0.00	0.08	0.28	0.12	0.22

Table 2: Results for offensive language detection. We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (NOT, OFF), and weighted averages. Macro-F1 is also listed (best in bold). † denotes the capsule net architecture integrated with emoji embeddings.

used 300 dimensional embedding spaces generated by both models; sum based and LSTM based for this experiment.

Two approaches were used to integrate emoji embeddings with the above mentioned architecture as follows:

1. *300 dimensional embedding layer* - Shared same 300 dimensional vector space for both word and emoji embeddings.

2. *600 dimensional embedding layer* - Used concatenation layer and resulted 600 dimensional vector space by both word and emoji embeddings.

Among the experiments we conducted using both emoji2vec models and integration approaches, combination of sum based emoji embeddings with 600 dimensional embedding layer and LSTM based emoji embeddings with 300 dimensional embedding layer resulted improvements compared to word embeddings only approaches. More details on experiment results are mentioned in Section 4.

3.4 Training

The network was trained on the training dataset provided for SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. It was trained using adam optimiser (Kingma and Ba, 2015), with a reduced learning rate once learning stagnates. Also the parameters of the network was optimized using five fold cross validation.

4 Evaluation

The capsule network architecture we proposed above was evaluated using the testing set provided for each of the subtask in SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media.

4.1 Offensive Language Detection

The performance on identifying offensive (OFF) and non-offensive (NOT) posts is reported in Table 2. The Capsule Network we proposed outperforms the RNN model, achieving a macro-F1 score of 0.81.

4.2 Categorization of Offensive Language

The results for the offensive language categorization is shown in Table 3. In this subtask too Capsule Network architecture outperforms all the other models having a macro F1 score of 0.71.

4.3 Offensive Language Target Identification

The results for the offensive language target identification is shown in Table 4. Capsule Network architecture outperforms all the other models having a macro F1 score of 0.49.

Model	TIN			UNT			Weighted Average			
	P	R	F1	P	R	F1	P	R	F1	F1 Macro
SVM	0.91	0.99	0.95	0.67	0.22	0.33	0.88	0.90	0.88	0.64
BiLSTM	0.95	0.83	0.88	0.32	0.63	0.42	0.88	0.81	0.83	0.66
CNN	0.94	0.90	0.92	0.32	0.63	0.42	0.88	0.86	0.87	0.69
CapsuleNet †	0.96	0.94	0.94	0.33	0.67	0.44	0.90	0.88	0.87	**0.71**
All TIN	0.89	1.00	0.94	-	0.00	0.00	0.79	0.89	0.83	0.47
All UNT	-	0.00	0.00	.11	1.00	0.20	.01	0.11	0.02	0.10

Table 3: Results for offensive language categorization. We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (TIN, UNT), and weighted averages. Macro-F1 is also listed (best in bold). † denotes the capsule net architecture integrated with emoji embeddings.

.

Model	GRP			IND			OTH			Weighted Average			
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	F1 Macro
SVM	0.66	0.50	0.57	0.61	0.92	0.73	0.33	0.03	0.05	0.58	0.62	0.56	0.45
BiLSTM	0.62	0.69	0.65	0.68	0.86	0.76	0.00	0.00	0.00	0.55	0.66	0.60	0.47
CNN	0.75	0.60	0.67	0.63	0.94	0.75	0.00	0.00	0.00	0.57	0.66	0.60	0.47
Capsule Net †	0.78	0.65	0.70	0.64	0.95	0.78	0.02	0.03	0.02	0.59	0.68	0.62	**0.49**
All GRP	0.37	1.00	0.54	-	0.00	0.00	-	0.00	0.00	0.13	0.37	0.20	0.18
All IND	-	0.00	0.00	0.47	1.00	0.64	-	0.00	0.00	0.22	0.47	0.30	0.21
All OTH	-	0.00	0.00	-	0.00	0.00	0.16	1.00	0.28	0.03	0.16	0.05	0.09

Table 4: Results for offense target identification. We report Precision (P), Recall (R), and F1 for each model/baseline on all classes (GRP, IND, OTH), and weighted averages. Macro-F1 is also listed (best in bold). † denotes the capsule net architecture integrated with emoji embeddings.

As shown in Table 2, 3 and 4 capsule network architecture outperformed all the other models in all sub tasks. It is worth noticing that the unbalanced nature of the dataset did not affect the performance of the capsule network architecture. Also eventhough the capsule layer is seemingly complex, results show that it does not need a large training set to optimize its parameters.

We did not fine tune the model analyzing data in this dataset since we wanted a general model capable of identifying offense. Hence, we did not compare our results with the final results of the shared task.

5 Conclusion

We have presented a novel capsule network architecture to detect type and target of offensive posts in social media. Also we propose a method to incorporate emoji knowledge to the architecture. Our approach was able to improve on the baseline system presented at SemEval-2019 Task 6: OffensEval: Identifying and Categorizing Offensive Language in Social Media. Importantly our system does not rely on language dependent features so that it is portable for any other language too.

The main conclusion of the paper is that even though the capsule networks are not widely used in NLP domain, they can achieve state of the art results. Also with the shown way of integrating emoji information to the network, results can improve.

In the future we hope to implement a multi purpose capsule network architecture for several tasks in NLP domain such as spam detection, gender identification etc. We hope to further explore capsule network architectures in various NLP tasks.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *Arxiv* .

François Chollet et al. 2015. Keras. https://keras.io.

Junyoung Chung, aglar Gülehre, Kyunghyun Cho, and

Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their descriptions. *Arxiv* .

Thiago Galery and Efstathios Charitos. 2018. Aggression identification and multi lingual word embeddings. In *TRAC@COLING 2018*.

Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with em routing. In *ICLR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9:1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *TRAC@COLING 2018*.

Nikola Ljubesic and Darja Fiser. 2016. A global analysis of emoji usage. In *WAC@ACL*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mohammad Sadegh Rasooli, Noura Farra, Axinia Radeva, Tao Yu, and Kathleen McKeown. 2017. Cross-lingual sentiment transfer with limited resources. *Machine Translation* 32:143–165.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. *ArXiv* abs/1710.09829.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing* 45:2673–2681.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pages 648–656.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardzic, Nikola Ljubesic, Jörg Tiedemann, Chris van der Lee, Stefan Grondelaers, Nelleke Oostdijk, Dirk Speelman, Antal van den Bosch, Ritesh Kumar, Bornini Lahiri, and Mayank Jain. 2018. Language identification and morphosyntactic tagging: The second vardial evaluation campaign. In *VarDial@COLING 2018*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Proceedings of NAACL*.

EoANN: Lexical Semantic Relation Classification Using an Ensemble of Artificial Neural Networks

Rayehe Hosseinipour, Mehrnoosh Shamsfard
Computer Engineering Department, Shahid Beheshti university, Tehran, Iran
Computer Engineering Department, Shahid Beheshti university, Tehran, Iran
r.hosseinipour@mail.sbu.ac.ir
m-shams@sbu.ac.ir

Abstract

Researchers use wordnets as a knowledge base in many natural language processing tasks and applications, such as question answering, textual entailment, discourse classification, and so forth. Lexical semantic relations among words or concepts are important parts of knowledge encoded in wordnets. As the use of wordnets becomes extensively widespread, extending the existing ones gets more attention. Manual construction and extension of lexical semantic relations for WordNets or knowledge graphs are very time consuming. Using automatic relation extraction methods can speedup this process.

In this study, we exploit an ensemble of LSTM and convolutional neural networks in a supervised manner to capture lexical semantic relations which can either be used directly in NLP applications or compose the edges of wordnets. The whole procedure of learning vector space representation of relations is language independent. We used Princeton WordNet 3.1, and FarsNet 3.0 (the Persian wordnet), as gold standards to evaluate the predictive performance of our model and the results are comparable on the two languages. Empirical results demonstrate that our model outperforms the state-of-the-art models.

1 Introduction

Lexical semantic relation classification is the task of identifying s semantic relation**(s)** which holds between word pairs among a set of predefined relation types. Relation classification can be done in a supervised manner, using a dataset, labeled with a certain number of relation classes. In addition to classification with known relations, there are some methods which go even further and learn new semantic relations and suggest new relation categories (Shamsfard and Barforoosh, 2003).

Relation identification plays an essential role in many natural language processing application such as question answering, recognizing textual entailment and discourse understanding.

There are two main approaches for classification of lexical semantic relations; distributional and path-based (Wang et al., 2017).

Path-based approaches try to recognize the type of semantic relation between word pairs according to their co-occurrence information in the corpus. These methods mainly use the dependency path between word pairs as their input feature (Snow et al., 2004; Riedel et al., 2013). As Ziph's law states that most of the words in vocabulary rarely occur in the corpus (Powers, 1998) these methods have some limitation for word pairs who do not co-occur in a context.

On the other hand according to the distributional hypothesis which states "words that occur in similar contexts tend to have similar meanings" (Harris, 1954), distributional approaches try to recognize the relation between words based on their separate occurrence in the corpus which can

Proceedings of Recent Advances in Natural Language Processing, pages 481–486,
Varna, Bulgaria, Sep 2–4 2019.

be represented for example by their word embedding vectors (Mikolov et al.,2013) and these methods have shown great performance (Baroni et al., 2012; Turney and Pantel, 2010; Roller et al., 2014).

In the last decades, several researches have been conducted on discovering hypernymy as an example of lexical semantic relations, and a key part of taxonomies and state-of-the-art models show significant results (Shwarts et al., 2016; Roller et al., 2016).

However, other types of relations have been less investigated.

Several types of models have been used for the task of semantic relation classification, but the results are not sufficiently admissible (Vu and Shwarts, 2018).

In this paper, we use an ensemble of models to improve prediction performance of relation classification. The idea of ensemble methodology is to combine some weighted classifiers in order to obtain a more accurate one (Rokach et al. 2009).

Main building blocks of this combinational model are some inducers named weak learner which perform slightly better than random. According to Condorcet Jury theorem which states " the ensemble of independent voters each of which performs better than random (p>0.5) has a probability of L>p to make the right decision." we used different structured neural networks as the model's weak learners.

The input of our model is a concatenation of word embedding vectors corresponding to target word pairs, and the output is a class label predicted based on learned vector space distributional representation of the semantic relation which holds between them.

Our final model has the best validation F1 score of 0.894 in predicting the relation between FarsNet (Shamsfard, et al., 2010) word pairs and 0.768 to predict Princeton Wordnet (Miller, 1995; Felbaum 1998) relation classes.

We summarize the contribution of this paper as follow:

- We propose EoANN, an ensemble of artificial neural networks for classifying all types of lexical semantic relations in target datasets, without any hand-crafted features.

- Our model addresses the sparseness issue and can classify word pairs which do not necessarily co-occur in the corpus.

- According to human expert reviews, our model goes beyond relation discovery and can be employed to correct the potential error in wordnet edges and suggest new missed relation instances.

The rest of this paper is structured in 6 sections:
Section 2 presents the existing approaches for the classification of lexical semantic relations; the next one presents our model in detail, section 4 describes the data set we used for evaluating our model, section 5 reports experimental results and finally section 6 dedicated to the conclusion and future works.

2 Related Work

There are two main lexical semantic relation extraction models, distributional and path-based (pattern-based) (Wang et al., 2017) and also there are methods that use an integration of these two approaches (Shwartz et al., 2016).

Distributional methods learn the relation between word pairs based on the disjoint occurrence of them. These methods usually use a combination of word embedding vectors (Mikolov et al., 2016) as their input features. Considering v1 and v2 being word embedding vector corresponding to w1 and w2, most common combinations are:

- concatenation of v1 and v2 (**Concat**)

- the offset of v1 and v2 (**Offset**)

- point-wise multiplication of v1 and v2 (**Mult**)

- squared difference between v1 and v2 (**Sqdiff**)

Offset (Roller et al., 2014; Weeds et al., 2014; Fu et al., 2014), **Concat** (Baroni et al., 2012) and **Concat**+**Offset** (Washio and Kato, 2018) is the most common type of feature vector combination which is used in this task. To capture the different notion of interaction information about relation Vu and Shwartz (2018) add **Mult**, studied by Weeds et al. (2014) and **Sqdiff** introduced by themselves as input feature and report **Mult**+**Concat** performs better than other combination.

These methods mostly focus on lexical entailment and relation classes such as hypernym, causality and other instances of relation which exemplified inference and have a state-of-the-art F1 score of 0.91.

Path-based or pattern-based methods utilize features derived from the context in which word pairs co-occur. For example, the dependency path between a word pair and observed predefined patterns are used as an informative feature to classify the relation. The methods of this category are limited to use only the word pairs that co-occur in corpus (Hearst et al., 1992; Snow et al., 2004; Navigli and Velardi 2010; Shamsfard et al,. 2010; Boella and Di Caro, 2013; Pavlick and Pasca, 2017)

Recently some approaches use an integration of these two methods and combine both distributional and dependency path information to obtain better results. HypNet (shwarts et al., 2016) is an examples of these approaches.

3 Our Model

In this paper, we propose a model to classify lexical semantic relations between a word pair using their word embedding vectors.

The rarity of co-occurring every candidate word pair which possibly involves in a semantic relation leads us to exploit a method which does not necessarily need to see the word pair in a context together.

The output of our model is a class label prediction based on learned vector space distributional representation of the semantic relation which holds between target word pairs.

Although using a single deep neural network (as a distributional method) showed some improvement in capturing semantic relations, in order to get the advantage of the diversity among predictions of separately trained models, we use an ensemble of two artificial neural networks.

The ensemble is a general statistical enhancing technique to improve the representational capacity of the model. This enhancement helps to find a hypothesis which is independent of the space of the model from which it starts to learn.

First, we train two neural networks separately on data our labeled data and evaluate their test results, then put these two models in an ensemble and re-evaluate the result. Comparing two result sets shows 0.1 improvement in F1 score of learned hypothesis.

Models can be assembled in many different ways like boosting, bagging and stacking. We use stacking which involves training a learning algorithm to combine the predictions of several learning algorithms.

The advantage of stacking is to increase the prediction power of the classifier. As the using of another neural network above the weak learners in order to learn the final prediction imposes excess overhead, we use the simplest stacking method which is averaging. Averaging has no parameter, so no training is needed.

We transfer the input embedding vector of word pairs to dense-valued feature vectors, next feed these vectors to both ANN to compose their own distributional representation of them. At the final layer of each, a softmax classifier predicts the label of input sample.

Finally, a weighted averaging mechanism is used to decide the relation class in which input words participate.

3.1 Input of EoANN

Our inputs are raw lexical entries (multi word expressions are excluded) of Wordnets. We first transform every single word to its embedding vectors using word embedding.

Word embedding is a method to map words and phrases from space with one dimension per word, to a continuous low dimensional vector space. There are many word embedding frameworks. We use Fasttext (Piotr et al., 2017) which represents words as the sum of the n-gram vectors. This method is actually an extension of the continuous skip-gram model (Mikolov et al., 2013), which considers sub-word information as well. We denote the word embedding vector of word w by vw ∈ ℝ

Given *R (a, b)* as a sample of semantic relation triple in target Wordnet, R is the class of relation which connects a to b and v_a and v_b are the embedding vectors corresponding to them. The input vector and labels of our classifiers is the concatenation of word vectors:

$$h1(a,b) = [va{:}\,vb]$$
$$out(a,b) \; = \; R$$

3.2 Weak Learners Structure

We use both convolutional neural network and LSTM network in the simplest structure as our model base inducers. These two learners are chosen because of their power in capturing of hierarchical patterns and the extraction of the temporal behavior.

The simple CNN inducer is composed of 3 main layers, a convolutional layer with 20 filters of size (1, 2), a pooling layer which is used to reduce the dimensions of feature map and finally a fully connected layer that flattens the results and passes it to a softmax classifier to decide which relation class the input belongs to.

LSTM neural network which we use as another weak learner is composed of a fully connected layer to encode 2-dimentional input feature vector to a dense flat vector, then passes its output to a LSTM layer with 200 memory units and a softmax classifier finally decides about data class label.

Combiner is responsible for getting the final decision by combining individual classifiers predictions. This component holds a majority voting among classifiers and declares the ultimate predicted label.

4 Datasets

In this study, we use four common data set to evaluate the performance of our model, FarsNet (Shamsfard, 2008), Princeton Wordnet (Miller, 1996), Root09 (Santus et al., 2015) and EVALution (Santus et al., 2016) as common semantic relation resources in Persian and English.

Table 1 shows the details about datasets, their relation class and number of instances used for train and test (90% for train and validation and 10% for test).

For embedding model, we used the Wikipedia dump in both English and Persian Languages. Our English word embedding model vocabulary contains 999,994 words and our Persian model has 347,636 words.

Fasttext embedding models had the following parameter set for training:

- Vector dimention:300
- Learning rate:0.04
- Min and max length of char n-gram:[3,6]
- Number of epochs:10

The rest of the parameters are as Fasttext default configuration.

data set	relation classes	# of instances
WordNet	Hypernym, Hyponym, Entailment, Cause, Instance-Hypernym, Instance-Hyponymy, Member, Holonym, Attribute	634,330
Farsnet	Hypernym, Hyponym, Antonym, Instrument, Domain, Instance-Hypernym, Instance-Hyponym Location, Patient	322,554
ROOT09	**Hypernym, co-Hyponym, Random**	**12,762**
EVALution	Hypernym, Antonym, meronym, possession, Attribute, Part Of	7,378

Table 1: data sets we use for evaluating our model, their main relation categories and the number of relation instances of each

5 Experimental Results

We use four wordnet-like data sets as our benchmark to evaluate the performance of our model:

We compared the results on root09 and EVALution with two most recent work, LexNet proposed by Vu and Schwartz (2018) and KSIM previously used and reported to be successful by Levy et al. (2015). We also compared our model performance with the previous effort result in extracting FarsNet relation in Persian which is a semi-automated pattern-based approach (Shamsfard et al., 2010).

Our experimental results which are summarized in table 2, show that our model can classify FarsNet word pairs relations with F1 score of 0.894 which is significant and it has an average F1 of 0.768 for WordNet relation classification.

As shown in table 2 the state-of-the-art models in the best case, has the F1 score of 0.606 on detecting relations in EVALution and 0.81 in ROOT09 and our model with F1 score of 0.655 for first and F1 score of 0.868 for last outperforms these methods.

Model	Data Set	Classifier feature composition	**F1**
EoANN	Root09	LSTM+**CNN Concat**	**0.868**

	EVALution	LSTM+**CNN Concat**	**0.655**
LexNet	Root09	RBF **Sum**+**SqDiff**	0.814
	EVALution	RBF **Concat**+**Mult**	0.6
KSIM	Root09	RBF **Sum**+**SqDiff**	0.723
	EVALution	RBF Concat+Mult	0.505

Table 2: best precision recall and F1 score for root09 and EVALution in 4 compared models

model	dataset	Model and Features	F1 score
EoANN	farsnet	LSTM+CNN **Concat**	**0.894**
Semi-auto	farsnet	Pattern-based +structured_based+ statistical	0.605

Table 3: best F1 score on farsnet

6 Conclusion and Future Works

Our objective in this research was to automatically classify lexical semantic relation employing the power of the simple but effective structured neural networks, which have shown their proficiency in many tasks of natural language processing (Collobert et al., 2011; Yao et al., 2013).

We used both LSTM and convolutional network to benefit the exhibition of temporal behavior by first and the extraction of the hierarchical pattern by last.

We also used the simplest distributional feature as input and entrusted the extraction of the most proper composition of features to the model.

In case of ROOT09 and EVALution our model has an improvement of 0.05 in F1 score from state of the art (LexNet). And for FarsNet dataset we have 0.11 improvement in F1 score.

The next step in extending lexical ontologies is to complete missed relation edges, then to learn new relation classes, which can be added to the target wordnet.

References

George A. Miller. 1995. WordNet: a lexical database for English. Communications of the ACM, 38(11):39-41.

Christiane Fellbaum 1998. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Mehrnoush Shamsfard, Akbar Hesabi, Hakimeh Fadaei, Niloofar Mansoory, Ali Famian, Somayeh Bagherbeigi, Elham Fekri, Maliheh Monshizadeh and Mostafa Assi. 2010. Semi-automatic development of farsnet; the Persian wordnet. Proceedings of 5th global WordNet conference, Mumbai, India; 29.

David Austen-Smith, Jeffrey S. Banks. 1996. Information aggregation rationality and the Condorcet jury theorem, American Political Science Review, vol. 90, pages. 34-45.

Mehrnoush Shamsfard and AA Barforoush. 2003. An introduction to HASTI: an ontology learning system. Proceedings of the iasted international conference artificial intelligence and soft computing, Acta Press, Galgary, Canada: 242-247.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In NIPS.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karle, Koray Kavukcuoglu, Pavel Kuks. 2011. Natural language processing (almost) from scratch. Journal of machine learning research, pages 2493-537

Sebastian Riedel, Limin Yao, Andrew McCallum, and M. Benjamin Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In Proceedings of NAACL-HLT 2013, pages 74–84.

David M. Powers 1998. Applications and explanations of Zipf's law. Association for Computational Linguistics: 151–160.

Zellig S. Harris. 1954. Distributional Structure. WORD, 10(2-3):146-162.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In NIPS, pages 3111–3119.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chungchieh Shan. 2012. Entailment above the word level in distributional semantics. In Proceedings of EACL 2012, pages 23–32.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research, 37:141–188.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In Proceedings of COL- ING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1199–1209, Baltimore, Maryland. Association for Computational Linguistics.

Koki Washio and Tsuneaki Kato. 2018. Neural Latent Relational Analysis to Capture Lexical Semantic Relations in a Vector Space. arXiv preprint arXiv:1809.03401.

Chengyu Wang, Xiaofeng He, and Aoying Zhou. 2017. A short survey on taxonomy learning from text corpora: Issues, resources and recent advances. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1190– 1203.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1025– 1036, Dublin, Ireland.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2389–2398, Berlin, Germany. Association for Computational Linguistics.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting Hearst patterns in distributional vectors for lexical entailment. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2163–2172, Austin, Texas. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th conference on Computational linguistics, pages 539–545.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In ACL, pages 1318–1327.

Guido Boella and Luigi Di Caro. 2013. Supervised learning of syntactic contexts for uncovering definitions and extracting hypernym relations in text databases. In Machine learning and knowledge discovery in databases, pages 64–79. Springer.

Ellie Pavlick and Marius Pasca 2017. Identifying 1950s American jazz musicians: Fine-grained isa extraction via modifier composition. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2099–2109. Association for Computational Linguistics.

Tu Vu and Vered Shwartz. 2018. Integrating Multiplicative Features into Supervised Distributional Methods for Lexical Entailment. arXiv preprint arXiv:1804.08845.

Lior Rokach. 2009. Ensemble-based classifiers. Artificial Intelligence Review, 33(1-2):1-39.

Bojanowski Piotr, Edouard Grave, Armand Joulin and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In Proceedings of the 4th Workshop on Linked Data in Linguistics (LDL- 2015), pages 64–69.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers, pages 38–42, Gothenburg, Sweden. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 970–976, Denver, Colorado. Association for Computational Linguistics.

Opinions Summarization: Aspect Similarity Recognition Relaxes the Constraint of Predefined Aspects

Huy Tien Nguyen
Japan Advanced Institute of Science and Technology (JAIST), Japan
ntienhuy@jaist.ac.jp

Tung Le
Japan Advanced Institute of Science and Technology (JAIST), Japan
lttung@jaist.ac.jp

Minh Le Nguyen
Japan Advanced Institute of Science and Technology (JAIST), Japan
nguyenml@jaist.ac.jp

Abstract

Recently research in opinions summarization focuses on rating expressions by aspects and/or sentiments they carry. To extract aspects of an expression, most studies require a predefined list of aspects or at least the number of aspects. Instead of extracting aspects, we rate expressions by aspect similarity recognition (ASR), which evaluates whether two expressions share at least one aspect. This subtask relaxes the limitation of predefining aspects and makes our opinions summarization applicable in domain adaptation. For the ASR subtask, we propose an attention-cell LSTM model, which integrates attention signals into the LSTM gates. According to the experimental results, the attention-cell LSTM works efficiently for learning latent aspects between two sentences in both settings of in-domain and cross-domain. In addition, the proposed extractive summarization method using ASR shows significant improvements over baselines on the Opinosis corpus.

1 Introduction

Opinions Summarization is the collection of typical opinions mentioned in social media, blogs or forums on the web. This task helps customers to absorb better a large number of comments and reviews before making decisions as well as producers to keep track of what customers think about their products (Liu, 2012).

Due to the fast growth of data over the Internet, automatically opinions summarization has received a lot of attention in recent years. Most research focus on extractive summarization, where the most salient text units are identified and construct a summary. Ranking candidates for generic summarization usually bases on various handcrafted features such as sentence position and length (Radev et al., 2004), word frequency (Nenkova et al., 2006) or using neural networks for learning salient scores (Zhou et al., 2018).

In opinions summarization, however, this task is required to consider aspects and/or sentiments of text candidates for generating a concise and informative summary (Hu and Liu, 2006). The popular framework of this problem involves three subtasks (Hu and Liu, 2004): i) aspect discovery which extracts the properties of interested entities (e.g., battery life, design, customer service); ii) sentiment analysis which assigns sentiment polarity (positive and negative) towards the aspects extracted in the first step; and iii) summary generation which selects the most salient opinions to build a summary.

For the aspect discovery task, there are two main techniques: supervised and unsupervised learning. The former models the aspect extraction as a sequence labeling task. Due to predefining a list of aspect and heavily relying on annotated data, this approach suffers from domain adaptation problems. The latter uses a large amount of unlabeled data for abstracting aspects via the statistical topic modeling LDA (Blei et al., 2003) or the aspect-based autoencoder model (He et al., 2017a). However, these unsupervised techniques have limitations. First, we have to decide on a suitable number of aspects for each domain. Second, the existing methods require a sufficient amount of data while some domains may not have enough reviews, known as the cold-start problem (Moghaddam and Ester, 2013).

In extractive opinions summarization, most existing approaches use the aspects information for discarding potentially redundant units. For minimizing repeated information on the same aspect, we only need to identify whether two text

No.	Sentence	Aspect Similarity
1	The pc runs so fast. I like its performance and price.	Yes
2	The food is cheap. The shop's location is good.	No
3	I love its pizza. I bought food from the restaurant.	Yes

Table 1: Some samples of Aspect Similarity Recognition

units have at least one aspect in common, which is called Aspect Similarity Recognition - ASR (Nguyen et al., 2018), rather than explicitly extracting aspects of each text unit. Table 1 shows some samples of the ASR task. Follow this observation, we propose an aspect-based summarization using ASR instead of aspect discovery. The advantage of ASR is to learn patterns and relations between two text units and not need to identify the aspects of each unit, therefore it is potential to cross-domain application. Our contributions in this work are as follows:

- We propose an attention-cell LSTM model (ACLSTM) for ASR which enhances the LSTM model via employing attention signals into the input gate and the memory cell. ACLSTM shows improvements compared to the conventional attention models for both settings of in-domain and cross-domain.

- We introduce a novel aspect-based summarization using Aspect Similarity Recognition. According to the experiments, our method outperforms strong baselines on Opinosis corpus. We also evaluate our method in regard to domain adaptation.

The remainder of this paper is organized as follows: Section 2 reviews the related research, Section 3 describes the problem formulation, Section 4 and 5 respectively introduce the attention-cell LSTM for ASR and the proposed summarization using ASR, Section 6 discusses the experiments for ASR and summarization, and Section 7 concludes our work and future work.

2 Related Work

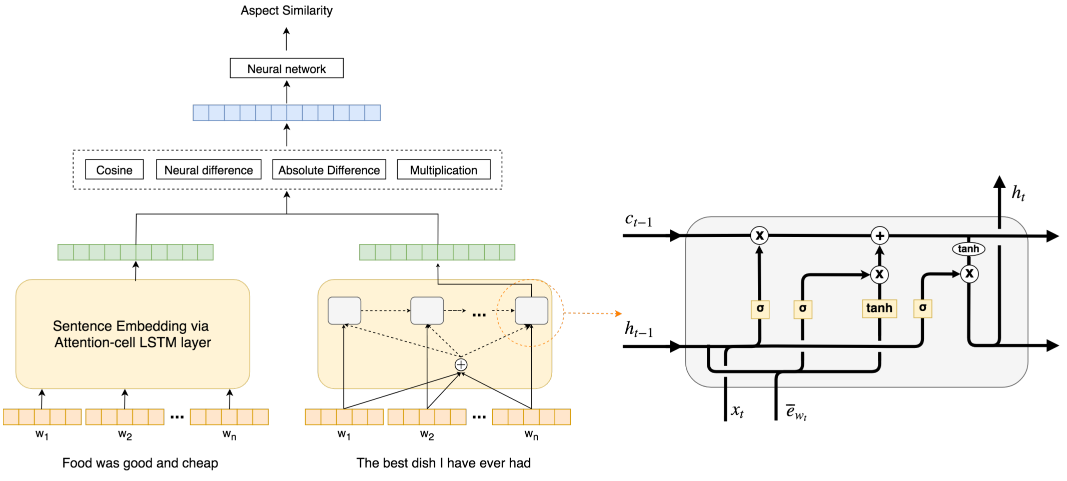
In the scope of this paper, we focus on discussing neural-based systems for generic and opinions summarization. For a comprehensive literature of non-neural techniques, we refer the reader to Liu and Zhang (2012).

For extractive generic summarization, Cao et al. (2015) rank sentences in a parsing tree via a recursive neural network. However, the model requires handcrafted features as input. Cheng and Lapata (2016) propose an end-to-end model for extracting words and sentences. In this system, a document is encoded via convolutional and recurrent layers, then an attention architecture is employed to extract sentences and words. Follow this work, Zhou et al. (2018) enhance the previous system by jointly learning to score and select sentences. By integrating sentence scoring and selecting into one phase, as the model selects a sentence, the sentence is scored according to the partial output summary and current extraction state.

To our knowledge, the first neural-based model of extractive opinions summarization is proposed by Kågebäck et al. (2014), which uses an unfolding recursive auto-encoder to learn phrase embeddings and measures similarity by Cosine and Euclidean distance. The limitation of this system is to purely rely on semantic similarity without taking into account the aspect information. Yang et al. (2017) use the unsupervised neural attention-based aspect autoencoder (ABAE) (He et al., 2017b) for presenting each aspect in an aspect embedding space. Then, the representative sentence for each aspect is selected via its distance with the centroid of that aspect. For summarization, however, ABAE is not efficient compared to K-mean in the aspects which occur more frequently in the dataset. Angelidis and Lapata (2018) introduce seed words of each domain to the autoencoder ABAE. This weakly-supervised model which is trained under multi-task objective outperforms the unsupervised model for aspect extraction. Different from the previous work in aspect-based opinions summarization, we apply aspect similarity recognition (ASR) instead of aspect extraction. ASR facilitates the problem of domain adaptation in summarization.

3 Problem Formulation

Every product e contains a set of reviews $R_e = \{r_i^e, ..., r_n^e\}$ expressing users' opinions on that product. A review r_i^e is viewed as a sequence of sentences $(s_1, ..., s_m)$. For each product e, our goal is to select the most salient sentences in reviews R_e for producing a summary. The proposed



The framework of Aspect Similarity Recognition

An attention cell of LSTM

Figure 1: The proposed framework for the aspect similarity task

approach is divided into subtasks as follows:

1. **Sentiment prediction** determines the overall polarity $p_s \in [-1, +1]$ a sentence carries, where $-1, +1$ respectively indicate maximally negative and positive. According to Angelidis and Lapata (2018), highly positive or negative opinions are more likely to contain informative text than neutral ones. In our system, we use the ensemble sentiment classifier proposed by Huy Tien and Minh Le (2017), which achieves strong performances at sentence level.

2. **Semantic textual similarity** measures the semantic similarity q_{ij} of two sentences i and j, which plays an important role in identifying the most informative sentences as well as redundant ones. We use the state-of-the-art multi-level comparison model (Tien et al., 2018) for this task.

3. **Aspect similarity Recognition (ASR)** predicts a probability r_{ij} that two sentences i and j shares at least one aspect. This subtask facilitates the elimination of redundant text in summarization, especially for domain adaptation.

4. **Summarization Generation** employs the three signals above for ranking sentences. A concise and informative summary of a product e is generated by selecting the most salient sentences from reviews R_e.

Section 4 describes in details the attention-cell LSTM for the ASR task and Section 5 explains how to combine the polarity, semantic and aspect similarity to produce a summary.

4 Attention Cell LSTM

According to Nguyen et al. (2018), recurrent neural networks efficiently capture aspect relationships. For dealing with the remaining difficulties of this task, the authors analyze the necessary of an attention mechanism. For that reason, we aim to emphasize salient words as encoding sentences over LSTM. A straightforward approach is to learn attention signals by self-attention and then apply these signals into inputs before feeding them into LSTM. In other words, these attention signals are applied to all gates of a LSTM cell. However, we assume emphasized input makes the cell forget more information on the previous state (the forget gate's function) while this state stores the most salient information by the support of attention signals. This conflict causes the inefficiency of integrating attention signals with LSTM. Therefore, we propose a novel LSTM cell which prevents the state from forgetting too much salient information as employing attention signals for encoding sentences. For the ASR task, the proposed attention-cell LSTM outperforms the conventional LSTM with/out using attention in both of settings: in-domain and cross-domain.

By representing a word w_i by a pre-trained word embedding e_{w_i}, we construct a sentence S of n words as a sequence of n word embeddings $S = [e_{w_1}, e_{w_2}, ..., e_{w_n}]$. Contextual information is incorporated in the word embeddings over the bidirectional GRU (Bahdanau et al., 2014) and then the self-attention signal a_i of w_i is learned as follows (from Yang et al. (2016)):

$$\overleftarrow{h_i} = \overleftarrow{GRU}(e_{w_i}) \tag{1}$$

$$\overrightarrow{h_i} = \overrightarrow{GRU}(e_{w_i}) \tag{2}$$

$$h_i = \overleftarrow{h_i} \oplus \overrightarrow{h_i} \tag{3}$$

$$u_i = tanh(W_a h_i + b_a) \tag{4}$$

$$a_i = \frac{exp(u_i^T u_a)}{\sum_i exp(u_i^T u_a)} \tag{5}$$

$$\bar{e}_{w_i} = e_{w_1} a_i \tag{6}$$

where $\oplus$ is concatenation operator, w_a, b_a, u_a are respectively a weight matrix, a bias, and a context vector. These parameters are randomly initialized and optimized during training.

A sentence s is transformed to a fix-length vector e_s by recursively applying a LSTM cell to each word embedding e_{w_t} and the previous step h_{t-1}. At each time step t, the LSTM unit with l-memory dimension defines six vectors in $\mathbb{R}^l$: input gate i_t, forget gate f_t, output gate o_t, tanh layer u_t, memory cell c_t and hidden state h_t (Tai et al., 2015). We modify the conventional LSTM cell to employ attention signals without the conflict of remembering and forgetting as follows:

$$i_t = \sigma(W_i \bar{e}_{w_t} + U_i h_{t-1} + b_i) \tag{7}$$

$$f_t = \sigma(W_f e_{w_t} + U_f h_{t-1} + b_f) \tag{8}$$

$$o_t = \sigma(W_o e_{w_t} + U_o h_{t-1} + b_o) \tag{9}$$

$$u_t = \tanh(W_u \bar{e}_{w_t} + U_u h_{t-1} + b_u) \tag{10}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot u_t \tag{11}$$

$$h_t = o_t \odot \tanh(c_t) \tag{12}$$

$$e_s = h_n \tag{13}$$

where $\sigma, \odot$ respectively denote a logistic sigmoid function and element-wise multiplication; W_i, U_i, b_i are respectively two weights matrices and a bias vector for input gate i. The denotation is similar to forget gate f, output gate o, tanh layer u, memory cell c and hidden state h. In the attention-cell LSTM, we introduce the attention signal a_t to only the input gate i_t and the tanh layer u_t, which are in charge of deciding what new information is going to be stored in the cell state. This approach allows the LSTM cell to employ attention for remembering salient information and avoid the unexpected effect of attention on the forget gate.

We visualize how the attention-cell LSTM manipulates attention signals in Figure 1. The four metrics are used to evaluate the relationship between two sentences e_{s_1} and e_{s_2} as follows (from Nguyen et al. (2018)):

Cosine similarity:

$$d_{cosine} = \frac{e_{s_1} \cdot e_{s_1}}{\|e_{s_1}\| \, \|e_{s_2}\|} \tag{14}$$

Multiplication vector & Absolute difference:

$$d_{mul} = e_{s_1} \odot e_{s_2} \tag{15}$$

$$d_{abs} = |e_{s_1} - e_{s_2}| \tag{16}$$

where $\odot$ is element-wise multiplication.

Neural difference:

$$x = e_{s_1} \oplus e_{s_2} \tag{17}$$

$$d_{neu} = W^{neu} x + b^{neu} \tag{18}$$

where W^{neu} and b^{neu} are respectively a weight matrix and a bias parameter.

As a result, we have a sentence-sentence similarity vector d^{sent} as follows:

$$d^{sent} = d_{cosine} \oplus d_{mul} \oplus d_{abs} \oplus d_{neu} \tag{19}$$

The sentence-sentence similarity vector is transferred into an aspect similarity label $\hat{y}$ through a two layers neural network as follows:

$$sim^{sent} = \sigma(W^{sent} d^{sent} + b^{sent}) \tag{20}$$

$$s\bar{i}m^{sent} = dropout(sim^{sent}) \tag{21}$$

$$\hat{y} = \sigma(W^y s\bar{i}m^{sent} + b^y) \tag{22}$$

where W^{sent}, W^y, b^{sent}, and b^y are weight matrices and bias parameters, respectively.

Dropout layer (Srivastava et al., 2014) is applied to our model. Dropout prevents networks from overfitting via randomly dropping out each hidden unit with a probability p on each presentation of each training case. We train this model under the cross entropy loss function and AdaDelta as the stochastic gradient descent (SGD) update rule. Details of Adadelta method can be found in Zeiler (2012).

5 Opinion Summarization

Given a product e, we aim to rank a set of sentences $D = \{s_i\}$ from the reviews talking about the product e. The procedure of scoring and selecting sentences for constructing an opinion summary K of the product e is as follows:

1. In the first step $t = 0$, we score each sentence $s_i \in D$ and select the most salient sentence $\hat{s}^0$ for the summary K:

$$asp_{s_i}^{t=0} = \frac{1}{|D|} \sum_{j \in D} r_{ij} \tag{23}$$

$$sim_{s_i}^{t=0} = \frac{1}{|D|} \sum_{j \in D} q_{ij} \tag{24}$$

$$sal_{s_i}^{t=0} = (1 + \alpha |p_{s_i}|) * asp_{s_i}^{t=0} * sim_{s_i}^{t=0} \tag{25}$$

$$\hat{s}^0 = \arg\max_{s_i \in D} \{sal_{s_i}^{t=0}\} \tag{26}$$

$$K^{t=1} = K \cup \{\hat{s}^0\} \tag{27}$$

$$D^{t=1} = D \setminus \{\hat{s}^0\} \tag{28}$$

At the step $t = 0$, the salient sal_{s_i} is computed by the semantic similarity sim_{s_i}, the aspect coverage sim_{s_i} and the polarity p_{s_i}. Different from the previous works, we also take into account the aspect coverage in which a sentence carrying more aspects has a higher salient score. In addition, the polarity of a sentence contributes to its ranking by a coefficient $\alpha \in [0, 1]$.

2. In the next step t, the salient sentence $\hat{s}^t$ is selected as follows:

$$asp_{s_i}^{t} = \frac{1}{|D^t|} \sum_{j \in D^t} r_{ij} \tag{29}$$

$$sim_{s_i}^{t} = \frac{1}{|D^t|} \sum_{j \in D^t} q_{ij} \tag{30}$$

$$\bar{sal}_{s_i}^{t=0} = (1 + \alpha |p_{s_i}|) * asp_{s_i}^{t} * sim_{s_i}^{t} \tag{31}$$

To avoid the redundant information, we penalize each sentence s_i by the aspect similarity $acov_{s_i}^t$ and semantic similarity $scov_{s_i}^t$ of that sentence with the selected sentences, in which β is a coefficient:

$$acov_{s_i}^{t} = \frac{1}{|K^t|} \sum_{j \in K^t} r_{ij} \tag{32}$$

$$scov_{s_i}^{t} = \frac{1}{|K^t|} \sum_{j \in K^t} q_{ij} \tag{33}$$

$$sal_{s_i}^{t} = \bar{sal}_{s_i}^{t} - \beta * acov_{s_i}^{t} * scov_{s_i}^{t} \tag{34}$$

$$\hat{s}^t = \arg\max_{s_i \in D^t} \{sal_{s_i}^{t}\} \tag{35}$$

$$K^{t+1} = K^t \cup \{\hat{s}^t\} \tag{36}$$

$$D^{t+1} = D^t \setminus \{\hat{s}^t\} \tag{37}$$

3. We repeat step 2 until the number of selected sentences is reached or the most salient score at the current step t is lower than a threshold. To avoid missing topic words in a summary, in step 1 and 2, we only select sentences containing words belonging to the list of frequent words on that topic. According to our observation, the topic words are the most frequent.

6 Experiments & Results

6.1 Aspect Similarity Recognition

We evaluate the attention-cell LSTM on ASRcorpus (Nguyen et al., 2018), which contains sentences from the SemEval 2016 dataset with two domains: RESTAURANT and LAPTOP. Each sample is a pair of sentences annotated as aspect similarity ($label = 1$) or not aspect similarity ($label = 0$). Table 2 reports the statistic of ASRCorpus in details.

	RESTAURANT			LAPTOP		
	Train	Dev	Test	Train	Dev	Test
Sentences	1239	469	587	1657	382	573
Sentence pairs	458K	68K	98K	447K	26K	44K
Similarity	229K	34K	49K	223K	13K	22K
Not similarity	229K	34K	49K	223K	13K	22K
Vocabulary	3769			3649		

Table 2: Statistic of ASRCorpus

We compare our model to some strong baselines as well as the conventional recurrent networks using attention. We choose the optimal values of hyper-parameters in our model and baselines via a grid search on 30% of LAPTOP domain. Because the number of RESTAURANT's categories is smaller than LAPTOP's, the performance of RESTAURANT domain is better.

Table 3 reports the experimental results. By employing efficiently attention signals, the attention-cell LSTM outperforms the conventional recurrent models using attention. As we analysis in Section 4, applying attention to all gates of a LSTM cell causes the conflict of remembering and forgetting. This drawback makes the training of the LSTM-attention model inefficient. Consequently, the trained LSTM-attention model predicts the same label for all inputs.

We also evaluate how the models perform in cross-domain setting where the models are trained on one domain dataset and tested on the other.

These results also prove that these approaches are potential to cross-domain application. We observe that a set of salient words in each domain is different. Therefore, the support of attention signals in domain adaptation is not significant compared to the recurrent models without attention.

Method	RES	LAP	→RES	→LAP
Word Average	70.75	65.12	54.5	54.59
CNN	77.57	67.23	54.08	54.49
LSTM	79.4	70.21	59.1	57.59
BiLSTM	79.2	71.14	59.2	57.95
Attention	78.79	68	57.92	54.55
LSTM-attention	50	50	50	50
Attention-Cell LSTM	**80**	**72.73**	**59.77**	**58.1**
Attention-Cell BiLSTM	79.42	71.65	59.3	58

Table 3: The in-domain and cross-domain experimental results on the two domains: RESTAURANT and LAPTOP. ”→Y” denotes that models are tested on Y but trained on the other. Accuracy metric is used for evaluation. The results are statistically significant at $p < 0.05$ via the pairwise t-test.

To obtain deeper analysis, we inspect the attention-cell LSTM's performance on each class (e.g., “similarity” and “not similarity”) by precision, recall and F1 scores reported in Table 4. In both of the domains and settings, the model performs better on “not similarity” class than “similarity” class in terms of F1 score. According to the results in cross-domain setting, we could conclude that the models learn rules, patterns for identifying aspect similarity rather than remembering topic words and keywords in a particular domain.

Domain	*Class*	*Precision*	*Recall*	$F1$
RES	Not Similarity	0.76	0.88	0.81
	similarity	0.86	0.82	0.78
LAP	Not Similarity	0.68	0.87	0.76
	similarity	0.82	0.59	0.68
→RES	Not Similarity	0.58	0.68	0.63
	similarity	0.62	0.52	0.56
→LAP	Not Similarity	0.56	0.72	0.63
	similarity	0.61	0.44	0.51

Table 4: The attention-cell LSTM's performance on each class.

6.2 Opinion Summarization

The Opinosis dataset (Ganesan et al., 2010) includes user reviews of 51 different topics (e.g., hotel, car, product). Each topic includes between 50 and 575 sentences made by various authors and around 4 reference summaries created by human. The corpus is suited for opinion summarization as well as evaluating the ability of domain adaptation.

We use ROUGE to assess the agreement of generated summaries and gold summaries. Our experiments include ROUGE-1, ROUGE-2 and, ROUGE-SU4, which base on one-gram, bi-gram and skip-bigram co-occurrences respectively.

The model for each subtask in our summarization system is implemented as follows:

- Sentiment prediction: the ensemble classifier (Huy Tien and Minh Le, 2017) is trained on Stanford Sentiment Treebank (Socher et al., 2013) with the accuracy of 88.6%.

- Semantic textual similarity : the multi-level comparison model (Tien et al., 2018) is trained on STSbenchmark[1] with the accuracy of 82.45%.

- Aspect similarity recognition: the attention-cel LSTM is trained on the ASRcorpus of the both domains with the accuracy of 76.2%.

- Summary generation: we set $\alpha = 1.67$ and $\beta = 0.1$. The number of the most frequent words is three. These parameters are optimized over a set of 5 topics randomly selected from the Opinosis dataset. According to the analysis of (Ganesan et al., 2010), the size of a summary is two sentences.

For comparison, we use MEAD (Radev et al., 2000) and CW-Add$_{Euc}$ (Kågebäck et al., 2014) as baselines. MEAD is an extractive method based on cluster centroids which selects the salient sentences by a collection of the most important words. CW-Add$_{Euc}$ measures the Euclidean similarity between two sentences by their continuous vector space. In addition, we also report the contribution of using aspect and sentiment information in summarization. The results denoted OPT$_R$ and OPT$_F$ in Table 5 describe the upper bound score

[1] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

Method	ROUGE-1			ROUGE-2			ROUGE-SU4		
	Recall	Precision	F	Recall	Precision	F	Recall	Precision	F
OPT$_R$	57.86	21.96	30.28	22.96	12.31	15.33	29.5	13.53	17.7
OPT$_F$	45.93	48.84	46.57	20.42	19.94	19.49	23.17	26.5	23.7
MEAD	**49.32**	9.16	15.15	**10.58**	1.84	3.08	**23.16**	1.02	1.89
CW-Add$_{Euc}$	29.12	22.75	24.88	5.12	3.6	4.1	10.54	7.59	8.35
The proposed summarizer									
Semantic	28.24	28.63	27.62	7.34	7.19	7	10.69	10.94	10.4
Semantic + Aspect	29.2	**29.19**	**28.24**	7.45	**7.29**	**7.12**	11.25	**11.26**	**10.78**
Aspect + Polarity	27.77	27.86	26.92	7.24	7.09	6.93	10.42	10.55	10.04
Semantic + Aspect + Polarity	28.56	28.31	27.5	7.06	6.84	6.71	10.92	10.83	10.4

Table 5: Performance comparison between the proposed methods and baselines.

of recall and F-score respectively. As the reference summaries of Opinosis are generated in abstractive approach by humans, our generated summaries cannot fully match with the reference summaries. For example, the maximum recall which an extractive method could achieve in ROUGE-1 is 57.86%.

While MEAD selects long sentences (around 75 words) containing a lot of salient words to achieve a high score in recall but low in precision, our approach obtains a balance between these scores with quite shorter sentences (around 17 words).

Positive sentences
I purchased a 2007 Camry because of the looks of the redesigned model and because of the legendary Toyota quality and reliability.
The Concierge staff, exceptional and extremely helpful, right from suggestions on transportation excursion options to recommending an amazing restaurant.
When I checked in, I asked to be shown several rooms and the staff was happy to do so.
Negative sentences
My wife does say the vehicle is not as comfortable for long trips as other cars we've owned.
We had to go up a floor and into a service area to find ice.
The rude and poor service started from the concierge who was curt when I asked a question .

Table 6: Some sentences carrying the most polarity in the Opinosis dataset.

To analyze why sentiment signals cause negative impacts on the summarization generation, we inspect the most polarity sentences in the corpus. Some typical sentences are listed in Table 6. We observe that most of these sentences express individual experiences and too subjective to be selected for summarization. According to the Opinosis dataset, overstrong words (i.e., rude, extremely) and subjective words (i.e., my wife, I, we) are seldom present in a summary. These factors lead to an unexpected result of using polarity information in summarization although sentences carrying the most polarity are still informative.

$Domain$	$Class$	$Semantic + Aspect$
Tablet	More informative	33%
	Less informative	13%
	Equally informative	54%
Others	More informative	17%
	Less informative	8%
	Equally informative	72%

Table 7: Informative test for using Semantic with Aspect against without Aspect.

We expect that aspect signals support to generate an informative summary, which is a summary carrying salient information on various aspects. However, the ROUGE metric measures the number of matches between two pieces of text, so it is difficult to compare which one is more informative. Therefore, we execute an **informative test** to understand whether aspect signals help to generate a more informative summary. Given reference summaries and two summaries generated by the system with/out using aspect signals respectively, three persons are asked to select one of the three answers: which system's summary is more informative, or both of them are equally informative. The inter-rater agreement Cohen's Kappa score for each pair of assessors is higher than 0.74. The overall answer is concluded by the majority vote scheme. In case of receiving three different answers, that pair of summaries is assigned as equally informative. The result reported in Table 7 includes domain specification (15 samples in $Tablet$ and 36 samples in $Others$), which facilitates the evaluation of domain adaptation. As the ASR system is trained on the restaurant and laptop

Summary on the Comfort of Toyota Camry 2007	
Human	[1] The Camry offers interior comfort, while providing a quiet ride. Comfortable seating and easy to drive. [2] Overall very comfortable ride front and back. Nice and roomy. [3] Its very comfortable and a quiet ride with low levels of noise.
Semantic	The ride is quiet and comfortable. Very comfortable, quiet interior.
Semantic + Aspect	The ride is quiet and comfortable. Very comfortable ride and seating.
Summary on the location of Holiday Inn London	
Human	[1] Location is excellent, very close to the Glouchester Rd. Tube stop. [2] Excellent location. Near the tube station. [3] The location is excellent. The hotel is very convenient to shopping, sightseeing, and restaurants. It is located just minutes from the tube stations.
Semantic	Great location but don't bring the car! Great location great breakfast!
Semantic + Aspect	Great location but don't bring the car! Great location for the tube and bus!

Table 8: Human and system summaries for some products/services. For each topic, we list three summaries by human.

dataset, we consider tablet's topics in the Opinosis corpus as in-domain and others as out-of-domain. According to the informative test, the system with aspect dominates in both of the domains ($Tablet$ and $Others$). This result proves the contribution of aspect signals and the domain adaptation of the ASR system.

To obtain a better view of the advantages and disadvantages in our system, we show some generated summaries against reference summaries in Table 8. In extractive methods, the most salient sentences are selected from different reviewers, so it is possible to have repeated information in a summary. For instance in the case #1, the first sentence mentions *quiet* and *comfortable ride* while the second one contains *ride* and *seating*. Although these sentences still have different opinions (i.e., *quiet* vs *seating*), the repeat of *comfortable ride* downgrades the generated summary's quality. For improvement, we suggest a post-processing for a more concise summary by filtering redundant information. As the proposed aspect-based system ranks a sentence by not only semantic cover but also aspect cover, it selects the more salient opinions for summarization. For instance, although both of the systems extract different features (e.g., interior vs seating, breakfast vs tube and bus), the opinions (i.e., seating, tube and bus) chosen by the system with aspect support are more suited to the reference summaries.

In each topic, although the reference summaries and generated summary share most of the meaning, they deliver information in different ways and words. This fact makes the quality evaluation of generated summaries difficult. In addition to the ROUGE metric, we conducted the informative test for quality evaluation. However, for a large corpus or multiple systems comparison, this test requires a huge amount of human effort. Therefore, it is a high demand to have a reliable metric for summaries evaluation without human involvement.

7 Conclusion

In this work, we introduced a novel aspect-based opinions summarization framework using aspect similarity recognition. This subtask relaxes the constraint of predefined aspects in conventional aspect categorization tasks. For ASR tasks, we proposed an attention-cell LSTM to integrate efficiently attention signals into LSTM. This approach outperforms the baselines on both settings of in-domain and cross-domain. For summarization, we evaluated our system on the Opinosis corpus. In addition to ROUGE metric, an informative test with human involvement was implemented to show the domain adaptation ability of our system and how informative our generated summaries are. In the corpus, we observe that sentences carrying the most polarity are not suited to summarization. Therefore, employing sentiment for summarization needs deeper analysis. Due to the ASR task's advantage, we believe that it has a high demand in some fundamental tasks of natural language processing such as information retrieval, and sentence comparison.

Acknowledgements

This work was supported by JST CREST Grant, Japan. We are also immensely grateful to the reviewers for their comments.

References

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.* 3:993–1022.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'15, pages 2153–2159.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 484–494.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 340–348.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017a. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017b. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '04, pages 168–177. https://doi.org/10.1145/1014052.1014073.

Minqing Hu and Bing Liu. 2006. Opinion extraction and summarization on the web. In *Proceedings of the National Conference on Artificial Intelligence*. volume 2.

Nguyen Huy Tien and Nguyen Minh Le. 2017. An ensemble method with sentiment features and clustering support. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, pages 644–653. http://aclweb.org/anthology/I17-1065.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*. Association for Computational Linguistics, pages 31–39.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5(1):1–167.

Bing Liu and Lei Zhang. 2012. *A Survey of Opinion Mining and Sentiment Analysis*, Springer US, Boston, MA, pages 415–463.

Samaneh Moghaddam and Martin Ester. 2013. The flda model for aspect-based opinion mining: Addressing the cold start problem. In *Proceedings of the 22Nd International Conference on World Wide Web*. ACM, New York, NY, USA, WWW '13, pages 909–918. https://doi.org/10.1145/2488388.2488467.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: Exploring the factors that influence summarization. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, SIGIR '06, pages 573–580. https://doi.org/10.1145/1148170.1148269.

Huy Tien Nguyen, Quan-Hoang Vo, and Minh-Le Nguyen. 2018. A deep learning study of aspect similarity recognition. In *Proceedings of the 10th International Conference on Knowledge and Systems Engineering*.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. Mead - a platform for multidocument multilingual text summarization. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*. European Language Resources Association (ELRA).

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, volume 1631, page 1642.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1556–1566.

Huy Nguyen Tien, Minh Nguyen Le, Yamasaki Tomohiro, and Izuha Tatsuya. 2018. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *CoRR* abs/1805.07882. http://arxiv.org/abs/1805.07882.

Yinfei Yang, Cen Chen, Minghui Qiu, and Forrest Bao. 2017. Aspect extraction from product reviews using category hierarchy information. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 675–680.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 1480–1489. https://doi.org/10.18653/v1/N16-1174.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 654–663.

Discourse-Aware Hierarchical Attention Network for Extractive Single-Document Summarization

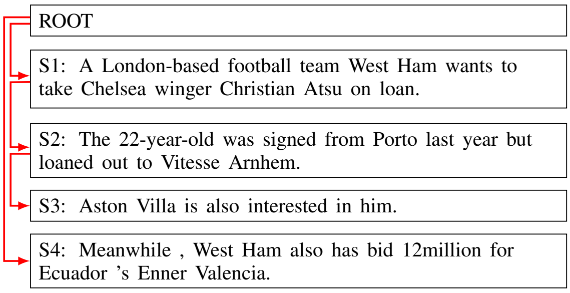
Tatsuya Ishigaki[⋆] **Hidetaka Kamigaito**[⋆] **Hiroya Takamura**[⋆⋄] **Manabu Okumura**[⋆]
[⋆]Tokyo Institute of Technology
[⋄]National Institute of Advanced Industrial Science and Technology
{ishigaki, kamigaito}@lr.pi.titech.ac.jp, {takamura, oku}@pi.titech.ac.jp

Abstract

Discourse relations between sentences are often represented as a tree, and the tree structure provides important information for summarizers to create a short and coherent summary. However, current neural network-based summarizers treat the source document as just a sequence of sentences and ignore the tree-like discourse structure inherent in the document. To incorporate the information of a discourse tree structure into the neural network-based summarizers, we propose a discourse-aware neural extractive summarizer which can explicitly take into account the discourse dependency tree structure of the source document. Our discourse-aware summarizer can jointly learn the discourse structure and the salience score of a sentence by using novel hierarchical attention modules, which can be trained on automatically parsed discourse dependency trees. Experimental results showed that our model achieved competitive or better performances against state-of-the-art models in terms of ROUGE scores on the DailyMail dataset. We further conducted manual evaluations. The results showed that our approach also gained the coherence of the output summaries.

1 Introduction

Document summarization is the task of automatically shortening a source document while retaining its salient information. In this paper, we present a recurrent neural network (RNN)-based extractive summarizer taking into account the discourse structure inherent in the source document.



ROOT

S1: A London-based football team West Ham wants to take Chelsea winger Christian Atsu on loan.

S2: The 22-year-old was signed from Porto last year but loaned out to Vitesse Arnhem.

S3: Aston Villa is also interested in him.

S4: Meanwhile , West Ham also has bid 12million for Ecuador 's Enner Valencia.

Figure 1: Example of discourse dependency structure.

The discourse structure consists of discourse relations between units in the input, and discourse information has been shown useful for summarization tasks. An example of a Rhetorical Structure Theory (RST) (Mann and Thompson, 1988)-based discourse structure, expressed as a dependency tree, is illustrated in Figure 1. In the figure, each node corresponds to a sentence. Regarding the relations between the sentences, sentence $S2$ elaborates the fact mentioned in sentence $S1$. In addition, $S2$ is further elaborated by $S3$. $S4$ is a contrast to the mention $S1$. Such relations are essential cues for generating a concise and coherent summary. For example, elaborated sentences tend to be more important than elaborating sentences, and the elaborated sentences should be included in the summary while the elaborating sentences are not.

Several Integer Linear Programming (ILP)-based summarizers (Hirao et al., 2013; Kikuchi et al., 2014) use the discourse information given by a discourse parser (Hernault et al., 2010). Thus, the performance of the summarizers is strongly affected by the performance of the discourse parsers. The performance of the parsers deteriorates especially when they are applied to documents of a domain different from the one which they were trained on.

Proceedings of Recent Advances in Natural Language Processing, pages 497–506,
Varna, Bulgaria, Sep 2–4 2019.

RNN-based approaches have achieved the state-of-the-art performance in document summarization (Cheng and Lapata, 2016; Nallapati et al., 2017). However, RNN-based summarizers treat the source document just as a sequence of sentences, and ignore the discourse tree structure inherent in the document. The lack of such information limits the ability to correctly compute relative importance between sentences and reduces the coherence of output summaries. Cohan et al. (2018) might be the only exception to the above, showing that the effectiveness of incorporating discourse information into an RNN-based summarizer for scientific papers by treating the source document as a sequence of sections such as "Introduction" or "Conclusion". However, they were not able to show how the tree-like discourse structure is effective in RNN-based approaches for extractive single-document summarization.

To effectively avoid the influence of parse errors and take advantage of the recent advances in neural network-based approaches, we propose a model that jointly learns the discourse tree structure of the source document and a scoring function for sentence extraction. Our model represents the discourse tree structure as an attention distribution and the probability of including a sentence in a summary as the softmax layer. In addition, recursive attention modules in our model can consider multi-hop dependencies between sentences. Therefore, our model can capture the relationships between sentences effectively and create a summary without losing the coherence between sentences.

We used an existing RST parser (Hernault et al., 2010) to add discourse dependency structure annotations to the DailyMail dataset (Hermann et al., 2015) and thereby obtained a large-scale annotated dataset to train the model. One of the advantages of our model is that we do not need the RST annotations in the inference phase because the model automatically infers the latent discourse tree structure of the source document and outputs the probability for each sentence as a salience score.

We empirically compared our model with other models. The results showed that discourse information improves the performance, and also that our models perform competitively with or better than state-of-the-art neural network-based extractive summarizers.

2 Related Work

There have long been many attempts at tackling extractive single-document summarization (Luhn, 1958), but there is still room for improvements in terms of ROUGE scores (Hirao et al., 2017). The recent focus has been on RNN-based approaches (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018). We further extend the attention mechanism used in RNN-based summarizers to capture a discourse structure.

RNN-based approaches were introduced to natural language processing tasks by the pioneering work by Bahdanau et al. (2015) and Luong et al. (2015), originally for machine translation. Rush et al. (2015) applied the approach to a sentence compression task. Nallapati et al. (2016) extended the model to abstractive document summarization. The DailyMail dataset (Hermann et al., 2015) has been commonly used for training abstractive summarizers. Cheng and Lapata (2016) and Nallapati et al. (2017) later proposed the methods to automatically annotate the binary labels, enabling us to train extractive models. Cohan et al. (2018) demonstrated the usefulness of incorporating discourse information into RNN-based summarizers. Unlike their model, our attention module explicitly captures the hierarchical tree structure inherent in the document.

Nallapati et al. (2016) and Yang et al. (2016) also used a hierarchical attention that consists of two simple attention modules; one is for words and the other is for sentences. Our attention mechanism differs from them in that ours captures discourse tree structures by new hierarchical attention networks, inspired by the models for capturing sentence-level dependency structures, e.g. machine translation (Hashimoto and Tsuruoka, 2017), dependency parsing (Zhang et al., 2017), constituency parsing (Kamigaito et al., 2017) and sentence compression (Kamigaito et al., 2018). Note that these models were designed for sentence-level tasks while we focus on the document-level summarization task.

Sentence selection modules that consider discourse structures of documents have been shown to be useful in ILP-based summarizers. Hirao et al. (2013) attempted to incorporate discourse information in ILP-based sentence extractors. Kikuchi et al. (2014) later proposed another ILP model that takes into account the discourse structure. Their model jointly selects and compresses sentences in an ILP summarizer. Unlike the researches above, our focus is on incorporating discourse information into RNN-based summarizers.

Penn Discourse TreeBank (PDTB) (Prasad et al., 2008) and RST are the most commonly used framework to represent a discourse structure. PDTB focuses on the relation between two sentences, and the annotated structure for a document is not necessarily a tree. In contrast, RST is forced to represent a document as a tree. Discourse parsers for both schema are available (Hernault et al., 2010; Feng and Hirst, 2014; Wang and Lan, 2015). There are at least two methods to convert an RST-based tree structure to a dependency structure (Hirao et al., 2002; Li et al., 2014). Hayashi et al. (2016) compared these methods and mentioned that DEP-DT by Hirao et al. (2002) has an advantage for applying to summarization tasks. We use DEP-DT for this research since we focus on integrating the tree structure into a summarizer.

We found only one model that jointly learns RST parsing and document summarization (Goyal and Eisenstein, 2016). They used the SampleRank algorithm (Wick et al., 2011), a stochastic structure prediction model, while our main focus is to take into account discourse structures in RNN-based summarizers.

3 Problem Formulation

We formulate extractive document summarization as a sentence tagging problem. We first briefly explain the notation in the paper and describe the details of the model in the following sections.

The source document $\boldsymbol{x}$ is represented as a sequence of sentences $x_1, ..., x_N$. Each sentence x_i is composed of a sequence of words $w_{i,j}$ $(1 \leq j \leq M_i)$, where M_i is the number of words in x_i. We also consider x_0 as a dummy root node. The summarizer outputs a sequence of binary decisions $\boldsymbol{y} = y_1, ..., y_N$, where $y_i = 1$ for the i-th sentence x_i to be included in the summary and $y_i = 0$ for the sentence not to be included. The binary decisions $\boldsymbol{y}$ are made by using a neural network-based probability distribution function $p(y_i|\boldsymbol{x}, \theta)$, where θ is the set of learned parameters. The model finds the best decisions $\boldsymbol{y}$ by a simple greedy search to maximize the sum of the probabilities within the length constraint.

Thus, our goal is to construct a better function $p(y_i|\boldsymbol{x}, \theta)$ given training data D. Each instance in D is a triple $(\boldsymbol{x}, \boldsymbol{E}, \boldsymbol{y})$, where $\boldsymbol{E}$ is a matrix to represent the discourse dependency tree of $\boldsymbol{x}$. Specifically, element $E_{k,l}$ equals 1 if the edge from x_k to x_l exists in the discourse tree; otherwise $E_{k,l} = 0$.

Note that we use the discourse structure matrices $\boldsymbol{E}$ only in the training phase. The model does not require the RST annotations of the source document when calculating the probability distribution $p(y_i|\boldsymbol{x}, \theta)$.

4 RNN-Based Extractive Summarizer

In this section, we first explain the base model and give the details of our proposed attention module in the following section. The base model is composed of two main components: a neural network-based hierarchical document encoder and a decoder-based sentence scorer. The document encoder is further split into two components; a sentence reader and a document reader. The hierarchical architecture is commonly used in recent neural network-based models (Cheng and Lapata, 2016; Nallapati et al., 2017; Cohan et al., 2018).

4.1 Word Reader

The goal of the Word reader is to convert sentence x_i to a sentence embedding h_i. For each word $w_{i,j}$ in a sentence x_i, the word reader first convert every word embedding $emb(w_{i,j})$ to hidden states $\overrightarrow{e}_{i,j} = LSTM(\overrightarrow{e}_{i,j-1}, emb(w_{i,j}))$ and $\overleftarrow{e}_{i,j} = LSTM(\overleftarrow{e}_{i,j+1}, emb(w_{i,j}))$ by using bi-directional Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997). Then, $\overleftarrow{e}_{i,j}$ and $\overrightarrow{e}_{i,j}$ are concatenated into a hidden state $h_{i,j} = [\overrightarrow{e}_{i,j}; \overleftarrow{e}_{i,j}]$, where $[\,]$ represents a concatenation operation of a vector. After that, all $h_{i,j}$ in the sentence x_i are averaged and represented as a sentence embedding h_i.

4.2 Sentence Reader

Once we obtain sentence embeddings h_i for each sentence x_i, the Sentence reader then reads sentence embeddings h_i by another bi-directional LSTM and generates context-aware sentence representation H_i for each $\boldsymbol{x_i}$. Specifically, two vectors generated by the forward recurrent neural network $\overrightarrow{H}_i = LSTM(\overrightarrow{H}_{i-1}, h_i)$ and the backward $\overleftarrow{H}_i = LSTM(\overleftarrow{H}_{i+1}, h_i)$ are concatenated into sentence representation H_i for x_i:

$$H_i = [\overrightarrow{H_i}; \overleftarrow{H_i}]. \tag{1}$$

We now obtain the context-aware sentence representations $\boldsymbol{H} = \{H_1, ..., H_N\}$. Finally, all H_i are averaged to make a document embedding K.

4.3 Decoder-Based Sentence Scorer

This module outputs the probability of including x_i in the summary, $p(y_i = 1|\boldsymbol{x}, \theta)$, by using an LSTM-based decoder. At each time step t ($1 \leq t \leq N$), the previous state of the decoder s_{t-1} and the sentence representation h_t are fed into the LSTM, and the LSTM outputs a new state; $s_t = LSTM(s_{t-1}, h_t)$. The initial state s_0 is initialized by the last states of the backward LSTM in the document reader $\overleftarrow{H_0}$.

Extractive document summarization often adopts a "hard attention", which focuses only on the encoder hidden state H_t in the decoding time step t (Cheng and Lapata, 2016). In addition, document representation K is also important to decode summaries (Nallapati et al., 2017). Based on them, the output layer calculates the probability distribution of x_t being included in the summary as:

$$p(y_t|\mathbf{x}, \theta) = softmax(\mathbf{W}_o tanh(\mathbf{W}_c[H_t; s_t; K])). \quad (2)$$

5 Discourse-Aware Hierarchical Attention Network

We assume that taking into account the discourse dependency structure is also useful in determining whether the summary includes a target sentence or not. Here, we make the model capable of accounting for the information of the parent sentences on the discourse dependency structure by incorporating our proposed hierarchical attention mechanism into the RNN-based extractive summarizer.

As shown in Figure 2, the goal of our attention mechanism is to generate an attention vector Ω_i containing the information from the parent sentences of x_i through the three-step attention modules. Below, we first give an overview of each step in the procedure and then formulate the components after that.

Step1: Parent Attention Module This module calculates the probability of x_k being the parent of x_i for all combinations of k and i where $k \neq i$. We denote this probability as $p(k|i, \boldsymbol{H})$. In the figure, the starting point of an edge is the parent, and the end point is the child. The probability $p(k|i, \boldsymbol{H})$ is used as the weight for the edge from H_k to H_i. The edge weights are passed to the Recursive Attention Module.

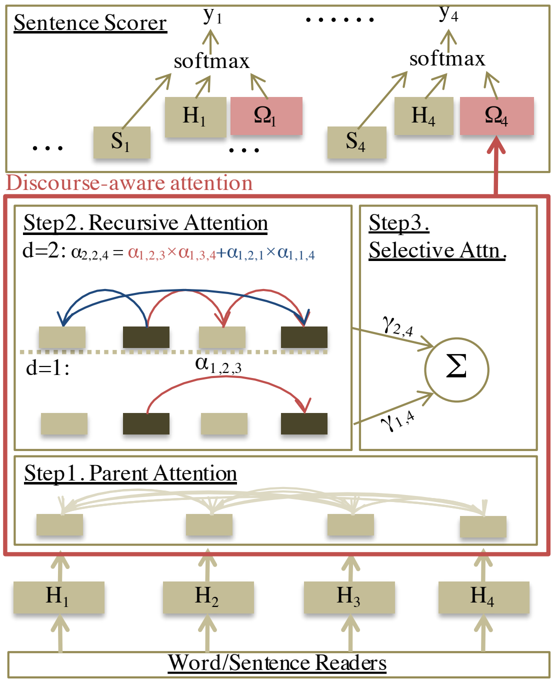


Figure 2: Overview of hierarchical attention mechanism for generating attention vector Ω_4. `Parent Attention` first calculates how likely the sentence x_i is the parent of x_j for all combinations. `Recursive Attention` then generates weighted sum vectors $\gamma_{d,4}$ over encoder hidden states H_i considering how likely the sentence x_i is the d-th order parent of x_4. `Selective Attention` finally generates another weighted sum vector Ω_4 over $\gamma_{d,4}$.

Step2: Recursive Attention Module This module outputs the weighted sum vectors $\gamma_{d,i}$ over $\boldsymbol{H}$, taking into account the d-th-order parents of x_i[1].

The module starts the calculation with the setting $d = 1$. Correspondingly, the module only considers the 1st-order parents of x_i. The Parent Attention Module has already calculated the probability of x_k being the parents of x_i. Thus, the Recursive Attention Module simply uses the probabilities as the weights $\alpha_{1,k,i}$ for every H_k and outputs the weighted sum vector $\gamma_{1,i}$.

When $d = 2$, the weights $\alpha_{2,k,i}$ for every H_k are calculated on the basis of how likely x_k becomes the 2nd-order parent of x_i. Here, d-th-order refers to the distance between x_k and x_i. For example, suppose there are two different paths connecting two nodes, and that their distances are both 2, illustrated by the path colored blue and red in Figure 2. The module multiplies the weights of the edges on each path, $\alpha_{1,2,3} \times \alpha_{1,3,4}$ for the red path and $\alpha_{1,2,1} \times \alpha_{1,1,4}$ for the blue path, and then the module sums the multiplied values;

[1] We use the plural form "parents" here because how likely a sentence becomes the parent of x_i is represented as a probability distribution in our model and multiple parents can be considered.

$\alpha_{2,2,4} = \alpha_{1,2,3} \times \alpha_{1,3,4} + \alpha_{1,2,1} \times \alpha_{1,1,4}$. We consider $\alpha_{2,2,4}$ to be the probability of x_2 being the 2nd-order parent of x_4. Then, the module uses the value as the weight and outputs the weighted sum vector $\gamma_{2,4}$.

When $d > 2$, the module recursively calculates the weight $\alpha_{d,k,i}$ by using the previously calculated weight $\alpha_{d-1,k,i}$ as shown in the next section.

Step3: Selective Attention Module Once weighted sum vectors $\gamma_{d,i}$ have been obtained taking into account the d-th-order parents of x_i, this module calculates the weights of each order d to select a suitable order. The module again calculates the weights for every order d and generates a weighted sum vector Ω_i.

5.1 Formulation of Attention Modules

Here, we describe the formulation of each attention module. The Parent Attention Module calculates the probability of x_k being the parents of x_i for all combinations of k and i where $k \neq i$:

$$\begin{aligned} p(k|i, \mathbf{H}) &= softmax(g(k,i)), \\ g(k,i) &= v_a^{\mathrm{T}} \tanh(U_a \cdot H_k + W_a H_i), \end{aligned} \tag{3}$$

where v_a, U_a and W_a are weight matrices.

The Recursive Attention Module recursively calculates the probability of x_k being the d-th-order parents of x_i:

$$\alpha_{d,k,i} = \begin{cases} p(k|i, \mathbf{H}) & (d = 1), \\ \sum_{l=0}^{N} \alpha_{d-1,k,l} \times \alpha_{1,l,i} & (d > 1). \end{cases} \tag{4}$$

Furthermore, in a discourse dependency tree, ROOT should not have any parent, and a sentence should not depend on itself. To satisfy these constraints, we impose the following on $\alpha_{1,k,i}$:

$$\alpha_{1,k,i} = \begin{cases} 1 & (k = 0, i = 0), \\ 0 & (k = i, i \neq 0). \end{cases} \tag{5}$$

The first equation constrains the ROOT node not to have any parent sentence. The second constraint ensures that a sentence does not depend on itself.

The calculated probabilities $\alpha_{d,k,i}$ are then used to weigh the vectors in $\mathbf{H}$, and the weighted sum vector $\gamma_{d,i}$ is generated as:

$$\gamma_{d,i} = \textstyle\sum_{k=0}^{N} \alpha_{d,k,i} H_k. \tag{6}$$

Once the weighted sum vector $\gamma_{d,i}$ is obtained for each order d, the Selective Attention Module calculates the weights $\beta_{d,i}$ for each $\gamma_{d,i}$ to find a suitable order:

$$\beta_{d,i} = softmax(\mathbf{W}_\beta [H_i; s_i; K]), \tag{7}$$

where $\mathbf{W}_\beta$ is a weight matrix. The attention vector is obtained as a weighted sum of $\gamma_{d,t}$:

$$\Omega_i = \textstyle\sum_d \beta_{d,i} \gamma_{d,i}. \tag{8}$$

Finally, the output layer receives the concatenated vector of H_i and Ω_i:

$$p(y_i|\mathbf{x}, \theta) = softmax(\mathbf{W}_o tanh(\mathbf{W}_{c'}[H_i; s_t; K; \Omega_i])). \tag{9}$$

5.2 Objective

The training updates the parameters to maximize both the label probability and 1st-order attention distribution $\alpha_{1,k,l}$. Specifically, we use the following loss function for optimization:

$$-\log p(\mathbf{y}|\mathbf{x}) - \lambda \cdot \textstyle\sum_{k=1}^{N} \sum_{i=1}^{N} E_{k,i} \log \alpha_{1,k,i}. \tag{10}$$

In this equation, $E_{k,i}$ is 1 if the edge from x_k to x_i exists in the training instance. Thus, all the parameters are updated to reproduce the correct labels and edges appearing in the training data D. λ is a parameter to control the priority of the output labels or the edges given by an RST parser.

6 Experiments

Data and Preprocessing: We used two different datasets for the experiments; the DailyMail dataset for training and evaluation, and the DUC2002 test set[2] only for evaluation.

The DailyMail dataset (Hermann et al., 2015) consists of news articles extracted from Daily Mail Online[3] and their "story highlights" created by human writers. Nallapati et al. (2016) regarded the highlights as human-generated abstractive summaries. For training extractive summarization models, we need to annotate sentences with binary labels for sentence extraction. To do this, Cheng and Lapata (2016) used a rule-based approach considering the similarity between the original document and extracted sentences. On the other hand, Nallapati et al. (2017) proposed a simple heuristic for labeling sentences to be included in the summary by maximizing the ROUGE scores, using the highlights as reference summaries. We used the latter scheme to annotate the binary labels for sentence extraction.

As a preprocessing, we applied the HILDA parser (Hernault et al., 2010) to annotate RST-based discourse information for all the documents. The RST trees were then converted into dependency structures by using the method described in Hirao et al. (2013). The parser requires the features extracted from word surfaces and the information on paragraph boundaries. However,

[2] https://www-nlpir.nist.gov/projects/duc/guidelines/2002.html

[3] http://www.dailymail.co.uk/

the preprocessed DailyMail dataset[4] provided by Cheng and Lapata (2016) and commonly used for summarization tasks was not suitable for our use. The dataset is anonymized; all named entities were replaced by the special token `@entity`, and paragraph boundaries were deleted. Therefore, we used the non-anonymized version of the dataset provided by Hermann et al. (2015). We obtained 196,557 training documents, 12,147 validation documents and 10,396 test documents from the DailyMail dataset.

The DUC2002 test set consists of 116 pairs of source documents and their extractive summaries, and 567 pairs of source documents and their abstractive summaries. We used the dataset for evaluation on out-of-domain data.

Compared Models: We compared our models with various baseline models. `DIS w/ PAR` is our model with the model parameter $\lambda > 0$. With this setting, all the parameters are tuned to reproduce the correct labels and the edges given by the RST parser. `DIS fixed` is the discourse-aware model with the attention vector $\Omega_t = H_{t-1}$. Thus, this model always treats the preceding sentence as the parent. `DIS w/o PAR` is the model with the model parameter $\lambda = 0$. Note that the objective function in this model does not take into account the RST annotations given by the RST parser. Thus, all the discourse structures are learned to reproduce the correct sentence labels without the information from the parser.

We compared the above models with the model without any discourse-aware attention mechanisms (`no-attn`) to verify the effectiveness of our attention mechanisms. `Lead-3` is a common baseline to select the first three sentences. `SummaRuNNer` is a well-known RNN-based summarizer by Nallapati et al. (2017). This model uses some types of information that we do not use, such as the similarity between the source document and the target sentence, and the novelty score of the target sentence, while our approach incorporates the information on the parent sentence of the target sentence. `NeuralSum` is also a neural network-based summarizer which uses convolutional neural networks in the encoder. `Refresh` is a state-of-the-art method using reinforcement learning (Narayan et al., 2018) [5].

In addition to the above methods, we compared our models with previously reported performances on the DUC2002 test set. `LREG` is a feature-rich logistic regression based approach used as a baseline in Cheng and Lapata (2016). `ILP` is a phrase-based extraction system proposed by Woodsend and Lapata (2010). The approach extracts the phrases and recombines them subject to the constraints in the ILP such as length, coverage or grammaticality. Both `TGRAPH` (Parveen et al., 2015) and `URANK` (Wan, 2010) are graph-based sentence extraction approaches, that perform well on the DUC2002 corpus.

Evaluation Metrics: We conducted both automatic evaluation and human evaluation. In automatic evaluation, we adopted ROUGE scores (Lin, 2004). We specifically calculated ROUGE-1, ROUGE-2 and ROUGE-L by using the Pyrouge library[6]. The highlights in the Dailymail dataset were treated as reference summaries when we calculated the scores. We used three length constraints; 75 bytes, 275 bytes (Nallapati et al., 2017; Cheng and Lapata, 2016) and the bytes of reference summaries. We truncated generated summaries in the middle to conform to the length constraints. We adopted the last constraint to evaluate whether a model can include sufficient information within the ideal summary length. For the evaluation on out-of-domain data, we report the ROUGE scores on the DUC2002 abstractive and extractive test sets. Our models are trained on the DailyMail dataset and tested on DUC2002.

We additionally carried out human evaluation because ROUGE scores cannot capture the coherence, though our attention modules are designed to improve the coherence of summaries. We used Amazon Mechanical Turk to conduct human evaluation. Specifically, randomly selected 100 documents and their four summaries generated by `DIS w/ PAR`, `Lead-3`, `no-attn`, and `SummaRuNNer` were shown to the workers. Five workers were asked to rate each summary on a 1-5 scale in terms of coherence and informativeness. The instruction shown to the workers follows the DUC quality question[7].

Training Details: We used Adam (Kingma and Ba, 2015) for the optimizer, where the learning rate was set to 0.001. In accordance with the model parameters used in Nallapati et al. (2017),

[4] `http://homepages.inf.ed.ac.uk/mlap/index.php?page=resources`

[5] We used the implementation provided by the authors.

[6] The options for the Rouge script were "-a -c 95 -m -n 2 -b 75" and "-a -c 95 -m -n 2 -b 275".

[7] `https://duc.nist.gov/duc2007/quality-questions.txt`

we limited the vocabulary size of the input to 150,000 and replaced the out-of-vocabulary words with the token UNK. The size of the mini-batch was set to 8. We used the size of 100 for hidden layers in the LSTMs and 300 for word embeddings, which were initialized with pre-trained embeddings, word2vec-slim. Note that the previous researches (Nallapati et al., 2017; Cheng and Lapata, 2016) also used pre-trained embeddings. We filtered the training instances consisting of 50 or more sentences in the source document, following Nallapati et al. (2017). The parameter λ of all the discourse-aware models was tuned on the validation set. We tried the following values for λ: 0.01, 0.1, 1.0, 2.0, 5.0 and 10.0.

7 Results and Discussion

ROUGE scores on DailyMail dataset: Table 1 shows the ROUGE scores evaluated on the DailyMail test set. For fair comparison, we also re-trained the baseline models by using our non-anonymized dataset.

DIS w/o PAR achieved better ROUGE scores than no-attn in all variations of length constraint except for ROUGE-L score on the setting with $d = \{1, 2\}$. Furthermore, we obtained better scores for DIS fixed than DIS w/o PAR. Incorporating the simple discourse information which treats the preceding sentence as the parent in the objective function improved the performance. Exploiting the discourse information given by the RST parser (DIS w/ PAR) further improved the scores in most settings. These observations suggest that discourse information is useful in RNN-based summarizers.

In the setting with the length constraint of 75 bytes, we observed a statistically significant difference between DIS w/ PAR and other neural network-based models (SummaRuNNer , Refresh and NeuralSum) on the settings with $d = \{1, 2\}$ and $d = \{1, 2, 3\}$. We also observed the similar tendency in the setting with the length constraint of reference summaries. Furthermore, we did not observe a statistically significant difference between DIS w/ PAR and SummaRuNNer in the setting with the length constraint of 275 bytes. Those facts would suggest that our models achieve a performance similar to the other baseline models in the setting with longer length constraints, and can perform better with shorter length constraints.

ROUGE scores on DUC2002 dataset: The results are shown in Table 2. Neural network-based approaches achieved similar scores on the abstractive test set because the length constraint is long; specifically it was set to 100-words. However, graph-based approaches (TGRAPH and URANK) performed better than the neural network-based approaches. As reported in Nallapati et al. (2017), neural network-based approaches suffer the difficulties in achieving high performance on out-of-domain data due to its high capability to fit in-domain data. Another possible reason might be the method for creating the binary labels for the training dataset. The binary decisions on the training dataset were made to maximize the ROUGE-F scores. Thus, the labels are strongly affected by the length of reference summaries in the DailyMail dataset. Since the average length of the reference summaries in the DUC test set is longer than the average length in the DailyMail dataset, the models trained on the DailyMail dataset might face difficulties. Our proposed models achieved the significantly better performances among the neural network-based approaches on the extractive test sets, which are for the settings with shorter length constraints (50 and 100 words) .

Human Evaluation: Table 3 shows the results. DIS w/ PAR were evaluated better than no-attn and SummaRuNNer in terms of coherence in the settings with all the different length constraints. These differences are statistically significant with the sign test ($p < 0.05$). Thus, human evaluation also supports the effectiveness of incorporating discourse information. Lead-3 is inherently strong in terms of coherence because this model is constrained to extract consecutive sentences while other models possibly extract non-consecutive ones. It was evaluated better in the setting with 75 bytes length constraint.

Analysis: Table 4 shows an example of the source document and outputs of two models; the sentences selected by our model are colored red and those selected by SummaRuNNer are blue, and those selected by both are purple. In this example, $S7$ elaborates $S6$. Our summarizer successfully extracted $S6$, that made the output summary more similar to the gold summary.

8 Conclusion

We presented a hierarchical attention network that captures the discourse dependency structure of the

	75			275			Ref.		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
DIS w/ PAR, d={1}	22.9	9.6	12.1	41.9	17.4	**35.2**	$^+$41.1	$^+$**16.9**	$^+$36.7
DIS w/ PAR, d={1,2}	$^+$**25.0**	$^+$11.1	$^+$13.4	41.7	17.4	35.0	41.0	16.7	$^+$36.7
DIS w/ PAR, d={1,2,3}$^{\#}$	$^+$24.6	$^+$**11.2**	$^+$**13.5**	41.8	17.2	**35.2**	$^+$41.1	$^+$16.8	$^+$36.8
DIS w/ PAR, d={1,2,3,4}	$^+$24.1	10.8	$^+$13.2	41.1	**17.5**	35.1	$^+$**41.2**	$^+$**16.9**	$^+$**37.0**
DIS fixed, d={1}	23.4	10.3	12.7	39.9	16.1	33.5	39.4	15.7	35.4
DIS fixed, d={1,2}	23.5	10.4	12.8	40.3	15.9	33.6	39.7	16.1	35.8
DIS fixed, d={1,2,3}	22.9	9.8	12.3	40.3	16.4	33.8	39.6	15.9	35.6
DIS fixed, d={1,2,3,4}	22.6	9.2	11.7	39.8	15.6	33.4	39.3	16.1	35.9
DIS w/o PAR, d={1}	21.2	8.1	11.0	40.1	15.8	33.7	39.6	15.5	35.5
DIS w/o PAR, d={1,2}	21.1	7.5	10.6	40.0	15.8	33.0	39.6	15.6	35.5
DIS w/o PAR, d={1,2,3}	20.9	7.9	10.9	40.5	16.1	34.1	40.0	15.8	35.8
DIS w/o PAR, d={1,2,3,4}	21.1	8.0	10.9	40.2	15.7	33.6	39.6	15.5	35.6
Lead-3	23.0	9.4	11.8	41.9	17.0	32.5	40.4	16.3	36.1
no-attn	20.1	7.1	10.4	39.6	15.4	33.3	39.3	15.3	35.2
SummaRuNNer (re-run)	23.2	9.6	11.0	**42.0**	17.2	32.5	37.6	14.8	33.7
Refresh (re-run)	23.1	10.9	12.6	37.9	16.5	31.4	36.6	15.8	34.1
NeuralSum (re-run)	22.4	9.1	11.8	40.8	16.3	34.8	40.3	15.9	36.1

Table 1: ROUGE Scores on DailyMail dataset. The models are **trained and tested on DailyMail dataset**. The length constraints are set to 75 bytes, 275 bytes and the reference length. The best scores among the models in bold. The symbol + indicates statistical significance using 95% confidence interval with respect to the nearest baseline, estimated by the ROUGE script. # indicates the model that achieved the best score in ROUGE-2 among the same methods with different **d** in the development dataset.

	Abstracts			Extracts (50 words)			Extracts (10 words)		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
DIS w/ PAR	44.7	21.8	38.2	**43.7**	**14.2**	**38.6**	**21.4**	**8.2**	**19.5**
DIS w/o PAR	46.1	23.3	**43.6**	42.1	13.5	36.8	19.0	7.5	17.3
no-attn	43.3	20.9	41.0	42.5	13.3	37.3	19.1	7.1	17.3
SummaRuNNer	46.6	23.1	43.0	42.1	13.4	36.7	20.9	7.8	19.0
Refresh	-	-	-	-	-	-	-	-	-
NeuralSum	-	-	-	-	-	-	-	-	-
LEAD-3	43.6	21.0	40.2	43.4	14.1	38.4	21.3	8.1	19.4
LREG	43.8	20.7	40.3	-	-	-	-	-	-
ILP	45.4	21.3	42.8	-	-	-	-	-	-
TGRAPH	48.1	**24.3**	-	-	-	-	-	-	-
URANK	**48.5**	21.5	-	-	-	-	-	-	-

Table 2: ROUGE scores on DUC2002 dataset. All neural network-based models are trained on DailyMail dataset and tested on DUC2002 test set.

	75		275		Ref	
	C	I	C	I	C	I
DIS w/ PAR	*3.86	3.57	***4.11**	***3.97**	***3.98**	3.78
SummaRuNNer	3.61	3.57	2.98	3.77	2.81	3.16
no-attn	3.73	3.52	3.92	3.86	3.80	3.70
LEAD-3	**3.98**	**3.69**	4.06	**3.97**	3.94	**3.80**

Table 3: Human evaluation on randomly selected 100 documents from DailyMail dataset. C and I stand for coherence and informativeness respectively. The mark * indicates that DIS w/ PAR achieved statistically significant difference, calculated by the sign test ($p < 0.05$), from both SummaRuNNer and no-attn.

Document:

S1: Bayern Munich is interested in Chelsea defender Branislav Ivanovic but are unlikely to make a move until Jan.

S2: The Serbia captain has yet to open talks over a new contract at Chelsea and his current deal runs out in 2016.

S3: Chelsea defender Branislav Ivanovic could be targeted by Bayern Munich in the January transfer window.

S4: Bayern like Ivanovic but don't expect Chelsea to sell yet they know he will be free to talk to foreign clubs from Jan.

S5: Paris Saint-germain will make a 7million offer for Chelsea goalkeeper Petr Cech this summer.

S6: The 32-year-old is poised to leave Stamford Bridge and wants to play for a champions league.

S7: Contender PSG are set to make a 7million bid for Ivanovic's Chelse a team-mate Petr Cech in the summer.

Gold Summary:

Branislav Ivanovic's contract at Chelsea expires at the end of next season.The 31-year-old has yet to open
talks over a new deal at Stanford bridge. Petrcech is poised to leave Chelsea at the end of the season

Table 4: Example of the extracted sentences. The sentences in bold are included in our summary. The sentences colored red were selected by our model, blue were by SummaRuNNer and purple were by both models.

source document. The experiments showed that incorporating discourse information into RNN-based extractive summarizers improves coherence and informativeness evaluated by human judges in addition to ROUGE scores. Our models outperformed or achieved competitive performances against the state-of-the-art methods. Improving the performance on out-of-domain data will be one of our future directions.

Acknowledgements

We thank to Tsutomu Hirao for helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR2015*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL2016*, pages 484–494, Berlin, Germany.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of NAACL2018*, pages 615–621.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of ACL2014*, pages 511–521.

Naman Goyal and Jacob Eisenstein. 2016. A joint model of rhetorical discourse structure and summarization. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 25–34.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. In *Proceedings of EMNLP2017*, pages 125–135.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for rst discourse trees. In *Proceedings of SIGDIAL2016*, pages 128–136.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS2015*, pages 1693–1701.

Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

Tsutomu Hirao, Hideki Isozaki, Eisaku Maeda, and Yuji Matsumoto. 2002. Extracting important sentences with support vector machines. In *Proceedings of COLING2002*, pages 1–7.

Tsutomu Hirao, Masaaki Nishino, and Masaaki Nagata. 2017. Oracle summaries of compressive summarization. In *Proceedings of ACL2017*, pages 275–280.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of EMNLP2013*, pages 1515–1520.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2018. Higher-order syntactic attention network for longer sentence compression. In *Proceedings of NAACL2018*, pages 1716–1726.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2017. Supervised attention for sequence-to-sequence constituency parsing. In *Proceedings of IJCNLP2018 (Volume 2)*, pages 7–12.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of ACL2014*, pages 315–320.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR2015*.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014. Text-level discourse dependency parsing. In *Proceedings of ACL2014*, pages 25–35.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of ACL2004 Workshop*, pages 74–81.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, (2):159–165.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP2015*, pages 1412–1421.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of AAAI2017*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of CoNLL2016*, pages 280–290.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of NAACL2018*, pages 1747–1759.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of EMNLP2015*, pages 1949–1954.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of LREC2008*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of EMNLP2015*, pages 379–389.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of COLING2010*, pages 1137–1145.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning-Shared Task*, pages 17–24.

Michael L Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. Samplerank: Training factor graphs with atomic gradients. In *Proceedings of ICML2011*, pages 777–784.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of ACL2010*, pages 565–574.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL2016*, pages 1480–1489.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of EACL2017*, volume 1, pages 665–676.

Semi-Supervised Induction of POS-Tag Lexicons with Tree Models

Maciej Janicki
University of Leipzig
NLP Group, Department of Computer Science
Augustusplatz 10, 04109 Leipzig
macjan@o2.pl

Abstract

We approach the problem of POS tagging of morphologically rich languages in a setting where only a small amount of labeled training data is available. We show that a bigram HMM tagger benefits from re-training on a larger untagged text using Baum-Welch estimation. Most importantly, this estimation can be significantly improved by pre-guessing tags for OOV words based on morphological criteria. We consider two models for this task: a character-based recurrent neural network, which guesses the tag from the string form of the word, and a recently proposed graph-based model of morphological transformations. In the latter, the unknown POS tags can be modeled as latent variables in a way very similar to Hidden Markov Tree models and an analogue of the Forward-Backward algorithm can be formulated, which enables us to compute expected values over unknown taggings. We evaluate both the quality of the induced tag lexicon and its impact on the HMM's tagging accuracy. In both tasks, the graph-based morphology model performs significantly better than the RNN predictor. This confirms the intuition that morphologically related words provide useful information about an unknown word's POS tag.

1 Introduction

Part-of-speech tagging is nowadays commonly thought of as a solved problem, with accuracy scores easily achieving 95% or more. However, such results are typically reported for English or other resource-rich European languages, for which large amounts of high-quality training data are available. Those languages also tend to have a simple morphology and utilize small to mid-sized tagsets. However, for many other languages, the reality is different: training data are expensive or not available, more fine-grained tagsets are needed and complex morphology accounts for large numbers of OOV words in corpora. (Straka and Straková, 2017) present a contemporary evaluation of state-of-the-art POS tagging for a very wide variety of languages. The scores for tagging with UPOS[1] tagset lie below 90% for many languages. The lack of sufficient amounts of training data is undoubtedly one of the main reasons for such results.

In this paper, we attempt to improve the POS tagging in a setting where only a small amount of labeled training data is available, as well as a significantly larger corpus of unlabeled text. We train a bigram Hidden Markov Model on the labeled part of the corpus and subsequently apply Baum-Welch estimation on the unlabeled part. Additionally, we use the labeled part to train a morphology model in an unsupervised setting. The key idea is to extend the tagger's lexicon with words occurring in the unlabeled part before the Baum-Welch estimation and to pre-guess their possible tags using the morphology model.

The choice of a bigram HMM for tagging is clearly suboptimal. However, our objective here is an initial proof-of-concept demonstrating that tagging in low-resource settings can benefit from unsupervised morphology. The choice of an HMM is dictated by the fact that it is easy to implement, well interpretable in its workings, possible to train on unlabeled data (using the Baum-Welch algorithm) and that it is possible to extend an exist-

[1] The tagset used in the Universal Dependencies project. It is very coarse-grained, containing e.g. only a single tag for nouns and verbs, respectively.

ing model with new vocabulary without complete re-training. Furthermore, the closely related trigram HMMs used to be state-of-the-art for a long time and are still used in popular tools like HunPos (Halácsy et al., 2007; Megyesi, 2009). Transferring this result to state-of-the-art tagging methods remains a topic for further work.

2 Related Work

2.1 Morphology-Based Induction of POS Lexicons

The idea of guessing possible tags for out-of-vocabulary words based on automatically induced morphological rules was proposed already by (Mikheev, 1997). He introduces a probabilistic model of string transformations, followed by a statistical significance analysis, which learns the correspondence between string patterns and POS tags. A related problem is learning morphological paradigms from inflection tables (Durrett and DeNero, 2013; Ahlberg et al., 2014). Here, complete paradigm tables of annotated word forms are used to learn string transformations between different forms, which are subsequently applied to unknown words to derive their possible tags.

Recently, (Faruqui et al., 2016) approached a task very similar to ours using a graph-based model. Without explicitly modeling morphology, they model structural similarities between words, like a regular affix change or sharing a common affix, as graph edges. They use a discriminative model of label propagation through edges which is similar to the Ising model of intermolecular forces. The induced lexicons are evaluated directly, as well as on how they improve a morphosyntactic tagger and a dependency parser.

2.2 Hidden Markov Tree Models

Hidden Markov Tree (HMT) models are a generalization of Hidden Markov Models, in which every node can have multiple descendents. They were introduced in the context of signal processing by (Crouse et al., 1998; Durand et al., 2004), together with an analogue of the well-known forward-backward algorithm. They remain relatively unknown in the field of language processing – the only mentions that we are aware of are (Žabokrtský and Popel, 2009) and (Kondo et al., 2013).

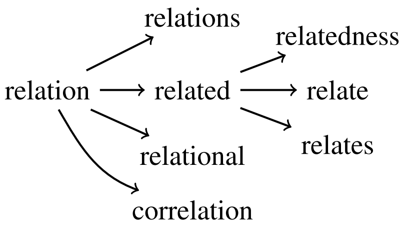


Figure 1: An example tree of word derivations. Edge labels are omitted for better readability. (reproduced from: (Sumalvico, 2017))

3 The Graph Model of Morphology

As a model of morphology suitable for unsupervised training, we use the graph-based model introduced by (Janicki, 2015), which is based on the Whole Word Morphology approach (Ford et al., 1997; Neuvel and Fulop, 2002). It expresses morphological relationships amongst words as a graph, in which words are vertices and morphologically related words are connected with an edge. A concrete morphological analysis of a set of related words is a tree, in which directed edges denote morphological derivation (Fig. 1). In general, the analysis of a vocabulary is a forest, i.e. a set of such trees.[2]

Every edge in the graph is an instance of a *morphological rule*, which describes a pattern applied to whole words. For example, the rule responsible for the pair (*relation*, *related*) might have the following form:

$$/X\text{ion}/ \rightarrow /X\text{ed}/ \qquad (1)$$

In this notation, X is a wildcard that can be instantiated with any string and a pattern between slashes refers to a whole word in its surface form. The rules may also include context: $/X\text{tion}/ \rightarrow /X\text{ted}/$ would be a possible alternative to (1). As there are multiple possible rules that can describe a relationship between a pair of words, the graph edges are defined as triplets of source word, target word and rule.

(Janicki, 2015) introduced a generative probabilistic model for trees such as the one depicted in Fig. 1. It assumes that the roots of the trees are

[2] It is important to point out that such trees are only an intermediary means in determining the strength of morphological relationship between string-similar words. The inference is always based on large samples of trees and finding the ‘right’ tree, i.e. the one that is consistent with linguistic analysis, is not the goal of the model.

generated independently from a distribution over arbitrary strings, called ρ. Furthermore, every rule r has a fixed probability θ_r of being applied to generate a new word. The probability of the whole forest is defined as follows:

$$Pr(V, E|R, \theta_R) \propto \prod_{v \in V_0} \rho(v) \\ \times \prod_{v \in V} \prod_{r \in R} \prod_{v' \in r(v)} \begin{cases} \theta_r & \text{if } \langle v, v', r \rangle \in E, \\ 1 - \theta_r & \text{if } \langle v, v', r \rangle \notin E \end{cases} \tag{2}$$

V denotes the set of vertices (words) and E the set of (labeled) edges of the graph. R denotes the set of rules and θ_R the vector of probabilities θ_r for the rules from R. Furthermore, $V_0 \subseteq V$ denotes the set of root nodes of the graph and $r(v)$ the set of words that can be derived from word v by application of rule r. Note that the latter might be an empty set if the context on the left-hand side of the rule is not matched. Each possible derivation from $r(v)$ corresponds to a Bernoulli variable with probability θ_r.

(Sumalvico, 2017) describes an unsupervised fitting procedure for this model using Monte Carlo Expectation Maximization algorithm. The computation involves drawing large samples of graphs from the conditional distribution $Pr(E|V, R, \theta_R)$ using a Metropolis-Hastings sampler.

4 Computing POS-Tags

We now turn to applying the model introduced in the previous section to the task of semi-supervised POS tag lexicon induction. The idea is to train a model of morphology on a labeled vocabulary (coming from a tagged corpus) and apply this model to infer the tags for another vocabulary.

The underlying intuition is that morphological relationships between words can give hints about their POS tags and inflectional forms, which are not visible in the isolated forms. For example, consider the following German[3] words: *Fichten* 'spruce.N.PL', *richten* 'judge.V.INF', *rechten* 'right.ADJ.NOM.PL.DEF'.[4] Phonetically and orthographically they are very similar and all include an inflectional suffix *-en*, which is highly ambiguous in German. The knowledge that German nouns are always capitalized does not provide much of a clue, because words belonging to any other part-of-speech may also occur capitalized. Even worse, many further similar words are ambiguous in their meaning and part-of-speech, e.g. *Dichten* ('density.N.PL' or capitalized 'dense.ADJ.NOM.PL.DEF' or 'compose (e.g. a poem).V.INF'), *schlichten* ('simple.ADJ.NOM.PL.DEF' or 'mediate.V.INF').

It is much easier to reason about possible tags for those words if we take into account morphologically related words. For example, we might observe words like *richtet* or *richtete*, which together with *richten* look unambiguously like a verb paradigm. Similarly, the occurrence of a form like *rechtes* can convince us that *rechten* is an adjective, because verbs do not take the suffix *-es*. For ambiguous forms, we will likely find parts of different paradigms, for example *schlichtet* (verb) and *schlichtes* (adjective), which will allow us to notice the ambiguity. Of course, in order to conduct such analysis, we have to know which affix configurations are characteristic for which part of speech. This is the part that we are going to learn from labeled data.

4.1 Applying Tagged Rules to Untagged Words

Let us assume that we have learned a morphology model on tagged data. Now we are presented with a new set of words, possibly containing many words not present in the original training set. In this section, we will show how the trained model can be applied to derive guesses for tags in the new vocabulary. The approach follows the idea sketched in the previous section: the tag of the word will be determined by the neighboring words, together with the knowledge about the morphology contained in tagged rules.

To illustrate the approach with a minimal example, let us assume that our tagset consists of only three tags: NN, VVINF and VVFIN, and that the untagged vocabulary consists of the German words *machen*, *mache*, *macht*. We compute the edge probabilities for every edge that is possible according to the model, under every tagging. For example, the model might consist of the rules and parameters listed in Table 1.

Using those values, we can reason about the possible taggings based on an untagged graph.

[3] It is very difficult to find plausible examples of this phenomenon in English due to its small inflection and very productive zero-affix derivation.

[4] All cited words are ambiguous in their inflectional form (but not part-of-speech). The glosses shown here are picked as examples.

r	θ_r
$/X\text{en}/_{\text{NN}} \rightarrow /X\text{e}/_{\text{NN}}$	0.3
$/X\text{en}/_{\text{VVINF}} \rightarrow /X\text{e}/_{\text{VVFIN}}$	0.01
$/X\text{en}/_{\text{VVINF}} \rightarrow /X\text{t}/_{\text{VVFIN}}$	0.2

Table 1: An example model learnt from a tagged vocabulary.

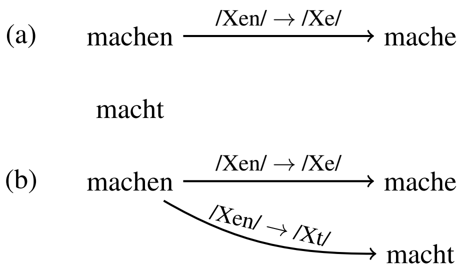


Figure 2: Two possible morphology graphs corresponding to the words *machen*, *mache*, *macht*. What does each of them tell us about the possible tags of those words according to Table 1?

Consider the two graphs shown in Fig. 2. What does each of them say about the possible taggings?

Graph 2a is consistent with either $\{machen_{\text{NN}}, mache_{\text{NN}}\}$ or $\{machen_{\text{VVINF}}, mache_{\text{VVFIN}}\}$, since the only edge in this graph is possible with both labelings. Note that the edge containing noun labels has much higher probability, so this graph suggests a strong preference for the noun hypothesis. It does not say anything about the possible tags of *macht*. On the other hand, the only tagging consistent with the graph 2b is $\{machen_{\text{VVINF}}, mache_{\text{VVFIN}}, macht_{\text{VVFIN}}\}$, since the edge between *machen* and *macht* is only possible if *machen* is a verb infinitive. It is important to notice that adding an edge between *machen* and *macht* diametrically changed the possible taggings for *mache*, although it is not touched by the edge in question. This illustrates how the graph model captures dependencies between the tags across a whole paradigm, although the edge probabilities are local.

Moving on to formalize the above introduction, let the tag of word v be denoted by a random variable T_v. We will be interested in the expected probability of a word v obtaining tag t, given an untagged vocabulary and a tagged morphology model. We call this value $\tau_{v,t}$ and define it as follows:

$$\tau_{v,t} = \mathbb{E}_{E|V,R,\theta}\mathbb{E}_{T|V,E,R,\theta}\delta_{T_v,t} \qquad (3)$$

$\delta_{x,y}$ denotes the Kronecker delta. Thus, we take the expectation over all possible graphs for the given vocabulary, and then over all possible taggings for a fixed graph. The inner expectation can be computed exactly by a variant of Forward-Backward algorithm introduced in Sec. 4.2. In order to approximate the outer expectation, we use Markov Chain Monte Carlo sampling over untagged graphs as described by (Sumalvico, 2017). However, the sampling algorithm will need some modifications, as contrary to the original approach, the edge probabilities are not independent of the graph structure. We describe those modifications in Sec. 4.3. Finally, the computed values of $\tau_{v,t}$ will be fed to an already pre-trained HMM to provide it with guesses for the tags of unknown words, before it is reestimated on untagged text. This procedure is described in detail in Sec. 4.4.

4.2 The Forward-Backward Algorithm for Trees[5]

In order to compute $\tau_{v,t} = \mathbb{E}_{T|V,E,R,\theta}\delta_{T_v,t}$ for a fixed graph (V, E), let us recall the well-known Forward-Backward algorithm used for Hidden Markov Models.[6] The HMMs employed for POS tagging operate on sentences, which are linear sequences of words (Fig. 3). The summing over all possible tag sequences is tackled by introducing the so-called *forward probability* (usually written as α) and *backward probability* (β). The forward probability $\alpha_{v,t}$ of a node v with tag t is the joint probability of v and all its predecessors *and* v having tag t, while the backward probability $\beta_{v,t}$ is the probability of all successors of v onwards *provided that* v has tag t. The product of the two is the probability of the whole sequence *and* v having tag t.

The concepts of *successor* and *predecessor* can be applied to a tree model as well (Fig. 4). In this case, $\beta_{v,t}$ is the probability of the subtree rooted in v provided that v has tag t, and $\alpha_{v,t}$ is the probability of the rest of the tree *and* v having tag t. Note that $\alpha_{v,t}$ involves not only the path leading

[5] The algorithm presented in this section is similar, but not identical, to the one presented by (Crouse et al., 1998). The underlying models differ slightly.

[6] See for example (Manning and Schütze, 1999) or (Jelinek, 1997) for an introduction to HMMs and the Forward-Backward algorithm.

$$v_1 \to v_2 \to v_3 \to v_4 \to v_5 \to v_6 \to v_7 \to v_8 \to v_9$$

Figure 3: The Forward-Backward computation for a linear sequence in an HMM. $\alpha_{v_6,t} = P(v_1, \ldots, v_6, T_6 = t)$, whereas $\beta_{v_6,t} = P(v_7, v_8, v_9 | T_6 = t)$.

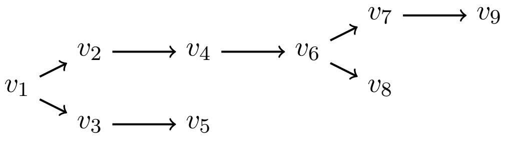


Figure 4: The Forward-Backward computation for a tree. Also here, $\alpha_{v_6,t} = P(v_1, \ldots, v_6, T_6 = t)$ and $\beta_{v_6,t} = P(v_7, v_8, v_9 | T_6 = t)$.

from root to v, but also all side branches sprouting from this path.

We will now derive recursive formulas for forward and backward probabilities for the tree case. For this purpose, we assign a *transition matrix* to each graph edge. For each possible tagging of the source and target node, the transition matrix contains a value $\frac{\theta_r}{1-\theta_r}$, where r is the corresponding tagged rule. Continuing the example from 4.1, the probabilities in Table 1 yield the following transition matrix:

$$T^{(\text{machen},\text{mache},/X\text{en}/\to/X\text{e}/)} =$$

$$\begin{array}{c c} & \begin{array}{ccc} \text{NN} & \text{VVINF} & \text{VVFIN} \end{array} \\ \begin{array}{c} \text{NN} \\ \text{VVINF} \\ \text{VVFIN} \end{array} & \begin{pmatrix} \frac{0.3}{1-0.3} & 0 & 0 \\ 0 & 0 & \frac{0.01}{1-0.01} \\ 0 & 0 & 0 \end{pmatrix} \end{array} \tag{4}$$

Furthermore, let $\lambda_{v,t}$ denote the probability that the node v with tag t is a leaf, i.e. it contains no outgoing edges. Then:

$$\lambda_{v,t} = \prod_{r \in R} \prod_{(v',t') \in r(v,t)} (1 - \theta_r) \tag{5}$$

It is trivial to see that for leaf nodes, $\beta_v = \lambda_v$. For a non-leaf node v, we multiply λ_v by the terms $\frac{\theta_r}{1-\theta_r}$ for each outgoing edge, summed over every possible tagging. In the matrix and vector notation, this corresponds to the following formula:[7]

$$\beta_v = \lambda_v * \prod_{(v,v',r) \in out_{\mathcal{G}}(v)} T^{(v,v',r)} \beta_{v'} \tag{6}$$

$out_{\mathcal{G}}(v)$ denotes the set of outgoing edges of node v. In order to compute the forward probability of a non-root node v, let us assume that it is derived by the edge (v', v, r). In addition to the forward probability of v' (the parent of v) and the edge deriving v, we also take into account the side branches, i.e. all subtrees rooted in children of v' other than v. The resulting formula is as follows:

$$\alpha_v = \alpha_{v'} * \prod_{\substack{(v',v'',r') \in out_{\mathcal{G}}(v') \\ v'' \neq v}} T^{(v',v'',r')} \beta_{v''} \cdot T^{(v',v,r)} \tag{7}$$

The last remaining issue is the forward probability of root nodes. The generative model defined by (2) contains a distribution $\rho(\cdot)$ over arbitrary strings, from which the string forms of the root nodes are chosen. In the tagged case, we augment this distribution to $\rho(v,t) = \rho_{\text{string}}(v)\rho_{\text{tag}}(t\ |v)$. As in (Sumalvico, 2017)'s experiments, we use a simple character-level unigram model for $\rho_{\text{string}}(\cdot)$. In order to model the distribution $\rho_{\text{tag}}(t|v)$, which predicts a word's tag from its string form, we use a character-level recurrent neural network. Note that the forward probability of root nodes is equal to their probability according to the root model, i.e. $\alpha_{v,t} = \rho(v,t)$.

4.3 Modifications to the Sampling Algorithm

In order to approximate the expected value over possible graphs given a vocabulary, we use the Metropolis-Hastings sampler proposed by (Sumalvico, 2017). The algorithm computes each new graph by proposing a small change in the previous graph. The possible moves are: adding or deleting a single edge, exchanging an edge for another one with the same target node and the so-called 'flip' move. The latter simultaneosly exchanges two edges for two others and is designed as a way to prevent the creation of a cycle while adding an edge. The algorithm subsequently computes an acceptance probability from the probabilities of the changed edges and decides whether to accept the proposed graph as a new sample point.

As we have seen in the analysis of Fig. 2, adding an edge typically has consequences for the whole subtree, in which the edge is added. The values τ_v for all nodes in the subtree may change, which in turn changes the probability of all edges in the subtree. This behavior constitutes a significant difference compared to the original sampling algorithm, in which the edge probabilities were independent of the graph structure and the cost of a

[7] Asterisk denotes element-wise multiplication, while dot or no symbol denotes the dot product.

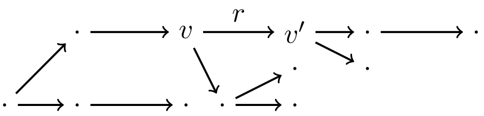


Figure 5: Adding or removing the edge (v, v', r).

change could be easily computed from the cost of added and removed edges. However, we can use the forward and backward probabilities to score the changes.

Observe that for every node v, the value $\sum_t \alpha_{v,t}\beta_{v,t}$ is the probability of whole subtree, to which v belongs. This property – being able to compute the probability of a whole subgraph using the values of a single node – is crucial in evaluating the sampler moves.

Adding or removing a single edge. Consider the graph in Fig. 5, to which the edge (v, v', r) is supposed to be added. Without this edge, we have two separate trees with a total probability expressed by $(\sum_t \alpha_{v,t}\beta_{v,t})(\sum_t \alpha_{v',t}\beta_{v',t})$. After adding this edge, we obtain a single tree. As v obtains a new child node, β_v will change. Let β'_v denote the new value, which can be computed as follows:

$$\beta'_v = \beta_v * T^{(v,v',r)}\beta_{v'} \tag{8}$$

Note that neither α_v nor $\beta_{v'}$ is affected by adding this edge. The probability of the new tree is simply $\sum_t \alpha_{v,t}\beta'_{v,t}$. If the move is accepted, the β values of all nodes on the path from the root to v have to be updated, as well as the α values of all nodes except for this path.

Deleting an edge involves a very similar computation. In this case, the probability of the graph before deletion is $\sum_t \alpha_{v,t}\beta_{v,t}$, whereas the probability after deletion is $(\sum_t \alpha_{v,t}\beta'_{v,t})(\sum_t \rho_{v',t}\beta_{v,t})$. Here, $\beta'_{v,t}$ is the updated backward probability of v excluding the deleted edge.

Other moves. When exchanging a single edge to another one with the same target node, we already need to be careful, as two distinct cases arise: either the change takes place within one tree, or it involves two separate trees. If we proceeded as in the previous paragraph, those cases would require different formulas. Instead of conducting such a detailed analysis of the changes, we apply a more general approach that covers the 'flip' moves as well.

First, we group all edges that are to be changed (added or deleted) according to the tree, to which they belong (more specifically, according to the root of the tree, to which the edge's source node currently belongs). In each tree, we look for a minimum subtree that contains all the changes (Fig. 6). We build a copy of this subtree with all changes applied and recompute the forward probability for its root and the backward probabilities for the whole subtree. Finally, we use the (newly computed) forward and backward probability of the subtree root to determine the probability of the whole tree after changes.

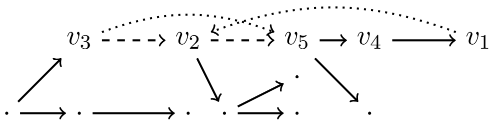


Figure 6: In case of a 'flip' move, the smallest subtree containing all changes is the one rooted in v_3. The deleted edges are dashed, while the newly added edges are dotted. In order to obtain the new β_{v_3}, we recompute the backward probabilities in the whole subtree. α_{v_3} is not affected by the changes. (The node labels are consistent with the definition in (Sumalvico, 2017).)

4.4 Extending an HMM with New Vocabulary

The vectors τ_v obtained from the sampling approach sketched in the previous subsections provide us with a morphologically motivated POS-tag distribution for words from the untagged corpus. We augment the HMM's emission probability matrix with those values for previously unseen words.[8]

5 Experiments

5.1 Experiment Setup

We conducted evaluation experiments for 9 languages: Ancient Greek (GRC), German (DEU), Finnish (FIN), Gothic (GOT), Latin (LAT), Latvian (LAV), Polish (POL), Romanian (RON) and Russian (RUS), using the Universal Dependencies[9] corpora. Each corpus is randomly split into

[8] The values τ_v are conditional probabilities of a tag given word, while the emission probabilities of an HMM are probabilities of a word given tag. We use the Bayes' formula to convert the former to the latter. We obtain the marginal probabilities of tags during the HMM pre-training and assume equal frequencies for unseen words.

[9] http://universaldependencies.org

1	Fitting on the training set.
2	**1** + estimation on the development set.
3	**2** + extension of the vocabulary with random initialization.
4	**2** + extension of the vocabulary with tag guesses provided by a character-based RNN.
5	**2** + extension of the vocabulary with tag guesses provided by the whole-word morphology model.
6	**2** + extension of the vocabulary with gold standard tag guesses.

Table 2: Different setups of the HMM tagger used in the tagging experiment.

training, development and testing dataset in proportions 10%:80%:10%. An HMM tagger is fitted to the (labeled) training data using ML estimation. The training dataset is also used to learn tagged morphological rules. Then, we remove the labels from the development corpus and re-estimate the HMM on this corpus using Baum-Welch algorithm. This step is performed in several configurations: either with or without vocabulary extension. In the latter case, all types occurring in the development corpus, but not known to the HMM (i.e. not occurring in the training corpus) are added to the vocabulary before the estimation. Finally, the tagging accuracy is assessed on the testing dataset. The details of the possible configurations are shown in Table 2.

For each corpus, two kinds of datasets are prepared: with *coarse-grained* and *fine-grained* tags. In the former case, the UPOS tagset is used, which amounts to around 15 tags for every language. In the latter, all inflectional information provided by the corpus annotation is additionally included, which results in several hundred different tags (depending on the language). For example, in the Latin corpus, we have tokens like `beati<ADJ>` in the coarse-grained and `beati<ADJ><NOM><POS><MASC><PLUR>` in the fine-grained case.

The morphology-based tag guessing approach developed in the previous sections is compared to the guesses made only based on the word's string form. The latter are provided by the distribution $\rho_{\text{tag}}(\cdot)$, i.e. a character-based recurrent neural network, mentioned at the end of Sec. 4.2.

5.2 Evaluation Measures

Two kinds of evaluation are performed: *lexicon* and *tagging* evaluation. In the first case, the quality of tag guesses for unknown words, τ_v, is measured directly. It is desirable for those values to not only predict the correct tag for unambiguous words, but also to handle ambiguity correctly, which means providing probabilities that correspond to the expected frequency of a word with the certain tag. We derive the gold standard data from the labels in the development set using the following formula:

$$\hat{\tau}_{v,t} = \frac{n_{v,t}}{\sum_{t'} n_{v,t'}} \qquad (9)$$

with $n_{v,t}$ being the number of occurrences of word v with tag t in the development set. This way, true ambiguities (with roughly equal frequency of different taggings) are treated differently than rare taggings, which may result from tagging errors or some obscure, infrequent meanings. The accuracy is computed as follows:

$$accuracy = \frac{1}{|V|} \sum_{v \in V} \sum_{t} \min\{\tau_{v,t}, \hat{\tau}_{v,t}\} \qquad (10)$$

It is intentionally a very demanding measure: it achieves 100% for a given word only if the probability mass is distributed exactly according to the corpus frequency of the tagging variants, which is virtually impossible for ambiguous words. Hence, low scores according to this measure are not surprising and do not necessarily represent a bad-quality tagging. We decided for this measure, because it is easier to interpret than e.g. KL-divergence.

In the tagging evaluation, we evaluate the impact of providing tag guesses on a real POS-tagging task. The evaluation measure used there is the standard accuracy, i.e. the percentage of correctly tagged tokens.

5.3 Results

Table 3 shows the results of the lexicon evaluation. The figures show clearly that the morphology-based method outperforms the RNN in predicting possible tags for a given word type. Probably the most important reason for that is that the RNN always makes unsharp predictions – it never attributes the whole probability mass to a single tag. On the other hand, the graph-based morphology model often makes unambiguous predictions

Lang.	coarse-grained		fine-grained	
	RNN	WWM	RNN	WWM
GRC	.607	.660	.229	.300
FIN	.507	.643	.255	**.411**
DEU	.616	.676	.154	.210
GOT	.483	**.661**	.157	.308
LAT	.544	**.704**	.236	**.448**
LAV	.506	.646	.240	.368
POL	.587	.695	.196	.315
RON	.540	**.705**	.241	.301
RUS	.682	.769	.330	**.522**

Table 3: Lexicon evaluation.

Lang.	1	2	3	4	5	6
GRC	.669	.742	.732	.789	.791	.857
FIN	.606	.744	.728	.795	**.823**	.838
DEU	.771	.755	.831	.849	.851	.836
GOT	.768	.770	.803	.819	.828	.865
LAT	.743	.808	.816	.825	**.866**	.856
LAV	.670	.723	.731	.796	.803	.848
POL	.715	.787	.745	.781	**.842**	.829
RON	.785	.846	.853	.880	.884	.870
RUS	.809	.877	.877	.903	.905	.914

Table 4: Tagging accuracy with coarse-grained tags.

Lang.	1	2	3	4	5	6
GRC	.566	.464	.569	.628	.638	.699
FIN	.535	.377	.567	.651	**.675**	.717
DEU	.594	.498	.587	.618	.620	.631
GOT	.636	.550	.659	.677	.678	.739
LAT	.590	.493	.617	.660	**.687**	.734
LAV	.585	.471	.594	.657	.668	.713
POL	.554	.437	.575	.625	.627	.679
RON	.712	.713	.759	.764	.761	.824
RUS	.731	.654	.736	.780	.789	.813

Table 5: Tagging accuracy with fine-grained tags.

because of the absence of rules allowing for alternatives (a phenomenon illustrated in Fig. 2). Thus, the latter achieves better scores especially on correctly tagged unambiguous words.

The comparison of tagging accuracies is shown in Tables 4 and 5. The columns correspond to the tagger configurations explained in Table 2. In general, a rise of the score from left to right is to be expected.

The difference between columns 1 and 2 illustrates the influence of reestimating the trained model on unlabeled data without adding the OOV words to the vocabulary. Interestingly, this results in an improvement in case of the coarse-grained tagset, but in a decline when using the fine-grained tagset, both significant. However, adding the OOV words from the development set to the vocabulary, even with randomly initialized probabilities (column 3), further improves the accuracy (with a few exceptions), so that the result is consistently better than column 1. Column 4 introduces intrinsic tag guessing as initial probabilities for newly added words, rather than random values. This results in a further improvement, especially significant for Finnish, Ancient Greek and Latvian (both coarse-grained and fine-grained).

The most important comparison in this evaluation is between column 4 and 5. This illustrates the benefit of using extrinsic tag guessing (column 5), rather than intrinsic. This results in a consistent improvement, ranging from very slight to significant. The most significant improvements are shown in bold. Finally, column 6 displays what one might expect to be the upper bound on the accuracy: the one that would be achieved if tags were guessed perfectly (i.e. as $\hat{\tau}_v$). Surprisingly, it is not always the highest value in a row. It looks as if taking into account some wrong taggings during Baum-Welch estimation could accidentally improve the estimation, because the wrong tag might *also* have occurred in the given context. This seems especially plausible for cases like common and proper nouns, which are often confused.

5.4 Discussion

Although the results speak consistently in favor of using morphology-based tag guessing, as well as using tag guessing at all, the benefits are somewhat less clear than one could expect. Especially in the case of fine-grained tags, our expectation was that, due to the discrete nature of morphological rules, at least the tags of unambiguous words would be identified mostly correctly. This was supposed to greatly improve the Baum-Welch estimation, as instead of considering many hundred possible tags, the correct one is already known, which turns the estimation into almost supervised learning. However, we had underestimated the impact of the small size of training corpus on the morphology component. Most fine-grained tags are very rare, so many morphological rules related to such forms are not learnt.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the EACL*. pages 569–578.

Matthew S. Crouse, Robert D. Nowak, and Richard G. Baraniuk. 1998. Wavelet-based statistical signal processing using hidden markov models. *IEEE Transactions on Signal Processing* 46(4):886–902.

Jean-Baptiste Durand, Paulo Gonçalvès, and Yann Guédon. 2004. Computational methods for hidden markov tree models – an application to wavelet trees. *IEEE Transactions on Signal Processing* 52(9):2551–2560.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL-HLT*. pages 1185–1195.

Manaal Faruqui, Ryan T. McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *TACL* 4:1–16.

Alan Ford, Rajendra Singh, and Gita Martohardjono. 1997. *Pace Pāṇini: Towards a word-based theory of morphology*. American University Studies. Series XIII, Linguistics, Vol. 34. Peter Lang Publishing, Incorporated.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. Hunpos – an open source trigram tagger. In *Proceedings of the ACL 2007 Demo and Poster Sessions*. Prague, pages 209–212.

Maciej Janicki. 2015. A multi-purpose bayesian model for word-based morphology. In Cerstin Mahlow and Michael Piotrowski, editors, *Systems and Frameworks for Computational Morphology – Fourth International Workshop, SFCM 2015*. Springer.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

Shuhei Kondo, Kevin Duh, and Yuji Matsumoto. 2013. Hidden markov tree model for word alignment. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. pages 503–511.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Beáta B. Megyesi. 2009. The open source tagger hunpos for swedish. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA)*.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics* 23(3):405–423.

Sylvain Neuvel and Sean A. Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*. pages 31–40.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. pages 88–99.

Maciej Sumalvico. 2017. Unsupervised learning of morphology with graph sampling. In *Proceedings to RANLP 2017*. Varna, Bulgaria.

Zdeněk Žabokrtský and Martin Popel. 2009. Hidden markov tree model in dependency-based machine translation. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*. Singapore, pages 145–148.

Word Sense Disambiguation Based on Constrained Random Walks in Linked Semantic Networks

Arkadiusz Janz, Maciej Piasecki
Wrocław University of Science and Technology, Poland
{arkadiusz.janz|maciej.piasecki}@pwr.edu.pl

Abstract

Word Sense Disambiguation remains a challenging NLP task. Due to the lack of annotated training data, especially for rare senses, the supervised approaches are usually designed for specific subdomains limited to a narrow subset of identified senses. Recent advances in this area have shown that knowledge-based approaches are more scalable and obtain more promising results in all-words WSD scenarios. In this work we present a faster WSD algorithm based on the Monte Carlo approximation of sense probabilities given a context using constrained random walks over linked semantic networks. We show that the local semantic relatedness is mostly sufficient to successfully identify correct senses when an extensive knowledge base and a proper weighting scheme are used. The proposed methods are evaluated on English (SenseEval, SemEval) and Polish (Składnica, KPWr) datasets.

1 Introduction

Semantic analysis is the process of understanding the underlying meaning in text. Building a rich semantic representation is a crucial element of modern Natural Language Processing NLP. It allows us to build comprehensive text representations in order to analyse the underlying text meaning more effectively. Semantic information helps us to resolve the issue of textual ambiguity. The main aim of semantic analysis is to differentiate texts which use the same vocabulary yet present different ideas about the same topic. One of the most important building blocks of semantic analysis is *word sense disambiguation* (WSD). WSD aims at solving the problem of lexical ambiguity, i.e. two or more meanings being represented by one word e.g. English words *space*, *shape*, or Polish words *tło* (background color, a context or an incidental music) *wnętrze* (a room, a soul or a set). This lexical ambiguity is still an open issue for semantic analysis due to the lack of required training data and the computational power required to process the growing number of possible classes. The task of word sense disambiguation is usually solved by mapping words onto their senses given a particular sense inventory.

WSD can be performed in a supervised way, i.e. on the basis of the annotated lexical meanings in training texts, or in an unsupervised way, i.e. sense induction from texts.

Supervised machine learning approaches can achieve a very good performance when trained on a large training data annotated with good inter-annotator agreement and well designed features.

However, it is difficult to acquire large training data for WSD because training data is usually manually annotated and manual annotation is labour-intensive and thus costly[1]. Semi-automatic approaches on the other hand often result in lower quality data blurred with statistical noise.

Thus, the best performing supervised approaches to WSD task are usually limited to a specific narrow subset of training vocabulary. Moreover, supervised approaches are strongly connected with the underlying domain of the training data. Sense induction methods require only large amounts of text, but representative in relation to all word senses. However, senses induced automatically are mostly difficult to be matched against dictionaries or other word sense inventories created manually. Thus, semi-supervised methods based on lexical knowledge bases describing word senses, e.g. wordnets, offer a poten-

[1] It is barely possible to scale it up to a couple of dozens of thousands words well described according to all its senses

Proceedings of Recent Advances in Natural Language Processing, pages 516–525,
Varna, Bulgaria, Sep 2–4 2019.

tially good compromise. Such methods can cover, at least potentially, all word senses described in a lexical knowledge base. Very large wordnets describing more than 100,000 words (lemmas) and their senses by hundreds of lexico-semantic relation instances have been constructed for several languages, including the English and Polish language. A large and dense network of relations seems to be a good basis for mapping word occurrences in texts onto their senses.

An enormous amount of semantic data appeared on the Web in recent years. With the growth of resources joining the Linked Open Data collection the available semantic information becomes a very important knowledge source for WSD tasks.

Semi-supervised methods are very often based on the idea of mapping texts onto the wordnet graph of lexico-semantic relations as the initial activation of the graph nodes. Next a recursive spreading activation method, mostly based on PageRank (Page et al., 1998) is applied. However, as seen in the following paragraphs, such methods raise problems with efficiency.

In this paper we explore Monte Carlo methods based on a random walk algorithm for the task of Word Sense Disambiguation. We show that the lexical knowledge base with its expansions and the way they are exploited has a strong impact on WSD performance and the proposed method allows for the efficient utilisation of large lexical knowledge bases. The presented solutions are evaluated on popular Polish and English benchmark datasets.

2 Related Work

The first group of approaches is based on supervised machine learning algorithms. The researchers have adapted many different ML methods for WSD task. One can find the classical NLP approaches based on feature engineering and popular classification algorithms e.g. decision trees, decision lists, Naive Bayes solutions, kNN, adaptive boosting, as well as more modern approaches using Deep Learning with LSTM, biLSTM and more sophisticated variants of neural architectures. The main issue is that most of the words are strongly imbalanced in terms of their sense distribution, thus, due to the lack of required training data, the supervised approaches present a lower recall in all-words WSD setting.

The knowledge-based solutions use the structural properties of existing sense inventories e.g. wordnets.

One of the first solutions to WSD task for Polish (Baś et al., 2008) was mainly focused on supervised approaches. The authors trained the classifiers on a relatively small dataset for a preselected, annotated vocabulary. Later works were focused mainly on WSD methods close to sense induction from text corpora, e.g. (Broda and Piasecki, 2011).

Weakly supervised WSD approaches are usually based on sense inventories and their semantic structure describing senses. The main assumption is that the words being a part of a text can be mapped onto their potential senses existing in a given sense inventory, mainly the synsets existing in a wordnet. The senses activate local subgraphs of wordnet's semantic structure to activate meanings possibly being relevant to a given text fragment. As the initial activation is sparse a kind of spreading activation algorithm is next applied in order to recursively concentrate this information in some "hot" areas and identify word senses located in them or close to them. The identified synsets should be the most likely senses for words in the text. There are several parameters to set in this general scheme: the initial activation (coming from the text words), spreading activation algorithm (topology and relations) and identification of association between "hot" areas and senses to be selected. Various methods following this scheme were proposed.

Weakly supervised WSD methods are mostly based on the recursive PageRank algorithm (Page et al., 1998) for spreading activation. Mihalcea et al. (2004) proposed application of the original PageRank to WSD called Static PageRank.

PageRank algorithm (henceforth PR) is an iterative method for ranking nodes in the graph G. In WSD the nodes in G represent synsets and the edges of G correspond to wordnet relations (linking synsets, and in some wordnets also linking specific word senses).

(Agirre and Soroa, 2009; Agirre et al., 2014) proposed a modified version called Personalised PageRank (PPR) in which the values in $\mathbf{v}$, called personalised vector, depend on the textual context of the disambiguated word. The non-zero score values are assigned to those nodes which are contextually supported. In PPR all words from the context are disambiguated at once. The $\mathbf{v}$ values are equal to:

$$\mathbf{v}[i] = \frac{1}{\frac{CS}{NS(i)}}, \quad i = 1, 2, ..., N \qquad (1)$$

where CS is the number of different lemmas in the context, $NS(i)$ – the number of synsets sharing the same context lemma with the synset i.

In addition a modified version of PPR called *Personalised PageRank Word-to-Word* (PPR_W2W) was also presented by (Agirre and Soroa, 2009; Stevenson et al., 2012), in which a word to be disambiguated is excluded from the occurrence contexts, i.e. all synsets of this word have initial scores in $\mathbf{v}$ set to zero. Thus, PPR_W2W cannot be run once for all ambiguous words in the context. The vector $\mathbf{v}$ must be initialised individually for each ambiguous word in the context – this is a disadvantage of PPR_W2W. A potential advantage is the removal of the effect of mutual amplification of the closely connected senses of the word being disambiguated. All PR-based WSD algorithms showed good performance that is increasing with the enlargement and enrichment of the knowledge base. However, with larger graphs the time of processing increases non-linearly causing a significant drop in efficiency. The problem is especially visible in the case of PPR_W2W in which the algorithm must be restarted several times per context.

In (Kędzia et al., 2014), PR-based WSD algorithm for Polish was presented and run with the help of plWordNet 2.1. The graph consisted of synsets linked by edges representing a selected subset of the synset relations. Next several versions of the PR-based algorithms, namely Static PR, PPR and PPR_W2W was applied to Polish texts and plWordNet 2.2 in (Kędzia et al., 2015). The achieved precision (on *KPWr*) was in the range 42.79%-50.73% for nouns and 29.79%-32.94% for verbs. PPR_W2W produced the best results. Different variants of combining plWordNet with the *Suggested Upper Merged Ontology* (SUMO) (Pease, 2011) on the basis of the mapping constructed in (Kędzia and Piasecki, 2014). All three PR-based algorithm were evaluated. A slight improvement of the precision for nouns up to 50.89% for PPR_W2W could be observed when the two joined graphs were treated as one large graph.

In (Pershina et al., 2015) the authors presented a graph-based algorithm using random walks algorithm for named entity disambiguation. The motivation for their work was the fact that PR-based methods mainly rely on global coherence, but the methods should utilise the local similarity more effectively.

3 Knowledge Base

The general sense inventory for all-words WSD is usually created on the basis of a wordnet. Wordnets can be presented as graphs with nodes representing word senses or synsets (but also with two types of nodes) and edges expressing the structure of the lexico-semantic relations described in a given wordnet. The methods based on lexical knowledge bases usually explore the lexico-semantic relations represented by a wordnet to disambiguate the words given the context.

Most wordnet relations are paradigmatic, but for WSD we also need syntagmatic relations, rarely covered by wordnets, because the plain wordnet structure might be insufficient to successfully disambiguate a text. As the large public sources like Linked Open Data are available, we can try to apply them for the expansion of the lexical knowledge base. They may also contain Named Entities which can be very important for WSD, e.g. helping to identify the narrow semantic context.

More formally, a *lexical knowledge base* is a graph $G(V, E)$ consisting of nodes and edges, where $V = \{v_1, v_2, ..., v_N\}$ represents a set of concepts (modelling lexical meanings) and $E = \{e_1, e_2, ..., e_M\}$ a set of edges corresponding to lexico-semantic associations linking these concepts.

3.1 Existing Knowledge Bases

In literature, we can find many attempts to combine Princeton WordNet with resources of many types to obtain a better knowledge base for WSD.

UKB lexical knowledge base, e.g. (Agirre and Soroa, 2009), consists of Princeton WordNet 3.0 or eXtended WordNet (Harabagiu et al., 1999) which was expanded by introducing links extracted from SemCor, manually disambiguated glosses from Princeton WordNet Gloss Corpus, and Wikipedia.

BabelNet (Navigli and Ponzetto, 2012) was initially created by linking the largest multilingual Web encyclopedia, i.e. Wikipedia, with the most popular computational semantic lexicon, i.e., WordNet. Later it was expanded with a number of resources:

- OmegaWiki, a large collaborative multilingual dictionary (January 2017 dump);
- Wiktionary, a collaborative project to produce a free-content multilingual dictionary (February 2018 dump);
- Wikidata, a free knowledge base that can be read and edited by humans and machines alike (February 2018 dump);
- Wikiquote, a free online compendium of sourced quotations from notable people and creative works in every language (March 2015 dump);
- VerbNet (Kippera et al., 2006), a Class-Based Verb Lexicon (version 3.2);
- Microsoft Terminology, a collection of terminologies that can be used to develop localised versions of applications (July 2015 dumps);
- GeoNames, a free geographical database covering all countries and containing over eight million place names (April 2015 dump);
- FrameNet (Ruppenhofer et al., 2016), a lexical database of English that is both human- and machine-readable (version 1.6);
- Open Multilingual WordNet (Bond and Foster, 2013), a collection of wordnets available in different languages.

The mappings provided by BabelNet (especially the links between synsets and Wikipedia pages) were built semi-automatically. We chose to manually map synsets of plWordNet by lexicographers instead of use the semi-automatic mappings provided by BabelNet to have more control over the accuracy of our results.

3.2 Presented Knowledge Base

For the work presented here, two knowledge graphs were built on the basis of the two largest wordnets, namely *plWordNet 3.2* (Maziarz et al., 2016) for Polish, and *Princeton WordNet 3.1* (Fellbaum, 1998) for English. They are mutually mapped on each other as a result of the laborious work of bilingual lexicographers.

3.3 Expansions

The initial performance of the algorithms can be moderate when the knowledge-base is limited only to the plain wordnet structure. We can significantly improve the overall performance by introducing new semantic links to the basis knowledge graph. In this work we expanded the ideas presented in (Agirre and Soroa, 2009), (Agirre et al., 2018) and (Moro et al., 2014). Following the procedure presented in (Agirre et al., 2018) we extended the structure of our knowledge graph by including the links extracted from the the Princeton WordNet Gloss Corpus[2] including manually disambiguated glosses.

The Suggested Upper Merged Ontology (SUMO) (Pease, 2011) is a formal representation of concepts, organised into hierarchies of classes and subclasses, which is widely used for semantic analysis in NLP. The lexical senses of PWN 3.0 and plWordNet 3.2 have been mapped onto their equivalent concepts of SUMO. The mapping procedure for plWordNet was based on interlingual links existing between plWordNet and PWN (Maziarz et al., 2016) and the initial mapping of PWN senses to SUMO ontology (Kędzia and Piasecki, 2014). We used the structure of SUMO ontology as a more general semantic description for lexical senses existing in wordnet. The semantic structure of our knowledge base was extended with concepts and links existing in the SUMO ontology by attaching the concepts to corresponding synsets and linking them with SUMO relations.

Wikipedia has opened many new opportunities for semantic analysis. The structure of Wikipedia has been used as a knowledge-base for the task of named entity disambiguation (NED), but also adapted for WSD. Graph-based approaches for computing semantic relatedness and disambiguation can be improved by using the semantic information contained in Wikipedia and expand the underlying knowledge base e.g. a wordnet. For this work we decided to add the links extracted from Wikipedia by using the mapping of lexical senses to equivalent Wikipedia articles. For every mapped synset we added new semantic links by analysing the content of the page and extracting monosemous words. The lexical senses of monosemous words were linked to mapped synset.

[2] http://wordnetcode.princeton.edu/glosstag.shtml

4 Methods

The main drawback of PageRank-based methods is that to compute the score of a sense given the context they have to compute the features with respect to the global structure of a given knowledge graph. This means the PageRank values have to be computed for every single node in the graph. To avoid this issue we use only the local approximation of PageRank as it was presented in (Avrachenkov et al., 2007).

4.1 PageRank Based Methods

The computational complexity of Personalised PageRank (PPR) is still a limiting factor for fast, real time WSD of large textual data. The naive algorithm for computing PageRank values requires to iterate over the entire graph. The most popular approach to compute PPR scores is the Power Iteration method (Berkhin, 2005), where the score is defined in a recursive way taking into account the global information. Recently the methods of local approximation of PageRank scores have received a lot of attention, especially in the case of dynamic graphs where the structure of the graph changes in time.

Let $G = (V, E)$ be a graph where V is a set of nodes and E represents a set of edges. The overall number of nodes and edges is N and M, respectively. We can define the adjacency matrix $A_{N \times N}$ of the graph G as $A_{N \times N} = [a_{ij}]$, $i \in \{1, 2, ..., N\}, j \in \{1, 2, ..., N\}$, where the values a_{ij} are representing links existing in the graph, where $a_{ij} = 1$ if there is an edge pointing from node v_i to v_j, otherwise $a_{ij} = 0$. A Markov transition matrix P can be defined as $P_{N \times N} = [p_{ij}]$ with p_{ij} values being normalised by the number of outgoing links (from node v_i) $p_{ij} = \frac{1}{d_i}$ if $a_{ij} = 1$, otherwise $p_{ij} = 0$. The graph might also contain dangling nodes without any outgoing links. To handle these cases p_{ij} values for dangling nodes are usually replaced by a constant $\frac{1}{N}$, which means adding a link to every node in the graph. The static PageRank can be interpreted then as a stationary distribution π of a Markov chain with final transition matrix $\widetilde{P}$ (Google's matrix):

$$\widetilde{P} = cP + (1 - c)\frac{1}{N}R \tag{2}$$

The matrix R consists of entries being equal to one. The value $c \in (0, 1)$ is the probability that the random walk follows a link according to distribution P instead of performing a random jump (usually $c = 0.85$). The $\pi = \pi\widetilde{P}$ represents a vector of PageRank scores for nodes of a given graph.

We can also show, that the final PageRank distribution π can be then defined as shown in (3), which directly follows from (2).

$$\pi = \frac{1 - c}{n} \sum_{k=0}^{\infty} c^k P^k \tag{3}$$

One of the first attempts to compute a local approximation of PageRank scores, namely (Fogaras et al., 2005), was based on the property presented in (3) which leads us to Monte Carlo methods. The authors in (Avrachenkov et al., 2007) proved that we can easily approximate PageRank values using random walks with restarts. Let the random walk start from a randomly chosen page and terminate with the probability $(1 - c)$. The random walk runs over the graph and makes the transitions according to transition matrix P with probability c. Let π_j be the final PageRank score for node j in the graph G. So, the initial PageRank formula (2) can be replaced with its rough estimate. We can show that using the properties of equation (3) we can also transform the Personalised PageRank formula and compute a rough estimate of its scores (Avrachenkov et al., 2010) with equation (4).

$$\hat{\pi_j}(s) = (1 - c)\frac{1}{m} \sum_{r=1}^{m} N_j(s, r) \tag{4}$$

A seed of initial nodes s is used to perform random walks over the graph, where c is the probability that the random walks terminates, m represents the overall number of required random walks, $N_j(s, r)$ is the number of random walks ending for node j, starting from seed node s in r-th random walk. The nodes representing a seed are usually randomly sampled from the graph.

This leads us to the following algorithm of PPR computation:

1. Simulate m runs of the random walks initiated at a node s.

2. Evaluate π_j as a fraction of m random walks which end at node $j \in \{1, 2, ..., n\}$.

The Monte Carlo approaches are based on random sampling, thus, we may obtain slightly different results for every run, especially when the number of the required iterations is too small to obtain faster progress of convergence giving a narrow confidence interval for estimated parameter. The unnecessary randomness can be avoided by initiating the random walk from each node in the same way, rather than jumping to random nodes over the graph. The accuracy of our estimate can be also improved by using variance reduction techniques e.g. by using Common Random Numbers approach (Clark, 1990).

4.2 Our Method

The initial idea was to approximate personalised PageRank value using random walk methods over semantic graphs, thus, we adopted the idea presented in (Fogaras et al., 2005) and (Pershina et al., 2015). The methods are usually based on random walk algorithm where the main assumption is that the local properties of senses are sufficient to accurately disambiguate the texts. This assumption allows us to reduce processing time of the algorithm when working with large lexical knowledge bases.

The underlying lexical knowledge base is a crucial element of a WSD method and it has a great impact on its performance. The algorithms and their properties presented in Sec. 4.1 were an inspiration for further work on disambiguation algorithms using large semantic networks.

The properties of personalised PageRank algorithm can be a limiting factor for WSD performance. The main issue is that in some specific cases a large node degree does not indicate a high significance, especially when the underlying knowledge base is a heterogenous graph constructed from different semantic resources (ontologies, dictionaries, wordnets). Adding or removing certain links can change the PageRank scores of the target nodes. The main problem is that not all of the links are correct, and to protect PageRank scores against incorrect links we can manipulate the importance of the links by using a heuristic weighting schema. We can also manipulate our seed to make the PPR algorithm more robust to textual noise. Since we are interested only in the importance of our personalisation nodes (representing senses of words for a given context), the initial seed for PPR computation is limited to a set of personalised nodes (usually a set of senses representing the words in a small textual window).

Simulate m random walks of length $L \sim Geom(p)$ starting from each seed node s. The seed consists of the nodes (synsets) representing all of available senses for the words from a given disambiguation context. The importance score γ_j for a given node j is the total number of visits to node j divided by the total number of visited nodes.

$$\hat{\gamma_j}(s) = \frac{1}{m}\sum_{r=1}^{m}(1-c)[N_j(s,r)+R_j(s,r)] \quad (5)$$

$$\hat{\gamma_j}(\boldsymbol{s}) = \frac{1}{m}\sum_{s\in\boldsymbol{s}}\sum_{r=1}^{m}(1-c)[N_j(s,r)+R_j(s,r)] \quad (6)$$

The parameter m denotes the overall number of performed random walks. The expected length of a single random walk is expressed as a geometric distribution $Geom(c)$, usually initialized with the value of parameter c (eq. (4)). To reduce the randomness effects we generate the lengths of walks only once, for all of our seed nodes, which is similar to the variance reduction with Common Random Numbers (Clark, 1990).

$R_j(s,r)$ represents the overall number of random resets, where the decision of a random jump to the starting node s is distributed according to $Bernoulli(p)$. The parameter p is dependent on the smoothed similarity score between usage example and a context.

We can compute a more accurate score estimate by using the recursive property of PageRank formula and averaging the scores of the neighbours. The recomputed final score for a single node j can be defined as:

$$\hat{\gamma_j}(\boldsymbol{s}) = (1-c)\frac{1}{|O(j)|}\sum_{v\in O(j)}\hat{\gamma_v}(\boldsymbol{s}) \quad (7)$$

In (7), the set $O(j)$ represents the neighbourhood of a single personalisation node j and $\hat{\gamma_v}(\boldsymbol{s})$ is a scoring function computed for neighbour v in $O(j)$.

In (Agirre and Soroa, 2009) and (Agirre et al., 2018) the authors proved that the sense frequency is a strong signal for accurate WSD. To make our methods comparable we decided to use the same source of sense frequency, namely SemCor corpus. Following the idea presented in (Agirre et al., 2018), the final score of a synset is computed as a linear combination of its normalised sense frequency and graph-based scores.

5 Evaluation

The proposed method incorporates several expansions to improve the overall performance of WSD. For the evaluation of WSD in English we utilised available semantic resources: i) manually disambiguated glosses, ii) synsets linked to Wikipedia, iii) the knowledge extracted from SemCor corpus.

In the case of Polish language the available semantic resources are limited. The glosses and usage examples for Polish senses were not disambiguated, thus, the synsets were linked only with the senses of monosemous words appearing in their glosses or usage examples. In the case of Wikipedia-based expansion we used a different mapping – the Polish synsets were partially linked to Wikipedia in a manual process (around 50,000 of synsets were mapped to Polish Wikipedia). We also translated the links by using interlingual synonymy links between Polish and English (enWordNet) parts of plWordNet.

5.1 Datasets

In the experimental part we evaluate our methods on the English dataset described in (Raganato et al., 2017) and the Polish dataset presented in (Kędzia et al., 2015). The former dataset consists of the five standard English texts prepared for Senseval and SemEval competitions for all-words WSD task. A sense inventory for the gold standard annotations was built on a basis of Princeton WordNet 3.0 which makes it approximately compatible with our knowledge-base, i.e. built on extended Princeton WordNet 3.1 in the case of English and plWordNet 3.2 (mapped to WordNet 3.1 and via it to SUMO) in the case of Polish. As we use WordNet 3.1 some small discrepancies can influence the results of the comparison with the test datasets, but they should not have significant impact on the outcome of the comparison.

Regarding the dataset for Polish only partially sense-annotated corpora exist. *Składnica* is a manually annotated dependency treebank with 13,035 sentences written Polish. The updated version of the semantic annotation in *Składnica* was based on plWordNet 3.2.

The *Polish Corpus of Wrocław University of Technology* (henceforth KPWr) consists of 1,127 documents manually annotated with the plWordNet 2.1 senses. The annotation was limited to a pre-selected set of words representing different cases of homonymy and polysemy. The original dataset consisted of 74 words in total, including 45 nouns and 29 verbs. The overall number of annotated word occurrences was 5,148 with 3,219 noun occurrences and 1,929 verb ones. Since there is a small inconsistency between senses used in the annotation and produced by WSD methods, i.e. coming from plWordNet 3.2, we decided to exclude from it 473 word occurrences for which the appropriate sense does not exist any more in our WSD model.

5.2 Experimental Setting

To accomplish a satisfactory convergence and obtain a small variance of final accuracy we adapted a following set of parameters for our experimental part: the transition probability $c = 0.3$, the overall number of random walks per node $rw\_iter = 1000$, the importance of sense frequencies on the final score $\alpha = 0.5$, and $(1-\alpha)$ for the importance of random walk-based scores. For a Polish dataset we did not use sense frequencies since there are no sense-tagged corpora available to compute the frequencies. The resultant performance (tables tab. 1 and tab. 2) was computed 10 times and averaged.

6 Results

Table 1. presents the final peformance for English dataset. The proposed method and introduced knowledge base expansions mostly outperformed a very strong most frequent sense (MSF) baseline. The method achieved the results being on the comparable level with other weakly supervised baseline methods, namely UKB (Agirre et al., 2018), Babelfy (Moro et al., 2014), and WoSeDon (Kędzia et al., 2015). Table 2. shows the average disambiguation time per context. As it was expected, the Monte Carlo approach provides better time efficiency (PPR vs PPRMC) and a comparable performance to power iteration method. The best results were achieved by mixing all available sources of semantic knowledge: the links extracted from disambiguated glosses, Wikipedia pages, SemCor texts and the links provided by SUMO ontology. The same tendency was observed for Polish dataset (table 3) – the best performance was noted for a mixed setting. The performance obtained for this dataset was also on a comparable level with the approaches based on power iteration method proposed in (Kędzia et al., 2015). PPRMC-1 uses a knowledge graph of synsets only. PPRMC-2 expands the model of

Method	*Sens-2*	*Sens-3*	*Sem-07*	*Sem-13*	*Sem-15*
MFS	66.80	66.20	55.20	63.00	67.80
Babelfy	67.00	63.50	51.60	66.40	**70.30**
UKB-nf	61.30	54.90	42.20	60.90	62.90
UKB-sf	67.50	**66.40**	54.10	64.00	67.80
UKB-nf-w2w	64.20	54.80	40.00	64.50	64.50
UKB-sf-w2w	68.80	66.10	53.00	**68.80**	70.30
PPRMC-1	66.26	64.28	54.06	65.08	67.12
PPRMC-2	66.35	65.13	55.60	65.56	66.63
PPRMC-3	66.47	65.94	56.04	65.26	67.71
PPRMC-4	66.78	66.28	**56.48**	65.90	68.10

Table 1: Averaged F1-scores of PPRMC for different knowledge base expansions: PPRMC-1: graph of synsets only, PPRMC-2: graph of synsets extended with links extracted from manually disambiguated glosses, PPRMC-3: PPRMC-2 extended with links extracted from SemCor and Wikipedia, PPRMC-4: PPRMC-3 with additional links extracted from SUMO ontology.

KB	*PPR*	*PPRMC*	*#nodes*	*#edges*
base	0.46	0.09	125,303	304,296
+gloss	0.59	0.12	125,303	659,860
+gloss_semcor_wiki	0.69	0.14	125,303	2,041,953
+gloss_semcor_wiki_sumo	0.73	0.16	152,966	2,158,986

Table 2: Average disambiguation time [s] per context for SemEval'15 dataset with respect to the size of underlying knowledge base. PPR settings: damping_factor=0.85, max_iterations=25, PPRMC settings: c=0.3, rw_count=1000. The disambiguation context was limited to a small window of three sentences.

Method	*Skład.-N*	*Skład.-V*	*KPWr-N*	*KPWr-V*
PPRMC-1	63.19	44.75	52.92	33.42
PPRMC-2	64.27	46.01	53.24	33.73
PPRMC-3	64.88	46.22	53.31	33.66
PPRMC-4	65.28	46.51	53.66	33.09
WoSeDon	63.92	46.43	53.61	33.71
WoSeDon	64.85	47.29	53.80	34.08
WoSeDon	65.27	47.55	54.02	34.00
WoSeDon	66.18	48.74	54.90	33.89

Table 3: Averaged precision of PPRMC for Polish datasets computed for nouns (N) and verbs (V) separately. A comparison with knowledge-based solution presented in (Kędzia et al., 2015) adapted to the structure of plWordNet 3.2.

PPRMC-1 with links to monosemous words extracted from glosses – in case of Polish glosses, or to disambiguated senses – in case of English glosses. PPRMC-3 adds the links to monosemous words extracted from Wikipedia, and PPRMC-4 introduces additional semantic links from SUMO ontology.

7 Conclusions

Weakly supervised Word Sense Disambiguation (WSD) methods express very good coverage that is only limited by the coverage of the underlying knowledge base used to define word senses and provide a basis for their disambiguation. However, very good results are obtained only when a rich and large knowledge base is utilised. In such a case, PPR-based WSD methods become slow that limits their applicability. We have shown that an estimation of the PPR algorithm on the basis of the Monte Carlo scheme can preserve most of the quality of the method while gaining a lot in terms of the efficiency of computation. We proposed a WSD tool[3] that achieves the performance comparable to the state-of-the-art among the weakly supervised methods, but it is 4-5 times faster. In addition, the efficiency of the proposed tool does not deteriorate so quickly with the increasing complexity of the knowledge base. The proposed method also offers an opportunity to balance between accuracy and processing speed by selecting the number of random walks.

[3] `gitlab.clarin-pl.eu/ajanz/wsd-mc`

References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for Knowledge-Based Word Sense Disambiguation. *Computational Linguistics* 40(1):57–84.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2018. The risk of sub-optimal use of open source nlp software: Ukb is inadvertently state-of-the-art in knowledge-based wsd. *arXiv preprint arXiv:1805.04277* .

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 33–41.

Konstantin Avrachenkov, Nelly Litvak, Danil Nemirovsky, and Natalia Osipova. 2007. Monte carlo methods in PageRank computation: When one iteration is sufficient. *SIAM Journal on Numerical Analysis* 45(2):890–904.

Konstantin Avrachenkov, Nelly Litvak, Danil A Nemirovsky, Elena Smirnova, and Marina Sokol. 2010. Monte carlo methods for top-k personalized pagerank lists and name disambiguation. *arXiv preprint arXiv:1008.3775* .

Dominik Baś, Bartosz Broda, and Maciej Piasecki. 2008. Towards word sense disambiguation of Polish. In *Proceedings of the International Multiconference on Computer Science and Information Technology — 3rd International Symposium Advances in Artificial Intelligence and Applications (AAIA'08)*. pages 65–71. http://www.proceedings2008.imcsit.org/pliks/162.pdf.

Pavel Berkhin. 2005. A survey on pagerank computing. *Internet Mathematics* 2(1):73–120.

Francis Bond and Ryan Foster. 2013. Linking and extending an Open Multilingual Wordnet. In *51st Annual Meeting of the Association for Computational Linguistics*. ACL, pages 1352–1362.

Bartosz Broda and Maciej Piasecki. 2011. Evaluating LexCSD in a large scale experiment. *Control and Cybernetics* 40(2):419–436.

Gordon M Clark. 1990. Use of common random numbers in comparing alternatives. In *Proceedings of the 22nd conference on Winter simulation*. IEEE Press, pages 367–371.

Christiane Fellbaum, editor. 1998. *WordNet – An Electronic Lexical Database*. The MIT Press.

Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. 2005. Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. *Internet Mathematics* 2(3):333–358.

Sanda M. Harabagiu, George A. Miller, and Dan I. Moldovan. 1999. Wordnet 2 - a morphologically and semantically enhanced resource. In *Proceedings of SIGLEX 1999*.

Paweł Kędzia and Maciej Piasecki. 2014. Ruled-based, interlingual motivated mapping of plwordnet onto sumo ontology. In Nicoletta Calzolari et al., editor, *Proc. Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association.

Paweł Kędzia, Maciej Piasecki, Jan Kocoń, and Agnieszka Indyka-Piasecka. 2014. Distributionally extended network-based Word Sense Disambiguation in semantic clustering of Polish texts. *IERI Procedia* 10(Complete):38–44. https://doi.org/10.1016/j.ieri.2014.09.073.

Paweł Kędzia, Maciej Piasecki, and Marlena Orlińska. 2015. Word sense disambiguation based on large scale polish clarin heterogeneous lexical resources. *Cognitive Studies* (15).

Karin Kippera, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extensive classifications of english verbs. In *Proceedings of the 12th EURALEX International Congress*.

Marek Maziarz, Maciej Piasecki, Ewa Rudnicka, Stan Szpakowicz, and Paweł Kędzia. 2016. plwordnet 3.0 – a comprehensive lexical-semantic resource. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. ACL, ACL, pages 2259–2268. **ACL Anthology**. http://www.aclweb.org/anthology/C16-1213.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. PageRank on semantic networks, with application to Word Sense Disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '04. https://doi.org/10.3115/1220355.1220517.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics* 2:231–244.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193:217–250.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper.

Adam Pease. 2011. *Ontology - A Practical Guide*. Articulate Software Press.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 238–243.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. pages 99–110.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*.

Mark Stevenson, Eneko Agirre, and Aitor Soroa. 2012. Exploiting domain information for Word Sense Disambiguation of medical documents. *JAMIA* 19(2):235–240.

Classification of Micro-Texts Using Sub-Word Embeddings

Mihir Joshi
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
mihir@dal.ca

Nur Zincir-Heywood
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
zincir@cs.dal.ca

Abstract

Extracting features and writing styles from short text messages is always a challenge. Short messages, like tweets, do not have enough data to perform statistical authorship attribution. Besides, the vocabulary used in these texts is sometimes improvised or misspelled. Therefore, in this paper, we propose combining four feature extraction techniques namely character n-grams, word n-grams, Flexible Patterns and a new sub-word embedding using the skip-gram model. Our system uses a Multi-Layer Perceptron to utilize these features from tweets to analyze short text messages. This proposed system achieves 85% accuracy, which is a considerable improvement over previous systems.

1 Introduction

One of the most challenging problems in text analysis is identifying the author of a micro-text, i.e. short text messages. Most of the data created on social media applications whether a Tweet, Facebook comment or text on a messaging application is a micro text. A micro or short text message could be a tweet or a comment which is around 140 characters or less. In general, text analysis and natural language processing approaches employ statistical feature extraction techniques such as term frequency, inverse document frequency and a bag of words. Due to the feature extraction process being statistical in nature, all these techniques require a certain amount of data to use them effectively for determining patterns or perform authorship attribution. Thus, having short text messages such as tweets which is around 140 characters or less makes it difficult to identify the patterns on a given text and make predictions about the author.

Shrestha et al. (2017) used a convolutional neural network (CNN) architecture using character embeddings instead of word embeddings for short texts. With this approach they showed less than five percent improvement on previous researches (Qian et al., 2015; Schwartz et al., 2013) in this area. In this paper, we start by implementing a simpler approach; combining the two consecutive records from the same author to train our model and then apply the feature known as Flexible patterns (Schwartz et al., 2013) after modifying the existing method and finally introducing our new feature based on the word vector approach (Bojanowski et al., 2017).

The paper is organized in the following sections. Section 2 represents related works. In section 3 we define our model architecture followed by the dataset and the feature extraction techniques. Section 4 shows the results of our approach compared with previous works. Lastly, in section 5, we discuss conclusion and future work.

2 Related Works

In short-text analysis, one of the earlier works by Layton et al. (2010) aims to identify the author based on the data collected from micro-blogging websites like Twitter. The authors create author profiles using character level n-grams. They find the frequency of the most common n-grams in an author profile and assume that text from the same author would have a similar pattern. This approach is further extended by Schwartz et al. (2013); they use two more features namely word n-grams and a Hyponyms acquisition technique (Hearst, 1992) called Flexible patterns along with character n-grams. They then input a combination of these features into a linear SVM and ten-fold cross validation is applied to evaluate the model.

Proceedings of Recent Advances in Natural Language Processing, pages 526–533,
Varna, Bulgaria, Sep 2–4 2019.

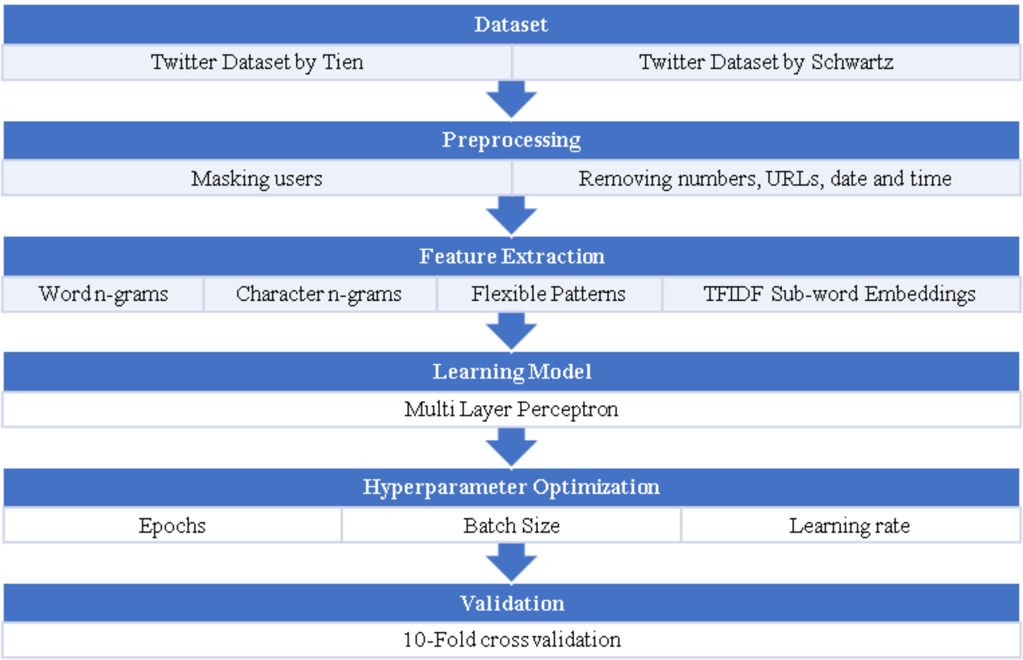


Figure 1: Overview of the proposed system

One of the first deep learning approaches by Rhodes (2015) used word embeddings and a convolutional neural network. Skip-gram method with negative sampling is used for word vectorization from the Google news dataset while using random initialization for unseen words. Furthermore, the convolutional neural network was based on a similar model (Collobert et al., 2011) for language processing, where the activation function is changed to rectified linear units and introducing dropouts. Finally, it demonstrated that the system performs well on longer text sequences.

Shrestha et al. (2017) makes use of a similar architecture; a convolutional neural network (CNN) using character embeddings instead of word embeddings for short texts. The CNN model takes character unigram or bigram as an input that passes through the character embedding layer before feeding it to the convolutional layer. Similar datasets and evaluation criteria as Schwartz et al. (2013) are used to determine the performance of the system. This approach increases the overall accuracy to approximately 76%. Even though the results are better than all previous researches there is a scope for further improvement.

A more recent study on this field (Phan and Zincir-Heywood, 2018) used word embeddings and the neural network to identify the authors of short text messages. They used the three different datasets namely Reuters Corpora (RCVI) (Lewis et al., 2004), Enron dataset (Klimt and Yang, 2004), and Twitter dataset (Yilu et al., 2016). The RCVI is used to train the embeddings which in turn generated the high-level features from the other two corpora by concatenating the vector of means and standard deviations. They then used the feed forward neural network to identify the authors.Therefor, in this work, we also used the same five authors (*Ashley_Nunn75, bradshaw1984, shawnevans81 , terrymarvin63, and WhieRose65*) from the twitter dataset that they used to test our proposed system.

3 Methodology

In this section, we describe our proposed approach and the features we used to train our system. Figure 1 shows an overview of our system in which we worked on two datasets: one by Schwartz et al. (2013) and the other one by Yilu et al. (2016). Section 3.1 explains more about the data.

We carried our pre-processing on the data before preforming feature extraction. There are a total of four features that are obtained from the text namely word n-grams, character n-grams, flexible patterns and word embeddings. An n-gram (section 3.2) is a sequence of n words where n is a positive integer. For example, in the phrase - *”This is a sentence”* -, a word unigram would be - *”This”, ”is”, ”a”* - and a word bigram would be *”this is”, ”is a”, ”a sentence”* - and so on. On the other hand, character n-grams are similar to word n-grams except, they are a sequence of characters. For instance, if we use the previous example, ”This is a sentence”, a character unigram would be - *”T”, ”h”, ”i”, ”s”* - and a character bigram would be - *”th”, ”hi”, ”is”*.

In section 3.3 we define Flexible patterns, explain the existing method and our new approach to create them. The idea behind flexible patterns is that some users tend to use the same sequence of words in their writing style and only change a few keywords called content words (CW). For example, the flexible pattern of the following phrases; *”I read the paper today”* and *”I drove the car yesterday”* is *”I CW the CW CW”*. The words *read, paper, today, drove, car* and *yesterday* are replaced by the word *CW* based on the pre-defined condition. Therefore, masking some words, would create a pattern and separate the texts of some users from other users. We modified the existing approach (Schwartz et al., 2013) to make it suitable for smaller datasets and also to make it easier to implement.

Next, we talked about the word embeddings (Mikolov et al., 2013) feature set in section 3.4. We used the skip-gram technique to create the 300-dimensional vector representation of our data and used that to create the embeddings by combining them using the weights obtained by using TF-IDF.

Later we analyzed (section 3.5) all the models we used and the reason for choosing a Multi-Layer Perceptron (MLP) architecture (Gillian, 2014). We also explained the architecture of our MLP model and the parameters we used to further optimize. Lastly, in section 4 we will compare the results of our approach to both datasets.

3.1 Datasets

To compare our results with the previously mentioned approaches, we used the same dataset that Schwartz et al. (2013) used. The dataset contains a total of 7000 authors, out of which we selected 50 authors at random, where each author has 1000 tweets. We masked the username (@user), numbers, links, date and time from the texts to minimize the bias and noise in the data. We also changed all of the letter characters to lowercase and employ word stemming to reduce the size of the vocabulary used.

Before extracting features and feeding data into the MLP, we concatenated sets of two adjacent records. For example, the first text is combined with the second, the third with the fourth and so forth. Though the initial number of records remains the same, combining the original texts enables us to have a larger sequence, which achieves better accuracy even with the earlier approaches of feature extraction (Schwartz et al., 2013; Shrestha et al., 2017). The larger sequence also helps us to achieve more meaningful patterns which is not otherwise possible.

We also applied the above approach on a different dataset (Yilu et al., 2016) used in Phan and Zincir-Heywood (2018). We used the same 5 authors (users) as they did, with 2000 tweets each. In that paper (Phan and Zincir-Heywood, 2018), the names of the tagged users in a tweet were not masked. However, we masked them, i.e. whenever we encounter a tagged user - *@user* - we change it to a specific word preventing our system from overfitting.

3.2 Word and Character N-Grams

Both word and character n-grams were extracted from the datasets used. For the value of *n* in n-grams and the minimum occurrence of each n-gram, we used the same parameters as used by Schwartz et al. (2013) for comparison purposes. We also took into consideration the maximum occurrence of the n-grams. For example, a pattern including prepositions (as shown in 1 below) or a masked username (as shown in 2 below) are very common in tweets.

1. *for a*
2. *<User> I*

This is even more common in character n-grams. For this reason, we kept the upper limit of the n-grams to 0.9, which means we do not consider any word or character n-grams that appear in more than 90 percent of the documents. This further helps us identify the unique writing style of an author.

We restricted each feature to a maximum of 50,000 in our experiments where each author had 1000 tweets. To assign weights to the n-grams, we used the term frequency-inverse document frequency (TF-IDF) weighted scheme.

TF-IDF (Manning et al., 2008) calculates the importance of the word based on how frequently it appeared in a text, which was then balanced by the frequency of the word in the entire corpus.

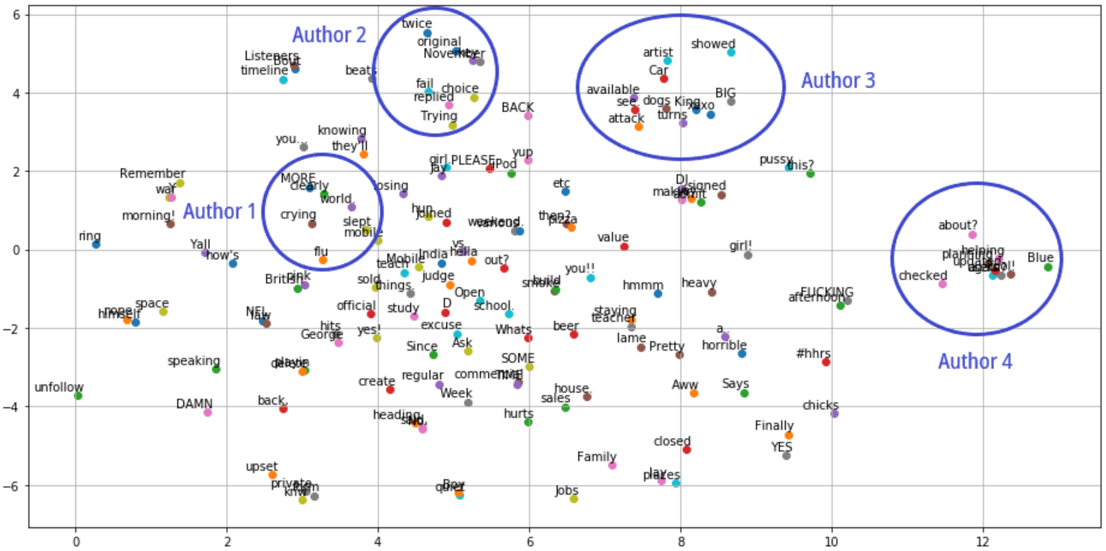


Figure 2: Skip-Gram word embeddings

In a text document d, the frequency of term t is: $tf_{t,d} = f_{t,d}$. This shows the total number of times (frequency count) the term occurs in the document, whereas the inverse document frequency is: $idf(_{t,d}) = log\frac{N}{t \in D}$, where $\hat{N}$ is the total number of documents D in the corpus. Moreover, we employed sub-linear scaling to the TF-IDF by taking the log of the term frequency, Eq.1.

$$wf_{t,d} = \begin{cases} 1 + logtf_{td}, & \text{if } tf_{td} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

3.3 Flexible Patterns

Flexible patterns are a branch of word n-grams, where each word is either a high-frequency word or a content word and some words can be both (Schwartz et al., 2013). For a corpus size s if a word appears more than $10^{-4} \times s$ times it is a High-Frequency Word (HFW) and if it appears less than $10^{-3} \times s$ times it is a Common Word (CW). Also, the previous method takes into consideration that flexible patterns start/end with an HFW and there can be no consecutive HFWs.

The problem with this approach is that it does not work with a smaller corpus. For a corpus where the value of s is 1000, no word is a CW. Thus, to overcome this limitation, Eq.2 is used to calculate the CW for a bigger corpus. Therefore in a corpus with a vocabulary size n, the CW is calculated as the common log of the threshold for CW is selected as twice of log of s which are total number of words in the training dataset vocabulary. This method is based on the various experiments we carried out to choose the optimum number.

$$CW \leqslant 2 \times log_{10}n \tag{2}$$

After replacing all the CWs in the corpus, we then used the same approach as we did for word n-grams. Then we applied the TF-IDF weighted scheme to those flexible n-grams before inputting them into the model.

3.4 TF-IDF Weighted Word Embeddings

Word embedding (Mikolov et al., 2013) is a semantic parsing technique used to create the vector representation of a text in a smaller dimensional space compared to the classic Bag of words approach (Zhang et al., 2010). The idea behind word embedding is that semantically similar words should be close to each other. That is, in an n-dimensional space, the angle between the similar words should be close to zero. There are two types of methods to achieve this; namely, Continuous Bag of Words (CBOW) and Skip-Gram methods (Mikolov et al., 2013). While CBOW predicts the target words by taking the context word as input, Skip-gram predicts the probability of the context words using the target words.

We used the word embedding approach from Bojanowski et al. (2017), which is based on the assumption that tweets contain several words, including, but not limited to, hashtags that are rare and sparsely occur in a corpus. To address this sparsity issue, each word is represented by the sum of the vector representations of its n-grams. Having a dictionary as size D and a word w belongs to this dictionary having $D_w \subset \{1, ..., D\}$, the set of n-grams appearing in w. Associating a vector representation of z_d to each n-gram d, the scoring function for w can be represented by Eq.3:

$$s(w,c) = \sum_{d \in D_w} z_d^T v_c \tag{3}$$

For the value of n, we chose $2 \leqslant n \leqslant 6$. Empirically, instead of using pre-trained embedding, we trained it on our corpus with a 300-dimension vector space. We constructed the embedding using the Skip Gram approach which works better for a smaller amount of data and is preferable when there are a greater number of rare words in the corpus (Schwartz et al., 2013). Figure 2 shows the created vector representation from the dataset.

Using t-distributed Stochastic Neighbor Embedding technique (Maaten and Hinton, 2008), we visualized the 300-dimensional data in a two-dimensional space (Figure 2). This shows that words which are used in the same context are grouped closer to each other. The idea behind this approach is that a particular author would use the same combination of words in his/her tweets and therefore, they should be close to each other.

Before using this as an input feature for our model, we weighted the embedding of each word in a text with the IDF value of that word. The resulting dimension is the same as the dimension of each word, which in our case is 300. Ultimately, we calculated the mean of the words to keep the dimensionality of the entire text the same as our vector space. Having a text of size T and words w where $T_w \subset \{1, ..., T\}$, the TF-IDF weighted embedding feature $f(w)$ of a word w_T is given by using Eq.4:

$$f(w) = \frac{\sum_{w \in T_w} idf(w_T) \times emb(w_T)}{T} \tag{4}$$

4 Experiments and Results

As shown in Figure 1, four different machine learning algorithms are employed at the learning phase of the proposed system in this work. We implemented SVM, Naive Bayes, Random Forest (Decision Trees) and MLP classifiers using the Scikit-Learn Python machine learning library (Pedregosa et al., 2011). As discussed earlier, the goal is to classify micro-text. In doing so, we aim to choose the most suitable classifier using the extracted features proposed in the previous section. To this end, we employed the same small dataset used in Phan and Zincir-Heywood (2018). Table 1 shows the 10-fold cross validation accuracy of the four classifiers on that dataset for the 5 authors previously mentioned, where 2000 tweets are used for each author. In this case, our system (Table 1) achieves more than 99% 10-fold cross-validation accuracy which is 15% more than the best result reported in Phan and Zincir-Heywood (2018). Moreover, Figure 3 (confusion matrix) demonstrates that these results are not biased to any specific author in the dataset used. Additionally, Table 2 shows how the 10-fold cross-validation accuracy improved the proposed MLP classifier as the different feature extraction techniques are used, where the first three columns show the accuracy of the combination of extraction techniques and the last column is the best test results given in Phan and Zincir-Heywood (2018).

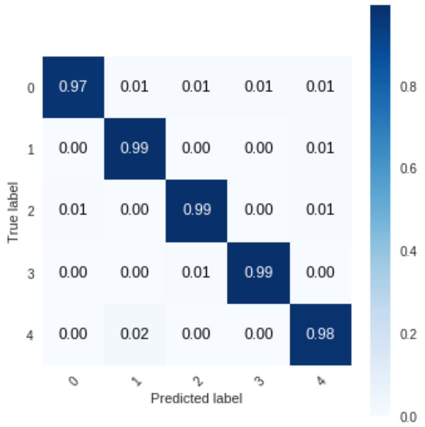


Figure 3: Confusion matrix for 5 authors

MLP	SVM	Naive Bayes	Random Forest
.99	.979	.96	.82

Table 1: Accuracy for 5 users with 2000 tweets using different classifiers having combination of same four features

Given the above observations, MLP was chosen as the most suitable classifier for our research purpose: classifying the tweets according to their

TFIDF Emb.	Mod. flex	Joining rec.	Phan2018
.99	.983	.972	.841

Table 2: Accuracy for 5 users with 2000 tweets each using proposed system with different feature sets

authors. Figure 4 presents the overview of the proposed MLP model. The number of nodes in the input layer depend on the size of the input feature. *None* represents that the dimension is variable, and in this case, it is *86121*. Then, there is a dense layer, which is a fully connected layer, where each input node is connected to each output node. In the proposed model, there are two Dense layers. One is a hidden layer of 1000 nodes and the other is the output layer of 50 nodes, which corresponds to 50 authors. This can be changed depending on the number of authors (output classes). Inbetween the two dense layers, there is a dropout layer for regularization.

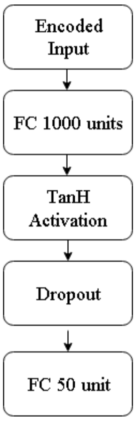


Figure 4: Overview of the proposed model using a Multi-Layer Perceptron as the classifier with one hidden layer and a dropout layer

The input layer of the MLP model depends on the number of features, which increases as the number of tweets increases to train the model. There is only one hidden layer with 1000 nodes. It should be noted, that increasing the hidden layers did not improve the validation accuracy in our experiments. The Tanh activation function (Weisstein, 2002) is used for the hidden layer. The last layer consists of the same number of nodes as the number of authors where the SoftMax activation function (Nwankpa et al., 2018) is used. Furthermore, we have a 30% dropout after the hidden layer to prevent overfitting. In the proposed system, ADAM (Kingma and Ba, 2014) is adapted as the optimization algorithm with a learning rate of 0.001. We divided the data into batches of 64 and trained our model for a maximum of 40 epochs. Figure 5 and Figure 6 show the number of epochs vs loss, and the number of epochs vs accuracy graphs for training and validation data, respectively. These results show that both the loss and accuracy are optimal at around 40 epochs.

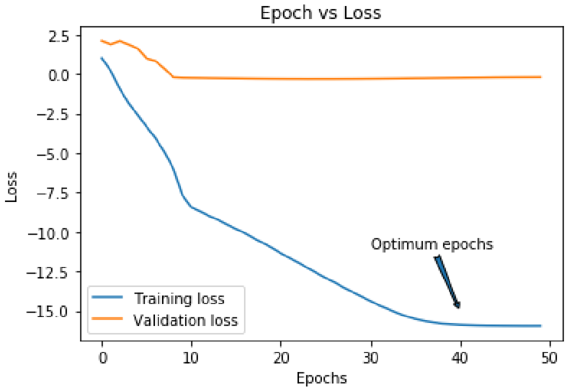


Figure 5: Epochs vs Loss for 50 epochs

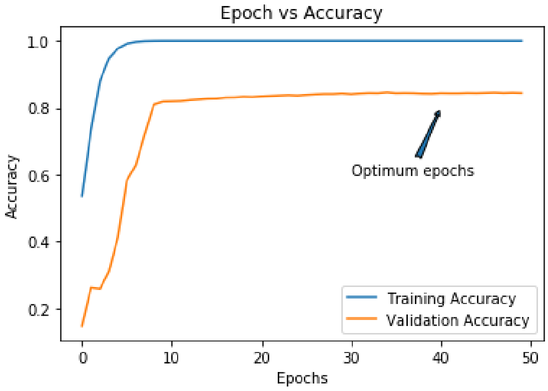


Figure 6: Epochs vs Accuracy for 50 epochs

TFIDF Emb.	Flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
.852	.829	.81	.761	.712	.703	.645

Table 3: Accuracy for 50 users with 1000 tweets each

In the following experiments, we incrementally applied all three feature extraction techniques using the MLP classifier and observed the improvements compared to the previous research results (see Section 2). To this end, we first combined the subsequent tweets and applied the approach presented by Schwartz et al. (2013), then we improved that method to calculate flexible patterns and their effect on the accuracy of the system. Last but not the least, we built weighted TF-IDF embeddings and combined them with the improved flexible patterns to form the combined set of features as input into the proposed MLP model. Ten-fold cross-validation approach was used to evaluate the performance of the proposed system. We evaluated the proposed system on the same dataset as used by Schwartz et al. (2013); Shrestha et al. (2017), where the dataset consisted of 50 authors, each having 1000 tweets.

Columns 1-3, in Table 3, shows the results of all three feature extraction techniques used with

#	TFIDF Emb.	Mod. flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
500	**.791**	.76	.748	.724	.672	.655	.597
200	**.730**	.694	.679	.665	.614	.585	.528
100	**.648**	.619	.608	.617	.565	.517	.438
50	**.589**	.563	.53	.562	.505	.466	.364

Table 4: Accuracy for 50 users from 500 to 50 tweets for each

the MLP, and columns 4-7 represents the results from the previous works on the same dataset. Our results are mutually inclusive, and the results are built upon combining all the feature extraction techniques. For example, we used the weighted TF-IDF embeddings in combination with the flexible patterns.

CNN-C, shown in Table 3, represents the best result obtained by Shrestha et al. (2017) where a convolutional neural network architecture is proposed using character n-grams, specifically unigrams and bigrams as input. The convolutional model used was a three-level architecture with the input layer as the character embedding layer, a convolutional module, and finally, a dense layer with a Softmax activation function for classification. The unigram model performed well on the smaller dataset. Alternatively, the bigram model had better accuracy on the bigger dataset.

SCH, shown in Table 3, represents the best result obtained by Schwartz et al. (2013). They used a linear SVM for classification and their model was a combination of word and character n-grams along with a new feature set called flexible patterns (see section 3.3), which is modified and used in our system as well.

Char, shown in Table 3, represents one of the systems used by Shrestha et al. (2017) in which they compared the performance of their system based on the earlier character n-gram approaches (Layton et al., 2010; Schwartz et al., 2013) and proposed a logistic regression model that employed character n-grams of sizes two to four.

LSTM-2, shown in Table 3, represents the state-of-the-art LSTM model based on the success of previous implementations (Tai et al., 2015; Tang et al., 2015). This model was also used with bigrams as input to evaluate the performance of those systems, with respect to other models on the same dataset.

Introducing the concatenation of the consecutive records enables the accuracy of the proposed system to be improved by 5% compared to the previous approaches. Then applying the flexible patterns, further improved the accuracy by approximately 2%. Finally, implementing the weighted TF-IDF word embedding, and combining it with all the other features increases the accuracy of the proposed system to approximately 85%, Table 3.

In Table 4, we also compared the proposed system's accuracy to other approaches as we reduced the number of tweets from 500 to 50 for each author (50). After reducing the number of tweets, the proposed system still outperformed all the other previous approaches and performed well even when the dataset became as small as 50 tweets per author.

5 Conclusion and Future Work

In this paper, we explored a feature extraction technique that was based on a word embedding model weighted by TF-IDF. We also worked on modifying and improving the existing implementation of flexible patterns and proposed a neural network architecture that makes use of a combination of these features to perform authorship attribution. Our model outperformed all the existing systems based on similar testing criteria and using the same datasets. Since we trained our embeddings from scratch our system performs equally well, irrespective of the language.

With the success of word embeddings, we look forward to working upon new word embedding techniques such as Elmo (Peters et al., 2018) and Bert (Devlin et al., 2018), which are based on the context of a word in a corpus. Alongside that, we are also interested to improve our neural network architecture using transfer learning models such as ULMFit (Howard and Ruder, 2018).

6 Acknowledgments

This research is supported by the Natural Science and Engineering Research Council of Canada. This research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Nick Gillian. 2014. Mlp nickgillianwiki. http://www.nickgillian.com/wiki/pmwiki.php/GRT/MLP. (Accessed on 04/25/2019).

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 539–545.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, pages 217–226.

Robert Layton, Paul Watters, and Richard Dazeley. 2010. Authorship attribution for twitter in 140 characters or less. In *2010 Second Cybercrime and Trustworthy Computing Workshop*. IEEE, pages 1–8.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK. http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. 2018. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378* .

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research* 12:2825–2830.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Tien D Phan and Nur Zincir-Heywood. 2018. User identification via neural network based language models. *International Journal of Network Management* page https://doi.org/10.1002/nem.2049.

Tie-Yun Qian, Bing Liu, Qing Li, and Jianfeng Si. 2015. Review authorship attribution in a similarity space. *Journal of Computer Science and Technology* 30(1):200–213.

Dylan Rhodes. 2015. Author attribution with cnns cs224.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1880–1891.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. volume 2, pages 669–674.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. pages 1422–1432.

Eric W Weisstein. 2002. Inverse hyperbolic tangent .

Zhou Yilu, Alsarkal Yaqoub, and Zhang Nan. 2016. Linking virtual and real-world identities twitter dataset.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics* 1(1-4):43–52.

Using Syntax to Resolve NPE in English

Payal Khullar **Allen Antony** **Manish Shrivastava**
Language Technologies Research Centre
International Institute of Information Technology, Hyderabad
Gachibowli, Hyderabad, Telangana-500032
{payal.khullar@research., allen.antony@research.,
m.shrivastava@}iiit.ac.in

Abstract

This paper describes a novel, syntax-based system for automatic detection and resolution of Noun Phrase Ellipsis (NPE) in English. The system takes in free input English text, detects the site of nominal elision, and if present, selects potential antecedent candidates. The rules are built using the syntactic information on ellipsis and its antecedent discussed in previous theoretical linguistics literature on NPE. Additionally, we prepare a curated dataset of 337 sentences from well-known, reliable sources, containing positive and negative samples of NPE. We split this dataset into two parts, and use one part to refine our rules and the other to test the performance of our final system. We get an F1-score of 76.47% for detection and 70.27% for NPE resolution on the testset. To the best of our knowledge, ours is the first system that detects and resolves NPE in English. The curated dataset used for this task, albeit small, covers a wide variety of NPE cases and will be made public for future work.

1 Introduction

Nominal Ellipsis or Noun Phrase Ellipsis (NPE, henceforth) is a type of ellipsis in linguists wherein the sub-parts of a nominal projection are elided, with the remaining projection pronounced in the overt syntax. For example in the sentence presented in (1), the noun *chairs* is elided at the position marked by [e].

1. There are three **chairs** in the living room and two [e] in the hall.

The full meaning of such a sentence can only be understood when we reconstruct the meaning of the elided part from the antecedent, which can be present in the linguistic context as in (1) or has to be retrieved from real world knowledge as in (2), where *Mary's* actually means *Mary's place*.

2. We are all partying at Mary's [e].

All world languages use some or the other mechanism to elide redundant information and, hence, ellipses is fairly pervasive in natural language, more so in conversational settings (Langacker, 1999). While human interlocutors effectively resolve and disambiguate any elided information in a sentence based on context and cognitive commonsense extension (Chen, 2016), the realization of the complexity of processing of ellipsis becomes evident when it poses a serious challenge for computational systems involved in natural language understanding.

The approaches that handle ellipsis in both theoretical linguistics and NLP are largely classified as syntactic, semantic and pragmatic (Merchant, 2010). For the current paper, we present a system that automatically detects and resolves NPE in English using a syntax-driven approach.

2 Previous Work

The syntax and semantics of the ellipsis phenomenon has been thoroughly studied in theoretical linguistics (Halliday and Hasan, 1976; Dalrymple et al., 1991; Lobeck, 1995; Lappin, 1996; Hardt, 1999; Johnson, 2001; Merchant, 2004; Frazier, 2008; Chung et al., 2010; Merchant, 2010; Rouveret, 2012; Gunther, 2011; van Craenenbroeck and Merchant, 2013; Park, 2017), in cognitive linguistics (Kim et al., 2019), and in language acquisition studies (Hyams et al., 2017; Lindenbergh et al., 2015; Goksun et al., 2010; Wijnen et al., 2003). In the context of NLP, most of the work on handling ellipsis has been done on Verb Phrase Ellipsis (VPE) and related phenomenon, for instance the preparation of annotated corpus for analysis of VPE (Bos and Spenader, 2011), the detection of VPE in Penn treebank (Hardt, 1997), the domain independent detection and resolution

Proceedings of Recent Advances in Natural Language Processing, pages 534–540,
Varna, Bulgaria, Sep 2–4 2019.

of VPE using machine learning (Nielsen, 2003) and parsed text (Nielsen, 2004), using linguistic principles (McShane and Babkin, 2016), with sentence trimming methods (McShane et al., 2015), reconstruction of sentences with gapping using improved parsing techniques that encode elided material dependencies (Schuster et al., 2018), etc.

There are no known systems that handle NPE detection and resolution in English. However, on a related linguistic phenomenon called one-anaphora or one-substitution, in which the elided noun is replaced by an overt pro-form, there is a thorough data-driven investigation (Gardiner, 2003) and machine-learning methods that use heuristics proposed in this study (Ng et al., 2005). Another phenomenon similar to NPE is zero-anaphora, which has been thoroughly studied in some pro drop languages such as Chinese (Yeh and Chen, 2019a,b) and Japanese (Iida et al., 2007; Asao et al., 2018; Chen, 2016). Zero-anaphora does not occur in English, although there is some evidence of the phenomenon being used to achieve certain interactional functions in ordinary conversational settings by English speakers (Oh, 2005). There are also proposed heuristics for determining antecedents of pronominal words (Lappin and Leass, 1994; Kennedy and Boguraev, 1996). In the present paper, we do not deal with one-anaphora, zero anaphora or pronominals, and restrict our focus to NPE.

3 Task Description

Resolution of ellipsis comprises two tasks - detection of the elided material and antecedent selection. In some cases, reference resolution might also be necessary (Liu et al., 2016; Nielsen, 2003). For example, in (3), the common sense interpretation is that Sam loves his girlfriend. But it could also lead to a sloppy reading where it means Sam loves John's girlfriend.

3. John **loves his girlfriend**. Sam does [e] too.

Note that (3) presents an example of VPE as the verb along with its predicate are elided. In this paper, we focus only on the first two tasks, i.e. detection of NPE and antecedent selection.

4 Dataset Preparation

There are no dedicated linguistic resources or datasets for the analysis of NPE in English. However, there are many well-known corpora that contain annotated instances of NPE. One such resource is the Universal Dependency (UD) tree-

No.	Syntactic Category	Examples
	Can License NPE	
1.	Cardinal Numbers	I read three **chapters** from this book and Mary read [$_{NP}$ four [e]].
2.	Ordinal Numbers	Mary got first **position** in the university and John got [$_{NP}$ second [e]].
3.	Demonstrative Determiners (Plural)	Of all the **candidates** that applied for the job, [$_{NP}$ these [e]] got selected.
4.	Quantifiers (Not all)	Some **students** love physics and [$_{NP}$ some [e]] don't.
5.	Superlative Adjectives	He is the funniest **guy** here. And also [$_{NP}$ the weirdest [e]].
6.	Noun Possessives	That big **car** standing over there is [$_{NP}$ Joey's [e].
7.	Pronoun Possessives	John is reading my **book** and I am reading [$_{NP}$ his [e]].
8.	Interrogative Determiners	I don't know which **pages** to read and [$_{NP}$ which [e]] to ignore.
	Cannot License NPE	
1.	Adjectives	* I have a big **house** and she has [$_{NP}$ a small [e]].
2.	Demonstrative Determiners (Singular)	* That **pen** belongs to Mary and [$_{NP}$ this [e]] belongs to John.
3.	Articles	* I really liked that **house** and I know you liked [$_{NP}$ the [e]] too.
4.	Quantifiers (Not all)	* There are 50 **students** in my class and [$_{NP}$ every [e]] went to see the movie.

Table 1: Syntactic categories that can and cannot license NPE in English, with examples for each category.

bank (Silveira et al., 2014) for English that contains example sentences for different types of ellipsis such as VPE, NPE, etc. The UD treebank marks NPE by raising the dependents of the elided noun to the position of head in cases where the dependents are overtly marked. Through a simple search for noun dependents that are given the status of noun heads, we get a total of 146 cases of NPE in 120 sentences from the UD treebank. There is another comparatively small corpus called the ParCorFull: a Parallel Corpus Annotated with Full Coreference (Lapshinova-Koltunski et al., 2018) that is dedicated to anaphora. This corpus targets anaphora, but deals partly with NPE cases as well, marking them with a nom-ellipsis tag. A simple search for this tag gives us 5 sentences containing 5 NPE cases. We also pick a total of 80 sentences containing 83 cases of NPE from linguistic textbooks on ellipsis (Lobeck, 1995; Saito et al., 2008; Menzel, 2017; Kim et al., 2019; Corver and van Koppen, 2011) to cover even the infrequently occurring cases. Finally, we randomly pick 132 sentences that do not contain NPE from UD treebank, ParCOrFull and the same linguistic textbooks. Some of these negative samples of NPE contain sentences with ellipsis other than NPE, such as VPE.

In total, we curate a small dataset of 337 sentences, of which 205 sentences have 234 instances of NPE (some sentences contain more than one instance of NPE) and the remaining 132 sentences without NPE. This dataset, albeit small, covers a wide variety of the cases of NPE discussed in the ellipsis literature and will be useful for future work. Since the focus of this paper is on presenting a system for detection and resolution of ellipsis, we do not undertake the formidable task of preparation of annotation guidelines and performing annotation to prepare a Gold dataset in this paper. However, this could be an important future work considering the limited available resources for the analysis of NPE.

To fine tune our rules, we only need positive samples of NPE. However, for testing, it is important to use both positive and negative samples. We split our dataset into two parts. For fine tuning the rules, we randomly pick 140 out of the 205 positive sentences (roughly 70%), containing a total of 158 NPE instances. The remaining 65 sentences (roughly 30%) contain 76 NPE instances and are included in the testset along with the 132 sentences without NPE. Hence, our testset has 76 positive and 132 negative samples.

5 System Overview

Our system is divided into two parts. An input sentence is fed into the NPE detection system that decides whether an NPE is present or not. If an NPE is detected, it sends the sentence to the Resolution system where a potential antecedent is selected. The output of the complete system is either a decision that there is no ellipsis present in the sentence or an ellipsis site marked along with its antecedent.

5.1 NPE Detection

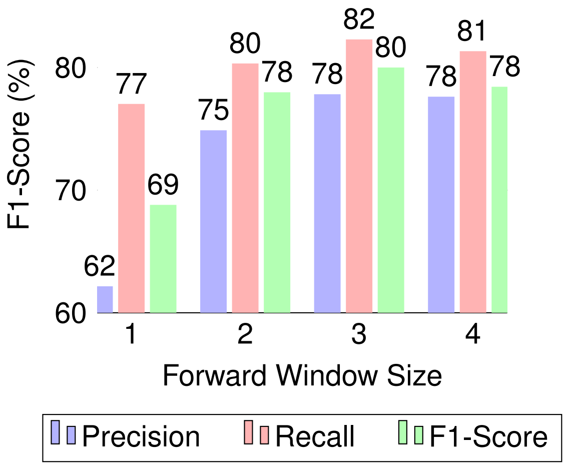
For the task of NPE detection, we exploit a very useful syntactic feature, which is the presence of overt remnants of the noun phrase at the ellipsis site. In case of NPE in English, these trigger words are often determiners and modifiers of the elided noun. These are also known as licensors of ellipsis. In the examples presented in (1) and (2), the licensors of elided noun are the cardinal number *two* and possessive proper noun *Mary's* respectively. We use these remnants or noun modifiers present at the ellipsis site as cues to locate the elided noun.

An interesting feature about these license of NPE in English is that they can only belong to certain syntactic categories. These include cardinal and ordinal numbers, plural demonstrative determiners, possessives, adjectives, quantifiers, and a certain types of determiners. Table 1 provides examples from each of these categories, along with examples of syntactic categories that cannot license NPEs in English. Hence, the idea is to use the syntactic environment of the nominal ellipsis site to perform detection. Linguistically, our approach is similar to detection of VPE by using auxiliary and modal verbs as cues (McShane and Babkin, 2016), as VPE in English are licensed by auxiliary and modal verbs.

5.1.1 Look for Pre-Modifiers and Determiners

NPE detection is carried out in two steps. In the first step, the input sentence is parsed using the state-of-the-art spaCy parser (Honnibal and Johnson, 2015) and using the Part-of-Speech (POS) tags, we check for the presence of nominal modifiers and determiners from the aforementioned syntactic categories that can potentially license an

Figure 1: F1-score corresponding to different window sizes for searching nouns.



NPE. If the system detects POS tags corresponding to any such category, it proceeds to the second step.

5.1.2 Filter Using Syntactic Features

In this step, the system decides if the selected noun modifier is a licensor of an elided noun or not. This decision is taken using the following syntactic features:

(a) Search for Noun Heads

This simple feature looks for a noun word after the selected noun modifier. We check nouns in a forward search window of 3 words. This window is forward because in English noun heads follow their dependents. If there is no noun present in the next 3 words, the system marks the modifier in question as a licensor. The optimum size of the window as 3 is obtained after experimentation with different window sizes on the 100 sentences from the curated dataset. Figure 1 presents the results of experiments done with different forward window sizes searching for noun heads after noun dependents.

(b) Check for Noun Modifiers as Verbal Arguments

The feature looks for verbs with the selected noun modifiers as the main argument. This is because, many times, spaCy raises a noun modifier to the position of a head in the absence of its noun head, which confirms the presence of an elided noun.

(c) Check for Punctuation

We first check for a simple feature that checks for noun modifiers close to punctuation marks. Since a punctuation can indicate a sentential or phrasal break, this could indicate the absence of a noun head for the given noun modifier in the sentence or noun phrase respectively.

(d) Check for Prepositions

We check if the selected noun modifier is immediately followed by a preposition as that would indicate the beginning of a new (prepositional) phrase and imply that the noun modifier in the given noun phrase does not have a noun head overtly present.

(e) Check for Verbs and Auxiliaries

We also check if the selected noun modifier is immediately followed by a verb or auxiliary verb as that would indicate the end of the given noun phrase immediately after the noun modifier.

5.2 NPE Resolution

Ellipses can be resolved textually when their antecedents are present in the same text as in (1). Such cases of ellipsis are called *endophoric*. However, not all ellipses can be recovered or inferred from a co-text. It is also possible that the antecedent of a given ellipsis is present outside the given text. For example, consider a speaker pointing towards pencils in a shop and uttering a sentence such as (4).

4. Give me three [e].

Using visual context, the shopkeeper can easily resolve the ellipsis in this sentence as *three pencils*. Such cases of ellipses are called *exophoric* or situational as they need situational context to resolve. Since we are only limited to text processing at this stage in the current paper, we only focus on resolving NPE that have textual antecedents.

5.2.1 Ellipsis-Antecedent Environment

It is shown that clauses that are linked by an ellipsis-antecedent relation often have similar syntactic structure and priming effects (Xiang et al., 2014). Further, there is evidence that parallelism in discourse can be applied to resolve possible readings for VPE and possibly other ellipsis and reference phenomenon (Hobbs and Kehler, 1997). This becomes our motivation to resolve NPE.

(a) Match POS tags of the Licensor with other Noun Modifiers

The syntactic environment of the an NPE comprises the remnants left in the noun phrase. One simple way to see structural parallelism between the syntactic environment of antecedent and that of ellipsis to locate antecedents is through matching the syntactic category information. From the detection task, we already have the POS tag information of the licensor of the NPE. In the first step of antecedent selection, the system checks for other noun phrases in the sentence that contain the modifiers with the same POS tag as that of the licensor of the NPE.

(b) Select Antecedent

If a POS tag matching the licensor of the NPE (detected in the NPE detection task) is found in the sentence, the system outputs the noun that the modifier with the same tag modifies as the antecedent of the NPE. If there are more than one such modifiers found, the system selects the one nearest to the NPE as distance generally has a role to play in anaphora and coreference resolution tasks (Lappin, 1996).

6 Results

Simply looking for nouns in the context of the noun modifiers gives a poor F1-score of 64.20%. Addition of only the feature that checks for noun modifiers raised as verbal arguments results into an increased in F1-score by 10.25%. Addition of only the punctuation feature after auxiliary and modal verbs resulted into an increase in accuracy by 3% for VPE detection task (Nielsen, 2004). However, in our task, this resulted into a drop in accuracy by 0.05%. Hence, we excluded this feature from the final system. Addition of only the feature that checks for prepositions immediately following the noun modifier results into an increase in F1-score by 13.73%. Finally, addition of only the feature that checks for verbs and auxiliaries that immediately follow the noun modifier gives an increase in F1-score by 11.46%. These features are independent of each other and do not follow any hierarchy.

The final system together with only the significantly important features is tested on the testset containing 76 positive and 132 negative samples of NPE. The detection system is able to correctly detect 65 instances of NPE out of 76. It also rightly predicts 113 out of 132 negative samples as not containing any NPE. It fails to detect 11 positive cases and falsely detects 29 others. This gives us a final precision of 69.15%, a recall of 85.53% and an F1-score 76.47%. Out of the 65 NPE cases detected by the system, 41 have a textual antecedent and the remaining 24 are exophoric and need extra-linguistic context to resolve. Our system is able to select a potential antecedent for 37 of these from the text, of which 32 are correct predictions. The system fails to select any antecedent for the 9 cases. This gives us a final precision of 78.79%, a recall of 63.41% and an F1-score 70.27%. See table 2 for precision, recall and F1-score values for the NPE detection and resolution tasks on the testset.

One of the main reasons of the low accuracy of our system is wrong POS tags generated for sentences with missing or incomplete information as in the case of ellipses (Menzel, 2017). Secondly, although, licensors of NPE and modifiers of the antecedent indeed show similarity in terms of syntactic category information, this might not always be the case.

5. The books were new, and all six [e] were on syntax.

For example in (5), the NPE licensor is a cardinal number, but antecedent *books* has the definitive article in its Noun Phrase.

7 Conclusion & Future Work

This paper described a syntax-based system for automatic detection of NPE in English. The sys-

Task	Positive Samples	Negative Samples	Precision	Recall	F1-Score
Detection	76	132	69.15%	85.53%	76.47%
Resolution	65	29	78.79%	63.41%	70.27%

Table 2: Performance of NPE detection and resolution systems on the testset.

tem takes in free English text and exploits syntactic constraints on the licensors of NPE to mark the site of ellipsis and syntactic parallelism between antecedent-ellipsis syntactic environments to select potential antecedents. Evaluated on a testset containing both positive and negative NPE samples, the system achieves an F1-score of 76.47% on the detection task and 70.27% on the resolution task. Although these numbers are not high, they can be useful as baselines for future work in this direction. NLP research on ellipsis and NPE in particular suffers from a scarcity of resources. While a rule-based system such as ours does not need sizable data for training, with more language resources available in future, machine learning methods can also be used.

References

Yoshihiko Asao, Ryu Iida, and Kentaro Torisawa. 2018. Annotating zero anaphora for question answering. In *LREC*.

Johan Bos and Jennifer Spenader. 2011. An annotated corpus for the analysis of vp ellipsis. *Language Resources and Evaluation*, 45(4):463494.

Wei Chen. 2016. The motivation of ellipsis. *Theory and Practice in Language Studies*, 6(11):2134–2139.

Sandra Chung, William Ladusaw, and James McCloskey. 2010. Sluicing (:) between structure and inference. *In Representing language: Essays in honor of Judith Aissen*.

Norbert Corver and Marjo van Koppen. 2011. Np-ellipsis with adjectival remnants: A microcomparative perspective. *Natural Language and Linguistic Theory*, 29.

Jeroen van Craenenbroeck and Jason Merchant. 2013. Ellipsis phenomena. In *The Cambridge Handbook of Generative Syntax*, pages 701–745. Cambridge University Press.

Mary Dalrymple, Stuart M. Shieber, and Fernando C. N. Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Lyn Frazier. 2008. Processing ellipsis: A processing solution to the undergeneration problem? In *Proceedings of the 26th West Coast Conference on Formal Linguistics*.

Mary Gardiner. 2003. *Identifying and resolving one-anaphora*. Department of Computing, Division of ICS, Macquarie University.

Tilbe Goksun, Tom W. Roeper, Kathy Hirsh-Pasek, and Roberta Michnick Golinkoff. 2010. From nounphrase ellipsis to verbphrase ellipsis: The acquisition path from context to abstract reconstruction.

Christine Gunther. 2011. Noun ellipsis in english: adjectival modifiers and the role of context. *The structure of the noun phrase in English: synchronic and diachronic explorations*, 15(2):279–301.

Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. 1976. Cohesion in english. page 76.

Daniel Hardt. 1997. An empirical approach to vp ellipsis. *Computational Linguistics*, 23(4):525541.

Daniel Hardt. 1999. Dynamic interpretation of verb phrase ellipsis. *Linguistics and Philosophy*, 22(2):187–221.

Jerry R. Hobbs and Andrew Kehler. 1997. A theory of parallelism and the case of vp ellipsis. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, pages 394–401, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Nina Hyams, Victoria Mateu, and Lauren Winans. 2017. Ellipsis meets wh-movement: sluicing in early grammar.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Trans. Asian Lang. Inf. Process.*, 6.

Kyle Johnson. 2001. What vp ellipsis can do, and what it cant, but not why. pages 439–479.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora for everyone: pronominal anaphora resolution without a parser. In *In Proceedings of COLING*, page 113118.

Nayoun Kim, Laurel Brehm, and Masaya Yoshida. 2019. The online processing of noun phrase ellipsis and mechanisms of antecedent retrieval. *Language, Cognition and Neuroscience*, 34(2):190–213.

Ronald W Langacker. 1999. *Grammar and Conceptualization (Cognitive Linguistics Research)*, volume 14. Mouton de Gruyter, New York.

Shalom Lappin. 1996. The interpretatin of ellipsis. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 145–176. Blackwell.

Shalom Lappin and Herbert Leass. 1994. A syntactically based algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20:535561.

Ekaterina Lapshinova-Koltunski, Christian Hardmeier, and Pauline Krielke. 2018. Parcorfull: a parallel corpus annotated with full coreference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*.

Charlotte Lindenbergh, Angeliek van Hout, and Bart Hollebrandse. 2015. Extending ellipsis research: The acquisition of sluicing in dutch. *BUCLD 39 Online Proceedings Supplement*, 39.

Zhengzhong Liu, Edgar Gonzalez, and Dan Gillick. 2016. Verb phrase ellipsis detection using automatically parsed text. pages 32–40.

Anne Lobeck. 1995. *Functional Heads, Licensing, and Identification*. Oxford University Press.

Marjorie McShane and Petr Babkin. 2016. Detection and resolution of verb phrase ellipsis. *Linguistic Issues in Language Technology*, 13(1).

Marjorie McShane, Sergei Nirenburg, and Petr Babkin. 2015. Sentence trimming in service of verb phrase ellipsis resolution. In *EAPCogSci*.

Katrin Menzel. 2017. *Understanding English-German contrasts: a corpus-based comparative analysis of ellipses as cohesive devices*. Ph.D. thesis, Universitat des Saar- landes, Saarbrucken.

Jason Merchant. 2004. Fragments and ellipsis. *Linguistics and Philosophy*, 27(6):661–738.

Jason Merchant. 2010. *Three Kinds of Ellipsis: Syntactic, Semantic, Pragmatic?*

Hwee Tou Ng, Yu Zhou, Robert Dale, and Mary Gardiner. 2005. A machine learning approach to identification and resolution of one-anaphora. pages 1105–1110.

Leif Arda Nielsen. 2003. Using machine learning techniques for vpe detection.

Leif Arda Nielsen. 2004. Verb phrase ellipsis detection using automatically parsed text.

Sun-Young Oh. 2005. English zero anaphora as an interactional resource. *Research on Language and Social Interaction*, 38(3):267–302.

Dongwoo Park. 2017. *When does ellipsis occur, and what is elided?* PhD dissertation, University of Maryland.

Alain Rouveret. 2012. Vp ellipsis, phases and the syntax of morphology. *Natural Language & Linguistic Theory*, 30(3):897963.

Mamoru Saito, Jonah Lin, and Keiko Murasugi. 2008. Nominal-ellipsis and the structure of noun phrases in chinese and japanese.

Sebastian Schuster, Joakim Nivre, and Christopher D. Manning. 2018. Sentences with gapping: Parsing and reconstructing elided predicates. *ArXiv e-prints*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Frank Wijnen, Tom W. Roeper, and Hiske van der Meulen. 2003. Discourse binding: Does it begin with nominal ellipsis?

Ming Xiang, Julian Grove, and Jason Merchant. 2014. Ellipsis sites induce structural priming effects.

Ching-Long Yeh and Yi-Chun Chen. 2019a. Using zero anaphora resolution to improve text categorization.

Ching-Long Yeh and Yi-Jun Chen. 2019b. An empirical study of zero anaphora resolution in chinese based on centering model.

Is Similarity Visually Grounded? Computational Model of Similarity for the Estonian language

Claudia Kittask
Institute of Computer Science
University of Tartu
Tartu, Estonia
claudiakittask@gmail.com

Eduard Barbu
Institute of Computer Science
University of Tartu
Tartu, Estonia
eduard.barbu@ut.ee

Abstract

Researchers in Computational Linguistics build models of similarity and test them against human judgments. Although there are many empirical studies of the computational models of similarity for the English language, the similarity for other languages is less explored. In this study we are chiefly interested in two aspects. In the first place we want to know how much of the human similarity is grounded in the visual perception. To answer this question two neural computer vision models are used and their correlation with the human derived similarity scores is computed. In the second place we investigate if language influences the similarity computation. To this purpose diverse computational models trained on Estonian resources are evaluated against human judgments.

1 Introduction

Various disciplines and research communities are interested in the study of similarity: Philosophy, Psychology, Computational Linguistics, Semantic Web and the Linked Data communities, to name a few. For example, to integrate heterogeneous semantic resources the Linked Data and Semantic Web communities estimate the degree of similarities between the concepts in these resources (Euzenat and Shvaiko, 2013; Harispe et al., 2015).

In Computational Linguistics researchers build computational models of similarity. To test them, the correlation between the human similarity scores and the scores assigned by the computational models is calculated. It is assumed that the best computational models predict better the human judge scores.

However, earlier studies of similarity suffered from a drawback: they do not distinguish between the relations of similarity and association. In psychology, for example, the distinction between these two notions is well understood. The association between two concepts is defined as the propensity of a subject to activate a representation of the second concept when the first concept is presented. In contrast, the similarity is defined as the proximity of two mental representations. In the Gestalt psychology for example (Wertheimer, 1938) the similarity is seen as the principle of organization of objects in perceptual groups. The concepts **cup** and **tea** are associated but not similar: there is no perceptual principle to group together an object like a cup and a liquid like tea. However, the objects denoted by the concepts **apple** and **pear** are perceptually similar. To remedy this problem SimLex-999 (Hill et al., 2015), a genuine data similarity set containing human judge similarity and concreteness scores for 999 English word pairs, has been built.

An interesting distinction is that between the surface similarity and deep similarity (Vosniadou and Ortony, 1989). The surface similarity is perceptually grounded and it is used in categorization. In contrast, the deep similarity is related to deeper properties not readily accessible to perception. A question we study is: How much of the similarity is grounded in the perceptual properties? In this research the degree of similarity grounded in visual properties is estimated by computer vision models.

If for the English language computational models of similarity have been implemented and evaluated, this is not the case for other languages. In particular, for the Estonian language there is no human annotated set that reflects the true similarity. We translate the SimLex-999 into Estonian and evaluate computational models of similarity based on Estonian language resources: corpora, taxonomies and lexical ontologies.

The rest of the paper is organized as follows. The next section puts our research in context and then the EstSimLex-999 set, the SimLex-999 set translated into Estonian, is presented 3. Section 4 discusses the families of models for computing the similarity between the word pairs in EstSimLex-999. Section 5 presents and discusses the results. In particular, we answer how similarity is influenced by language and quantify the power of the computer vision models to capture the similarity for concrete concepts. The paper ends with the conclusions.

2 Related Work

Felix Hill and coauthors (2015) undertook an extensive discussion of the concept of similarity in Computational Linguistics, introduced the genuine similarity set SimLex-999 and computed the correlations between the similarity measures of corpus based computational models and the human judge scores. Closely following this line of research Ira Leviant and Roi Reichart (2015) translated SimLex-999 in Italian, German and Russian and collected similarity scores from native speakers. They compute correlations between the human judgments and Vector Space Models (VSM) in a multilingual setting.

In the subsequent research the authors improve the similarity computational models and boost the correlation coefficient with the human judgments. For example, Schwartz et al. (2015) learn a word level representation based on symmetric patterns that achieves a Spearman correlation of 0.517 with SimLex-999. An interesting work belongs to Faruqui and Dyer (2015), who used non-distributional word representations derived form Princeton WordNet, FrameNet and Penn Treebank to reach a Spearman correlation of 0.58 with SimLex-999. Hybrid models (Recski et al., 2016),combining features from lexical ontologies and word embeddings, seem to be even better (Spearman Correlation 0.76).

In this work we were not interested in obtaining the best correlation between the computational models and the human judgments. That will be the topic of a future work. Instead, we were concerned with three problems. First, we are interested in how much of similarity is grounded in the visual features, that is, how much of the similarity is surface similarity. By evaluating the similarity using computational vision models we contribute to a better understanding of the notion of similarity itself. Second, we ask how the traditional models derived from Estonian corpora and lexical ontologies correlate with the judgments of native Estonian speakers. In this way we extend the similarity study to other language, a less explored one, yet an interesting one. Third, we study if our computational models trained on Estonian data predict better the EstSimLex-999 scores or the SimLex-999 scores. More precisely we want to know if the language influences the similarity judgments.

3 EstSimLex-999

To translate SimLex-999 the Google Translation API and a bilingual English-Estonian dictionary containing 87665 entries have been used, obtaining rough Estonian equivalents. A native Estonian speaker has chosen the correct translations. If an English word in a similarity pair is ambiguous, the sense that makes the pair more similar is preferred. Finally, after correction and the discussion with an Estonian linguist we have produced the similarity set referred from now on as EstSimLex-999. When translating, we have been careful to preserve the part of speech of the English concepts. This makes the comparison between the computational models of similarity for English and Estonian easier. Nevertheless, due to cultural and linguistic differences some English similarity pairs were hard to translate. For example, the English pair (taxi, cab) was translated as (taksi, takso) even if the second term of the Estonian pair is not widely used. Another example is the pair (supper, dinner). The Estonian culinary tradition does not distinguish between the two concepts, therefore we have translated the pair with the synonymous words (õhtusöök, õhtueine). Please, notice, that for many non-British native English speakers the words *supper* and *dinner* are also synonymous. Some translations would have been more accurate using multiwords, but we abide by the original requirement that the similarity pairs should contain single words only. Overall, we have produce an accurate translation of the English original SimLex-999 set preserving the distribution of the part of speeches and satisfying the demand that the word pairs should not contain multiple words.

Four native Estonian speakers have rated the degree of similarity between each of the 999 pairs.

The rating instructions are the same as in the original study (Hill et al., 2015). These instructions do not attempt to define what similarity is, but rather clarify the concept contrasting it with association, and comparing it with synonymy. The inter-annotator agreement was computed as the average of pairwise Spearman correlations between the scores of all raters. The overall agreement is 0.766. A direct comparison with the correlation coefficient computed in the English study (0.67) is not possible because the number of annotators is different. At this stage we were not interested in recruiting many annotators through platforms like Mechanical Turk, but rather in gaining insights into human similarity judgments by direct discussion with the annotators. In any case, recruiting a comparable number of Estonian speakers is unlikely, as this language is natively spoken by less than 1 million people. The main thing noticed is that there are few pairs of adjectives and verbs highly rated in English but with a low score in Estonian. For example, the English verb pair (appear, attend) has a score of 6.28 in SimLex-999 and its Estonian translation (ilmuma, osalema) has a score equal to 0.5.

4 Models for Similarity Computations

Three families of similarity models are evaluated : distributional models, semantic network models and computer vision models.

The distributional models are an implementation of John Rupert Firth's hypothesis "You shall know a word by the company it keeps" (Firth, 1961) which basically states that words that have similar meanings appear in comparable syntagmatic contexts. Nowadays, the most advanced distributional models are the neural word embeddings.

The second family of models derive the semantic similarity from the taxonomic structure of semantic networks. The IS-A relation induces the inheritance of the properties.The above mentioned concepts, *apple* and *pear*, are similar because they inherit all the properties from their superordinate concept (fruit). Unlike the distributional models, the semantic networks tells us also why the concepts are similar.

The third family of models are the computer vision models. The similarity between two concepts is the distance between their image representations. Because of the visual nature of this similarity the computer vision models work best for concepts representing concrete objects.

In what follows we will briefly describe the models tested.

1. **Word2Vec**. Word2Vec is a distributional model (Mikolov et al., 2013) implemented as a two layer neural network. If two words appear in similar contexts in a corpus the network will output embedding vectors, known as neural vector embeddings, which are close in the embedding space. Word2Vec computes the neural vector embeddings either predicting the target word from the context (this method is known as continuous bag of word (CBOW)) or as the target context from the word (this method is know as Skip Gram).

2. **SenseGram**. SenseGram (Pelevina et al., 2016) is not a distributional model per se, but a method to obtain word senses from word embeddings. This word discrimination method takes as input word embeddings (like those generated by Word2Vec or any other distributional model) and clusters them. The induced word senses correspond to the clusters of word embeddings.

3. **Path Similarity Measures**. The path similarity measures exploit the graph structure of semantic networks to find similarities between concept pairs. We have explored various similarity measures like Leacock-Chodorow similarity (Leacock and Chodorow, 1998).

4. **Autoencoders**. Autoencoders are deep neural networks which learn to reconstruct the input. In the reconstruction process one of the autoencoder layers contains less nodes than the input layer, thus forcing the network to learn a lower level representation of the input. The idea behind using the autoencoders is that the sparse representations learned when encoding similar concepts will be close in the embedding space.

5. **Pretrained Convolutional Neural Networks**. Convolutional Neural Networks (CNN) are deep neural networks architectures suitable for extracting patterns from images. Inspired by experiments in neuroscience (Hubel and Wiesel, 1959), CNN's

first layers train convolution filters to detect low-level features of an image like lines and corners. Higher network levels combine the low level-features to find high-level image features roughly corresponding to the human language semantic descriptions of the objects. For example, they might detect parts, like the wheels or the hood of a car. When CNN are trained on big databases of classified images the semantic representations of the concrete concepts can be "read" from the deeper network levels. These representations are then used to compute concept similarity.

5 Results

First the results for the distributional models are shown, then the results for the semantic network models will be presented. Finally, the results for the neural computer vision models will be shown. The correlation coefficients between the scores assigned by the computational similarity models for interesting subsets (e.g. abstract and concrete concepts) and the two similarity sets are also computed. In the tables in this section the SimLex-999 is abbreviated as SL-999 and the EstSimLex-999 as ESL-999.

5.1 Distributional Models

When evaluating the **Word2Vec** and **SenseGram** models, if the embedding vectors corresponding to the words in the SimLex-999 or EstSimlex-999 are missing, the word-pair is eliminated. The model word similarity score is computed as cosine similarity between the vector embeddings corresponding to the words in the each word pair. Pearson (r), Spearman (ρ), and Kendall (τ) correlations are calculated between EstSimLex-999 and Simlex-999 human judge scores and the model word similarity scores. The word embeddings were trained on Estonian monolingual corpora and the Estonian Wikipedia. The following word embeddings have been used:

- **EA word embeddings**. 9 Skip-Gram and 20 CBOW models, with different parameter settings, were trained on the lemmatized version of etTenTen corpus of Estonian Web [1] by Eleri Aedma. Word senses were induced from the traditional word embeddings using SenseGram. SenseGram finds 1.6 senses/concept, with about 300 word pairs having more than one sense. For the ambiguous word pairs (where at least one of the words in the pair has more than one sense) the word sense that maximizes the cosine similarity score of a word pair is evaluated.

	SL-999			ESL-999		
Model	r	ρ	τ	r	ρ	τ
cbow_1	.42	.42	.29	.46	.47	.33
sg_2	.37	.36	.24	.41	.42	.3
cbow_3	.33	.33	.23	.33	.34	.24

Table 1: The results for the best three distributional models

- **Estnltk pretrained word embeddings**. Estnltk (Orasmaa et al., 2016) contains 8 word pretrained embeddings. 4 of them are trained with the CBOW method and the other 4 were trained with the Skip-Gram method, on the raw and lemmatized versions of the Estonian Reference Corpus (Kaalep et al., 2010). The Estonian Reference Corpus is a 1.3 billion word corpus, crawled from the web, containing mainly newspaper text.

- **Facebook pretrained word embeddings**. The Facebook word embeddings (Bojanowski et al., 2017) have been trained with CBOW method on 294 language versions of Wikipedia.

The distributional models evaluate on average 985 word pairs. Each CBOW and Skip Gram model has 4 meta-parameters : the number of dimensions, the window size, the minimum count threshold and the number of iterations. Due to consideration related to space we only present the best three results in the table 5.1. The whole set of results for the 67 distributional models trained and all the figures and the tables in this paper are available online linked from our github repository. [2]. The best model on the first row in the table 5.1, for example, has been trained with the 300 dimensions, a window size equal to 1, the minimum count threshold being 10, and 20 iterations.

In the first place one can notice that CBOW trained word embedding perform better than Skip-Gram trained word embeddings. Moreover, the correlation coefficients between EstSimLex-999

[1] DOI: 10.15155/1-00-0000-0000-0000-0012EL

[2] https://github.com/estsl/EstSimLex-999

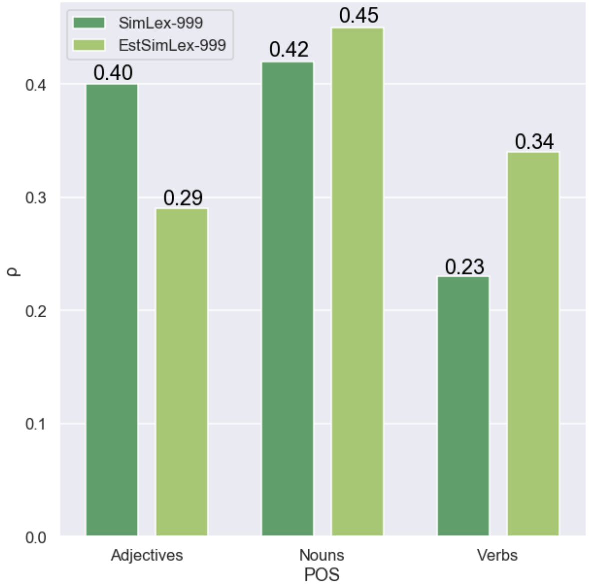


Figure 1: The average performance for POS-based subsets

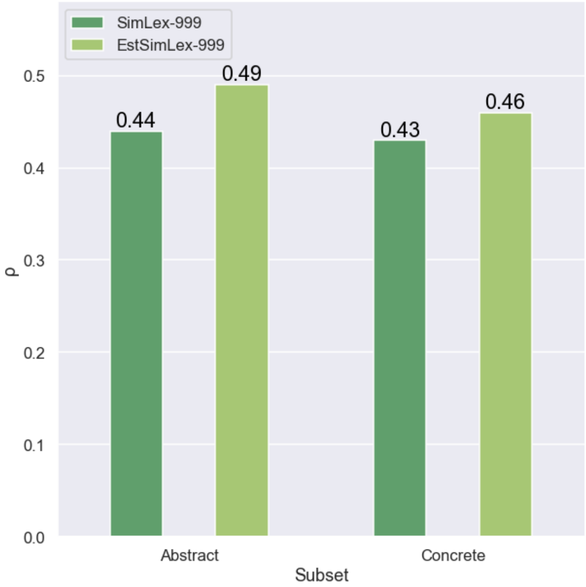


Figure 2: The average performance for the most concrete and abstract subsets

human scores and the model computed word similarity are higher than the correlation coefficients between SimLex-999 human scores and the model computed word similarity. The automatic sense discrimination had a negative influence on the results, as the SenseGram induced vector senses show a slight drop in performance over the traditional word-embeddings. The Estnltk trained word embeddings perform worse than EA word embeddings, but better than Estonian Facebook pretrained word embeddings.

Furthermore, we study if the part of speech category influences the strength of the correlation between human judgments and the distributional models. The model similarity scores between the words in the word pairs is the average similarity scores for all the distributional models.The correlation coefficients between the model scores for 666 noun pairs, 111 adjective pairs and 222 verb pairs and the human judgment scores is calculated.

As it can be seen in figure 1, the best (Spearman) correlation coefficients between the models and the similarity sets are obtained for the nouns. The (Spearman) correlation coefficient between the distributional models and the human judgments is higher for EstSimLex-999 set than for the original SimLex-999 set.

The correlation coefficients between the 250 most concrete word pairs and the 250 most abstract word pair and the distributional models have also been computed. The results presented in figure 5.1 show that, on average, the distributional models correlate better with the abstract human judgments scores and that the correlation coefficient is higher for the EstSimLex-999 set.

5.2 Semantic Network Models

The similarity between the concepts corresponding to the words in EstSimLex-999 word is computed for two semantic networks: the Estonian Wordnet and a taxonomy derived from the Estonian Wikipedia.

As the Estonian Wordnet lists multiple senses for words, a disambiguation procedure to select the most likely sense is implemented. The Cartesian product between the word senses in the semantic network corresponding to the words in the EstSimLex-999 is generated. Thus we obtain a set of word-sense pairs. Subsequently, as explained below, a similarity scores is assigned to each word-sense pair in this set. The word sense pair that maximizes the similarity score is chosen. This procedure of mapping the words onto a semantic network is very effective, obtaining over 90 percent precision for the Estonian Wordnet (Barbu et al., 2018) .

Three similarity measures between the semantic network concepts have been computed: path similarity (PS), Leacock & Chodorow similarity (LC) (Leacock and Chodorow, 1998) and Wu &

	SL-999			ESL-999		
	r	ρ	τ	r	ρ	τ
PS	.47	.47	.35	.54	.52	.39
LC	.36	.36	.26	.41	.43	.31
WuP	.41	.45	.32	.49	.53	.39

Table 2: The results for the Estonian Wordnet

Palmer similarity (WuP) (Wu and Palmer, 1994). Pearson (r), Spearman (ρ), and Kendall (τ) correlations were calculated between EstSimLex-999 and SimLex-999 human judge scores and the network similarity scores for the disambiguated word pairs.

The Estonian Wordnet is an ongoing effort, it pursues roughly the same organization principles as Princeton WordNet (Miller et al., 1990), and it is manually built by a group of linguists. The version used in this study contains approximately 85.000 synsets. The above described disambiguation procedure maps approximately 770 word pairs onto the Estonian Wordnet. The results corresponding to the Estonian Wordnet are in the table 2.

The best results are obtained for Wu & Palmer similarity measure and EstSimLex-999. This similarity measure considers the depth of the concepts in the semantic network hierarchy along with the depth of their Lowest Common Subsumer. Unlike the path similarity measure it favour the concepts that are deeper in the hierarchy. As in the case of the distributional models the EstSimLex-999 correlation scores are better than the SimLex-999 ones. A fact worth noticing is that the difference between the correlation scores for the two similarity sets is greater when we compute the similarity score based on the Estonian Wordnet structure instead of estimating the similarity using distributional models.

The taxonomy was extracted from the (Estonian) Wikipedia page text (Wikipedia Page Taxonomy) by the language technology research group at Università Roma Tre (Flati et al., 2016). The Wikipedia Page Taxonomy contains approximately 87000 concepts. We could map around 200 word pairs onto the Wikipedia Page Taxonomy. The results for this taxonomy are in Table 3.

	SL-999			ESL-999		
	r	ρ	τ	r	ρ	τ
PS	.32	.31	.22	.37	.34	.24
LC	.31	.3	.21	.35	.34	.28
WuP	**.39**	**.37**	.28	.4	**.37**	.27

Table 3: The results for the Wikipedia Page Taxonomy

The correlation coefficients between the similarity measures computed for the Wikipedia Page taxonomy and the human judgments scores are much lower than those computed before (with the Estonian Wordnet). Also, surprisingly and for the first time in this study, there is no statistically significant difference between the correlation coefficients computed with SimLex-999 human judgments scores and the EstSimLex-999 human judgment scores.

5.3 Computer Vision Models

Because the visual similarity is correlated with the level of concreteness of an object, the computer vision models are fed with word pairs where both words have a degree of concreteness higher than the a threshold equal to 4.8 . This criterion gives us 136 word pairs. We have downloaded, using Yandex image search engine 200 images for each word in the selected word pairs.

The first architecture trains a Convolutional Autoencoder (CAE) on the downloaded images. The encoder consists of 3 convolutional layers, each followed by a max-pooling layer. The decoder consists of 3 convolutional followed by upsampling layers. The similarity between two images is calculated as the cosine similarity between the corresponding encoder vectors. The similarity score for a word pair is the average score between all the images corresponding to the words in the pair.

The second architecture is the winner of the ImageNet 2015 competition. It is a CNN network architecture invented by Microsoft Research, called ResNet(He et al., 2016) (abbreviation for Residual Network). DNNs with many layers are difficult to train due to vanishing and exploding gradient problems. ResNet solves these problems with residual learning. The ResNet architecture comes in many variants, depending on the number of layers the network has. The widely employed ResNet-18 variant with 18 layers is used in this study. The network is pretrained on the ImageNet database (Deng et al., 2009) which contains over 1 million images classified under 1000 Princeton WordNet categories. Being trained on such a big

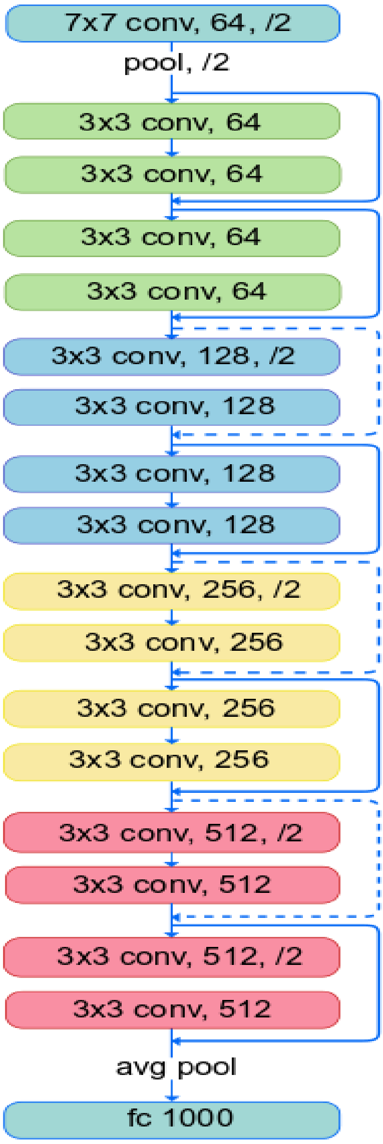


Figure 3: Architecture of ResNet-18

and diverse database of images, ResNet learns rich feature representations that help discriminate between images belonging to different categories.

The architecture of ResNet-18 network is presented in 5.3. As can be noticed the network repeats the same structure, and it ends with the average pool layer that feeds a fully connected layer. The probability that an image belongs to a category is computed by a softmax layer. The network is fed with the downloaded images corresponding to the words in the selected concrete set. The image representation learned by the network is read from the average pool layer. The score this visual model assigns to a word pair is the average cosine similarity score between the pair words image representations. As a way of example let's take the hypothetical word pair (cat, dog). The score assigned by the model to this pair is the average cosine similarity score between the ResNet-18 representations of the images corresponding to the words dog and cat.

Model	SL-999			ESL-999		
	r	ρ	τ	r	ρ	τ
CAE	.25	.28	.19	.17	.22	.15
RN18	.37	**.38**	.26	.34	**.38**	.27

Table 4: The results for the convolutional autoencoder (CAE) and ResNet-18 (RN18) architectures

The results for the two computer vision architecture are presented in Table 4. As expected, the rich visual features learned by the ResNet-18 architecture boost the results (the best results are bold marked in the table) for both similarity sets. The highest correlation coefficients are between the computer vision models and both similarity sets human judge scores.

For the three families of models three correlation coefficients have been computed. These coefficients do not induce a different order on the results, therefore the usage Spearman correlation coefficient in the previous studies is justified.

In general the correlation coefficients between all families of computational models and the human judge scores of EstSimLex-999 are better than the same correlation coefficients and the human judge scores of SimLex-999. This result shows that there is a slight language effect on the perception of similarity.

Regarding the magnitude of the Spearman correlation coefficient for the Estonian side of the equation, the distributional models show a moderate strength [3] of correlation with the human judge scores. This means that the distributional models do capture some of the notion of similarity between the words, but they also capture something else. The best Spearman correlation coefficient is obtained with the Estonian Wordnet (0.53), being better than the best correlation coefficient for distributional models (0.47), but still in the moderate range. It seems that the best predictor of human similarity is derived from a manually built resource containing clearly defined semantic relations.

Less than 40 percents of the similarity of the concrete concepts can be explained by the visual

[3] In the literature the strength is moderate if the coefficient is in the range 0.4-0.59.

semantic features when these features are computed from huge databases of images like ImageNet. Although the empirical evidence heavily depends on the quality of the ImageNet database this finding shows that other factors have significant weight in human similarity judgments. Maybe a case can be that these factors are the deep semantic features we have briefly mentioned in the introduction section. However, a definitive answer to what these factors are and how to account for them goes beyond this research.

6 Conclusions

In this study have addressed some aspects of the computational models of similarity applied to the Estonian language.

In the first place we have found that the neural visual models of similarity can explain a part of the similarity between the words representing concrete concepts. This invites the conclusion that deep similarity models might be involved in accounting for the unexplained part of similarity.

In the second place and differently from the finding in (Leviant and Reichart, 2015) an effect of the language on the similarity has been found. Unlike in that study the Estonian computational models of similarity better correlate with the word pair similarity score assigned by the Estonian subjects that with the scores assigned by English speaking subjects.

In the third place the best computational models are those derived from human built semantic networks. They are better than the neural distributional models but still they correlate moderately with the human judgments. This means that there is more to similarity than taxonomic similarity. On the other hand the Estonian Wordnet is still work in progress, therefore we cannot rule out that a more complete wordnet can boost the similarity scores. Unlike the original study (Hill et al., 2015) we have found that the word embedding computational models better correlate to the scores for nouns and not the adjectives. Intuitively, this result makes sense as nouns have richer mental representations than other morphological categories, therefore one expects that the similarity is better defined for nouns than for other parts of speech.

In the future we will work to better understand the other components of similarity for the concrete concepts, improve and refine the computational models of similarity for the Estonian language and address the same problem for different languages.

Reproducibility

The EstSimLex-999 set annotated with the human judge similarity scores, the code used to compute the results in this paper, the complete set of tables and the figures and most of the resources used in this paper can be referenced from the Github repository https://github.com/estsl/EstSimLex-999.

Acknowledgments

A number of persons have helped us with this study. We like to thank Aliis Aalmann and Kadri Muischnek for the aid in translation of the SimLex-999. We also like to thank Eleri Aedmaa for making available the word and sense embeddings and the anonymous annotators for rating the Estonian word pairs. This study was supported by the Estonian Ministry of Education and Research (IUT20-56).

References

Eduard Barbu, Heili Orav, and Kadri Vare. 2018. Topic interpretation using wordnet. In Kadri Muischnek and Kaili Müürisep, editors, *Baltic HLT*. IOS Press, volume 307 of *Frontiers in Artificial Intelligence and Applications*, pages 9–17.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jrme Euzenat and Pavel Shvaiko. 2013. *Ontology Matching*. Springer Publishing Company, Incorporated, 2nd edition.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. *CoRR* abs/1506.05230. http://arxiv.org/abs/1506.05230.

J.R. Firth. 1961. *Papers in Linguistics 1934-1951: Repr.* Oxford University Press. https://books.google.ee/books?id=VxiWHAAACAAJ.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2016. Multiwibi: The multilingual wikipedia bitaxonomy project. *Artif. Intell.* 241:66–102.

Sebastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. 2015. *Semantic Similarity from Natural Language and Ontology Analysis*. Morgan & Claypool Publishers.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pages 770–778.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* .

David H. Hubel and Torsten N. Wiesel. 1959. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology* 148:574–591.

Heiki-Jaan Kaalep, Kadri Muischnek, Kristel Uiboaed, and Kaarel Veskis. 2010. The estonian reference corpus: Its composition and morphology-aware user interface. In *Proceedings of the 2010 Conference on Human Language Technologies – The Baltic Perspective: Proceedings of the Fourth International Conference Baltic HLT 2010*. IOS Press, Amsterdam, The Netherlands, The Netherlands, pages 143–146. http://dl.acm.org/citation.cfm?id=1860924.1860949.

Claudia Leacock and Martin Chodorow. 1998. *Combining Local Context and WordNet Similarity for Word Sense Identification*, volume 49, pages 265–283.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR* abs/1508.00106. http://arxiv.org/abs/1508.00106.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to wordnet: An on-line lexical database. *Journal of Lexicography* 3(4):235–244. ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.pdf.

Siim Orasmaa, Timo Petmanson, Alexander Tkachenko, Sven Laur, and Heiki-Jaan Kaalep. 2016. Estnltk - nlp toolkit for estonian. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. Making sense of word embeddings. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 174–183. http://anthology.aclweb.org/W16-1620.

Gábor Recski, Eszter Iklódi, Katalin Pajkossy, and Andras Kornai. 2016. Measuring semantic similarity of words using concept networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 193–200. https://doi.org/10.18653/v1/W16-1622.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Beijing, China, pages 258–267. https://doi.org/10.18653/v1/K15-1026.

Stella Vosniadou and Andrew Ortony, editors. 1989. *Similarity and Analogical Reasoning*. Cambridge University Press, New York, NY, USA.

Max Wertheimer. 1938. Laws of organization in perceptual forms. In W. Ellis, editor, *A Source Book of Gestalt Psychology*, Routledge and Kegan Paul, London, pages 71–88.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '94, pages 133–138. https://doi.org/10.3115/981732.981751.

Language-Agnostic Twitter Bot Detection

Jürgen Knauth
Institute for Computer Science
University of Göttingen
jknauth@uni-goettingen.de

Abstract

In this paper we address the problem of detecting Twitter bots. We analyze a dataset of 8385 Twitter accounts and their tweets consisting of both humans and different kinds of bots. We use this data to train machine learning classifiers that distinguish between real and bot accounts. We identify features that are easy to extract while still providing good results. We analyze different feature groups based on account specific, tweet specific and behavioral specific features and measure their performance compared to other state of the art bot detection methods. For easy future portability of our work we focus on language-agnostic features. With AdaBoost, the best performing classifier, we achieve an accuracy of 0.988 and an AUC of 0.995. As the creation of good training data in machine learning is often difficult - especially in the domain of Twitter bot detection - we additionally analyze to what extent smaller amounts of training data lead to useful results by reviewing cross-validated learning curves. Our results indicate that using few but expressive features already has a good practical benefit for bot detection, especially if only a small amount of training data is available.

1 Introduction

While at the end of the 90s digital communication was still handled exclusively via well-defined Internet protocols, a completely new form of communication has established over the decades using web-based technology: Social media. These are web-based platforms that allow users to publish and exchange messages with each other after registration. These platforms are highly attractive: on the one hand, they establish a form of social community that enables people to enter into new relationships with each other or to maintain existing relationships. On the other hand, these platforms are designed in such a way that their users can easily find new content based on algorithmic recommendations.

Today, these social media platforms cover a wide variety of media. This work focuses on Twitter[1] - a service that enables the publication of short text comments.

As information and opinions are widely disseminated and exchanged via these platforms, the rise of social media as mass communication tools has increasingly attracted the interest of different actors who try to use social media as influencing factors. Businesses recognize the potential to advertise their products in this way at a very early stage, while political actors try to promote their views.

With the rise of such platforms to mass media and the desire to influence public perception and opinion, a comparatively new phenomenon has developed: The use of bots. In addition to normal human users, some social media platforms also feature computer programs that generate and provide content. This phenomenon has been found on Twitter as well as other social media platforms.

Some of these bots are indistinguishable from normal users on a superficial level. This is desired by their creators: Though there are legitimitate applications that justify the use of bots, bots are also used for malicious purposes. E.g. bots can be used for spamming to advertise products or for drawing attention to certain political statements.

As technical methods seem to make it possible to promote certain opinions, there exists at least a

[1] https://twitter.com/

theoretic danger that democratic processes might be in influenced by bots. Badawy et al., 2018 analyzes a set of Russian bots and how their information is picked up by American twitter users related to these bot accounts. Their research shows that information is introduced selectively into social media such as Twitter. Badawy et al., 2018 show that this information gets picked up and is spread - by users and other bots. The extent to which such influence can actually influence public opinion is not yet clear and requires further research. However, the very fact that such information is disseminated makes detecting Twitter bots an important issue to address.

2 Related Work

Detecting Twitter bots has been addressed as a research topic for quite some time. Wang, 2010 reconstructed a graph model based on crawled Twitter accounts and tweets exploiting follower and friend relationships. They use their data to implement a spam detection approach using classic Bayesian classication to distinguish between normal behavior from suspicious behavior on Twitter.

In 2012 a large scale classification study was conducted by Chu et al., 2012 to detect bots, humans and ”cyborgs” in Twitter data. The term ”cyborgs” is used here for accounts that are a symbiosis of humans and bots, making this a multinomial problem. Chu et al., 2012 used a dataset of 2000 manually classified Twitter accounts and achieved an accuracy of about 96% with a random forest approach.

Clark et al., 2015 uses a different approach. They not only crawl Twitter directly to extract tweets but additionally use data from a honeypot. They provided a setup with Twitter accounts being bots themselves. Twitter users following these accounts for no real reason are considered as being bots. Clark et al., 2015 focus on content of the tweets and use three different features to measure tweet similarity and hyperlinks. Based on this they propose a classifier to distingusish between ”organic” and ”robotic” texts achieving an accuracy of 90.32% of the ”organic” and 95.2% of the ”robotic” accounts. They additionally identify different behavior among these accounts indicating that there are several different classes of spammers and spam bots.

Cresci et al., 2015 and Cresci et al., 2017 manually analyzed and annotated over 8000 account data records and conducted an even more in depth analysis of the phenomenon of Twitter bots. Building a classifier they achieved an accuracy of 95% (Cresci et al., 2015). In their 2017 paper they discovered a new generation of social bots being more sophisticated compared to bots already known. They improved their performance by using advanced clustering algorithms and a large set of 126 features and were able to detect up to 97.6% of Twitter bots on one of their data sets.

Newer work uses sophisticated machine learning techniques. Cai et al., 2017 detect bots by modeling users and their behavior using deep learning by focusing on topic, latent sentiment and temporal aspects. Their CNN-LSTM model learns from user content as well as user behavior. With this approach they achieve an F1 score of 88.30% percent for their deep learning model outperforming reference models based on content or behavior alone.

Efthimion et al., 2018 use an approach with support vector machines. They make use of the Levenshtein distance to detect similar posts for their classification approach. This is noteworthy, as we make use of Levenshtein distance as well, but in contrast to Efthimion et al., 2018 use it to detect similarities not among tweet content but on account level. (See below.) Efthimion et al., 2018 use the same data as we use in our work and achieve and accuracy of 95.77% (which we outperform in our work).

Lundberg et al., 2018 provide two classifiers that are used to not only build an offline classifier, but a system that performs online analysis of tweets as they emerge from the Twitter community. They achieve an accuracy of about 98%.

3 Contribution and Outline

3.1 Research Questions

According to the importance of social media and Twitter bots outlined in the introduction, the following scientific questions arise regarding the recognition of Twitter bots:

- Is it possible to detect bots at account level only, without taking into account the content provided by these accounts?
- Does bot detection benefit from content analysis?
- Is it possible to reliably detect bots in a way that avoids language-specific features?

- Creating training data for bot detection is difficult. How large does such a data set have to be in order to achieve useful training results for machine learning? Can a small data set be sufficient?

4 Methodology

4.1 Considerations about Bots and Humans

Our approach to identifying Twitter bots is based on the assumption that bots are fundamentally different from humans in some aspects. Our consideration is that two categories should be distinguished here:

- Technical differences
- Purpose-related differences

4.1.1 Technical Differences

Due to the fact that bots are computer programs, they are not subject to certain human limitations. Computer programs can act instantly in contrast to humans who need time to reflect and are often occupied with other tasks of daily life and work. It can therefore be assumed that human behaviour is different from bot behaviour with regard to timing and the orientation of published content.

Our considerations are also based on the assumption that it is difficult for computer programs to imitate human behaviour. While it is possible to simulate human inadequacies, for example by delaying reactions, it would be difficult for bots to accurately mimic human behavior: This would require extensive statistical analysis of the behaviour of Twitter users. We can therefore assume that bots are created using simpler methods, such as random temporal behaviour, which only resemble human behaviour at first glance.

4.1.2 Purpose-Related Differences

Bots have clear objectives, for example spreading political messages or references to products. Bots bring specific content to attention, hashtags, URLs. So they have some kind of ”agenda” which should be able to be exploited to some extent in general.

It should be noted here that the fact that we and other researchers are able to identify bots quite reliably clearly shows that these assumptions are not unfounded.

4.2 Dataset

Our work is based on the MIB dataset (Cresci et al., 2017), which contains 8375 annotated Twitter accounts.

- 3473 accounts - humans
- 991 accounts - political candidate retweeters
- 3457 accounts - paid apps spammers
- 464 accounts - amazon.com spammers
- Total number of accounts: 8375 accounts

This data set contains data records about the accounts themselves as well as tweets created.

4.3 Baseline

In the next sections we lay out our feature extraction and machine learning process. In order to compare our results with a baseline we reimplement one of the machine learning classification systems described in Kudugunta and Ferrara, 2018. This classifier is quite a high baseline as it performs very well and is - to our knowledge - the current state of the art.

4.4 Feature Extraction

For building a machine learning classifyer we require a matrix of feature values for training and - later on - classifying unseen test data. So in a first first step we extract a variety of features from the Cresci et al., 2017 datasets. These features are discussed in the next subsections.

4.4.1 Account Based Features

The first group of features is derived from account metadata. Our *simple user profile features* directly reflect values the Twitter API provides about users. We additionally derive features with some processing from the screen and user names.

Some of the features are self explanatory or explained by the Twitter API documentation. [2] Nevertheless some of these features require additional discussion.

- **Simple user profile features:** We hypothesize that metadata from the user profile provides valuable information about the user account. Some of this data is generated by

[2] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object.html

Twitter itself and sometimes difficult to control directly by users. This data contains characteristics about a user's account we can exploit for machine learning. These account features include:

  - *default_profile:* Has the user altered the profile?
  - *geo_enabled:* This feature reflects if users enable adding geographic information if they publish a tweet.
  - *protected:* When true, indicates that this user has chosen to protect their Tweets.
  - *is_verified:* This is some kind of quality marker provided by Twitter: Accounts that are run by people of public interest can be verified as being authentic by Twitter itself.
  - *friends_count:* The number of users this account is following.
  - *followers_count:* The number of followers this account has.
  - *favourites_count:* The number of tweets this user has liked.
  - *listed_count:* The number of public lists this account is a member of.
  - *statuses_count:* The number of tweets issued by this account.
  - *profile_use_background_image:* Has the user provided a background image?

- **User profile name features:** User and screen names are very much subject to a user's choice. Therefore we hypothesize that it provides valuable information that helps to distinguish bots from humans. These features include:

  - *screen_name_length:* The length of the screen name provided by a user.
  - *user_name_length:* The length of the account name provided by a user.
  - *screen_name_digits:* Number of digits in the screen name.
  - *user_name_unicode_group:* See below.
  - *screen_name_unicode_group:* See below.
  - *levenshtein_user_name_screen_name:* See below.

We use some features that are closely related to the screen and user names of accounts. In particular, we determine which of the 105 Unicode code groups an account uses in the screen and user names. This is reflected in the categorical features *user_name_unicode_group* and *screen_name_unicode_group* where we have one feature for every Unicode code group. The rationale behind this feature is that humans tend to be quite creative in their choice of names and sometimes tend to pick characters completely unrelated to the alphabet of their own language. By comparing occurrences of characters in various Unicode code groups we take this behaviour into account.

Furthermore we want to make use of possible differences between an account's screen name and user name. We model this by calculating the respective Levenshtein distance (Levenshtein, 1966). We observed that bot accounts tend to choose user names and screen names that are similar, while humans can be more creative in this respect.

4.4.2 Content Based Features

We derive additional features from the content a user provides.

For that purpose we tokenize the tweets. We tokenize by breaking the tweet text at a variety of spaces and punctuation such as commas, colons, exclamation marks, brackets and similar characters that are commonly used in sentences. This tokenization respects characters occuring in emojis as well. While tokenization in itself is not entirely language agnostic, these heuristics should nevertheless work for a fairly large set of (Latin script) languages.

Then we can extract features based on these tokens. Other features are directly derived from the metadata of tweets.

- **Behavioural features:** We hypothesize that the tweeting behaviour of bots and humans should exhibit differences. We model this behaviour by calculating statistical properties from the data such as minimum, maximum, average, mean, median, standard deviation, skewness, kurtosis, and others.

  - *time_between_retweets (distributional feature):* This set of features models time between retweeting activities of an account.
  - *time_between_tweets (distributional):* This set of features models time between tweeting activities.

  - *tweet_rate (average tweet rate):* The average number of tweets per day

- **Core content features:** We hypothesize that some aspects of intention and emotions can be derived from a tweet's content. The following features honor this in a language independent way.
  - *emojis_classic (distributional feature):* See below.
  - *emojis_kaomji_faces (distributional):* See below.
  - *emojis_line_art (distributional):* See below.
  - *emojis_other (distributional):* See below.
  - *number_of_tokens (distributional):* This feature models the size of a tweet to some extent.
  - *number_of_hashtags (distributional):* The number of hashtag based references contained in tweets.
  - *n_of_tokens_wo_hashtags_urls_symbols (distributional):* A distribution based on the number of plaintext tokens in the tweets excluding hashtags, URLs and non-alphanumeric tokens.
  - *number_of_urls (distributional):* The number of URL based references contained in tweets.
  - *special_char_repeats_rate:* This feature detects sequences of question marks and exclamation marks. These are also often used for affirmation to give special weight to what has been expressed in a tweet. We want to model this aspect.

We assume that a precise sentiment analysis of the twitter text is not available for all languages tweets could be written in. Nevertheless, we want to extract and use emotional aspects from the content. In order to model some basic aspects of emotion we detect various different sets of emoticons here. The extraction is pattern based and derived from public emoticon collections as provided by Wikipedia. The four emoji features above are distributions derived on individual occurrences of emojis in single tweets.

4.5 Feature Groups

Based on these features we build sets of different features for use in different machine learning experiments.

For reasons of getting a more detailed understanding about the performance of various feature sets we group all behavioral features to a feature set named `tweet-behav`, all tweet content related features to a set named `tweet-cont` and all account related features to `account`. We use `account-lev` as a feature set that only includes the Levenshtein distance between user and screen names to test this feature specifically.

As we want to compare our work to Kudugunta and Ferrara, 2018 we reimplented all their account features mentioned in their paper. We later on refer to this feature group with `k_f_reimpl`.

Our own feature group `all` contains almost all features described above leaving out only four minor features as they don't seem to improve the quality of our classifier: Information about the default profile, location, the content of the ”protected” field and information about user background image.

5 Results and Discussion

We generate feature matrices based on different sets of features as described in section *4.5*. This data is normalized by scaling each feature to the unit interval and then used to train and evaluate different machine learning models.

To evaluate the performance of our machine learning classifiers we separate our dataset into a training and validation set by an 80:20 ratio using (deterministic) random selection provided by the scikit-learn framework (Pedregosa et al., 2011).

We have experimented with classical methods such as Logistic Regression[3], Support Vector Machines[4], Random Forests[5] and Multi-layer Perceptrons[6]. The best results were achieved with *AdaBoost*[7] (Freund and Schapire, 1999). Since the training labels are not fully balanced, we follow Kudugunta and Ferrara, 2018 and resampled the training data using SMOTE-ENN (Lemaître et al., 2017).

By using AdaBoost with SMOTE-ENN we achieve the results displayed in table 1. Figure 1 contains some of the corresponding ROC curves.

The feature set **tweet-behav** contains all features related to the tweeting behavior of an account

[3] `sklearn.linear_model.LogisticRegression`
[4] `sklearn.svm.LinearSVC`
[5] `sklearn.ensemble.RandomForestClassifier`
[6] `sklearn.neural_network.MLPClassifier`
[7] `sklearn.ensemble.AdaBoostClassifier`

Feature-Set	P	R	F1	ACC	AUC_ROC
tweet-behav	0.9920	0.8888	0.9376	0.9314	0.9475
tweet-cont	0.6413	0.9701	0.7721	0.6685	0.9295
account	0.9907	0.9835	0.9871	0.9851	0.9929
account-lev	0.9481	0.9041	0.9838	0.9159	0.9604
all	**0.9958**	**0.9835**	**0.9896**	**0.9881**	**0.9959**
k_f_reimpl	0.9886	0.9804	0.9845	0.9821	0.9935
k_f-AdaBoost-SMOTEENN	1.00	1.00	1.00	0.9981	0.9981

Table 1: Classification performance of models using different feature sets. k_f_reimpl is our reimplementation of our reference classifier. Though we reimplemented all features we were not able to confirm their original results. These are provided in k_f-AdaBoost-SMOTEENN for reference.

Feature	ANOVA F-value
number_of_tokens_without_hashtags_urls_symbols_median	8255.4
number_of_tokens_median	7843.3
levenshtein_user_name_screen_name	7736.9
number_of_tokens_without_hashtags_urls_symbols_mean	7331.9
number_of_tokens_without_hashtags_urls_symbols_stdev	7157.1
number_of_tokens_stdev	7060.3
number_of_tokens_mean	7007.8
emojis_classic_skewness	5836.8
emojis_other_skewness	5836.8
number_of_tokens_min	4783.9
number_of_tokens_without_hashtags_urls_symbols_min	3236.3
number_of_tokens_without_hashtags_urls_symbols_entropy	2262.7
number_of_tokens_entropy	2257.4
number_of_tokens_without_hashtags_urls_symbols_max	2245.3
number_of_tokens_max	2237.1
number_of_hashtags_skewness	1970.1
special_char_repeats_rate	1829.5
emojis_classic_kurtosis	1558.2
emojis_other_kurtosis	1558.2
statuses_count	1171.2

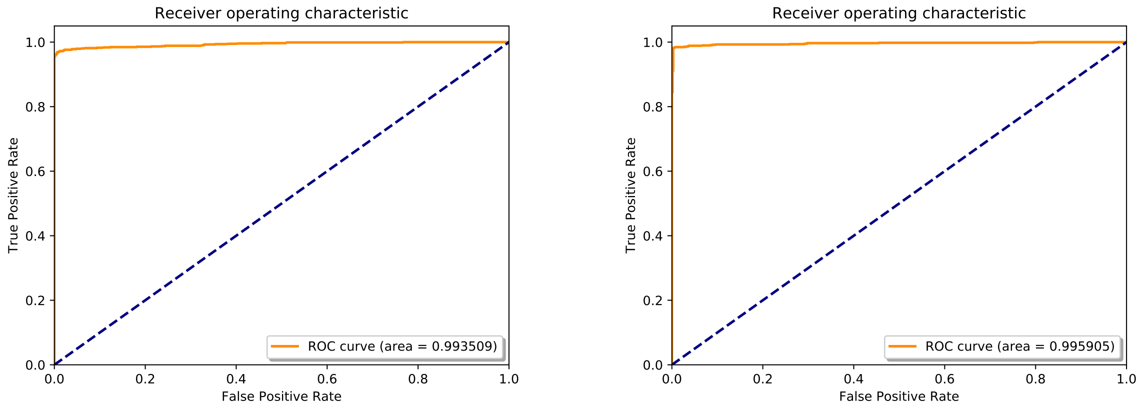
Table 2: The 20 most important features for the system ”all”, sorted by ANOVA F-value.

and **tweet-cont** contains all content features. Our experiments show that neither feature set is able to achieve sufficiently good results, as there is a large number of false positives.

It turned out that using the Levenshtein distance as a feature provides a considerable benefit. It is the most informative feature in the **account** feature set, followed by *geo_enabled*, *statuses_count*, *user_name_length*, *screen_name_length* and features related to account metadata, such as *default_profile*, *favourites_count* and *profile_use_background_image*.

In order to get an idea of how well this feature performs we conducted an experiment using it as the only classifier. We found that by using only this features alone (the **account-lev** feature set) it is possible to achieve an accuracy of 0.8611.

In order to compare our system we reimplemented a classification system as described by Kudugunta and Ferrara, 2018, where AdaBoost with SMOTE-ENN is used to get excellent results. However, although we use the same dataset and the same features, we were not able to achieve the same results. We assume the reason for this is that the exact selection of their training data is unknown to us. For our own analysis we divide the gold standard data using deterministic random selection. Depending on the exact nature of this random selection, it is understandable that each trained system based on this selection will



(a) ROC curve for ”k_f_reimpl”

(b) ROC curve for ”all”

Figure 1: ROC curves

be slightly different and the evaluation results will therefore vary slightly.

With our feature set including features that address the tweet content as well as the the meta data, the unicode groups, the emojis in the content, the user and screen names as well as the the Levenshtein distance between both names we achieved a comparable accuracy of 0.9881 and ROC-AUC value of 0.9959. (For details see table 1.)

6 Further Experiments

In order to estimate the influence of the amount of training data on the quality of such a classifier, we conducted additional experiments and calculated learning curves for several of our systems.

To achieve representative results we use five-fold cross-validation here. Our data set contains the manually tagged data of the 8385 accounts. Therefore 1677 records will be used for validation and 6708 data records remain for training.

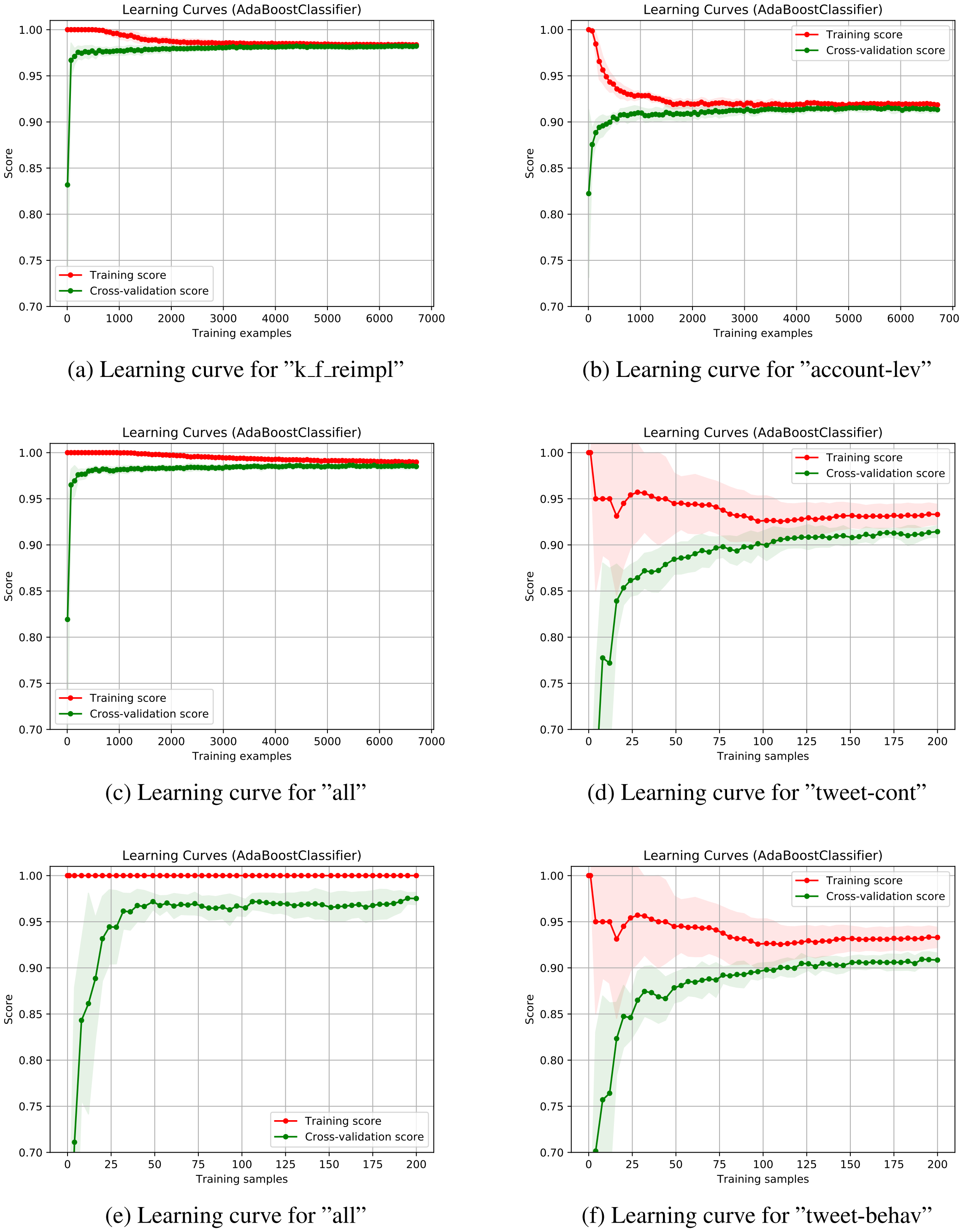
Figure 2 displays learning curves for some of our classifiers, all based on AdaBoost. Targeting one of our research questions understanding if smaller amount of training data could be sufficient for building classifiers we extracted upper left areas of some of our learning curves and present them in figure 2. These results indicate that even if data about only a very limited set of Twitter accounts would be available for training bot detection should be possible though - of course - it would not achieve the very best results. As each account in our data set in average provides 607 tweets this seems to be already a sufficient basis to learn from for practical applications.

7 Conclusion

In this paper we addressed the problem of detecting Twitter bots. For easy reuse of our system we extracted various language-agnostic features including account specific features such as comparing screen name and user names as well as behavioural specific features modeling latent temporal aspects of tweeting. We have shown that with AdaBoost and SMOTE-ENN we can achieve a precision of up to 0.9969, an accuracy of 0.9881 and an AUC-ROC value of 0.9959. Additionally we calculated learning curves of several of our approaches in order to understand the impact a smaller amount of data might have on training of classifiers. We concluded that patterns existing in the data emerge quite soon during training. For practical approaches already a set of a view thousand data records will be sufficient for training. Our data set consisted of 8385 manually classified records. This is sufficient data for training classification systems of good quality.

8 Future Work

Naturally bot developers will pick up on results of researchers and improve their implementations of bots in order to perfect their disguise. Classifiers training on existing data might not work as well in future experiments on real data as bots could produce tweet content in such a way that is misleading for existing classifiers. Tweeting behavior could be modeled in such a way that it is less distinguishable from real users. Future research should focus on addressing these aspects closely.



(a) Learning curve for ”k_f_reimpl”

(b) Learning curve for ”account-lev”

(c) Learning curve for ”all”

(d) Learning curve for ”tweet-cont”

(e) Learning curve for ”all”

(f) Learning curve for ”tweet-behav”

Figure 2: Learning curves for six of our classifiers

References

Adam Badawy, Emilio Ferrara, and Kristina Lerman. 2018. Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign. *arXiv e-prints* page arXiv:1802.04291.

Chiyu Cai, Linjing Li, and Daniel Zeng. 2017. Detecting social bots by jointly modeling deep behavior and content information. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore*. pages 1995–1998.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6):811–824.

Eric M. Clark, Jake Ryland Williams, Chris A. Jones, Richard A. Galbraith, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Sifting Robotic from Organic Text: A Natural Language Approach for Detecting Automation on Twitter. *arXiv e-prints* page arXiv:1505.04342.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake Twitter followers. *Decision Support Systems* 80:56–71.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2017. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia*. pages 963–972.

Phillip George Efthimion, Scott Payne, and Nicholas Proferes. 2018. Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Science Review* 1(2). Article 5.

Yoav Freund and Robert E. Schapire. 1999. A short introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pages 1401–1406.

Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467:312–322.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* 18(17):1–5.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10:707.

Jonas Lundberg, Jonas Nordqvist, and Antonio Matosevic. 2018. On-the-fly Detection of Autogenerated Tweets. *arXiv e-prints* page arXiv:1802.01197.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

A.H. Wang. 2010. Don't follow me: Spam detection in Twitter. In *Proceedings of the 2010 International Conference on Security and Cryptography, Athens, Greece*. pages 1–10.

Multi-level analysis and recognition of the text sentiment on the example of consumer opinions

Jan Kocoń
Wrocław University
of Science and Technology
Wrocław, Poland
jan.kocon
@pwr.edu.pl

Monika Zaśko-Zielińska
University of Wrocław
Institute of Polish Studies
Wrocław, Poland
monika.zasko-zielinska
@uwr.edu.pl

Piotr Miłkowski
Wrocław University
of Science and Technology
Wrocław, Poland
piotr.milkowski
@pwr.edu.pl

Abstract

In this article, we present a novel multi-domain dataset of Polish text reviews, annotated with sentiment on different levels: sentences and the whole documents. The annotation was made by linguists in a 2+1 scheme (with inter-annotator agreement analysis). We present a preliminary approach to the classification of labelled data using logistic regression, bidirectional long short-term memory recurrent neural networks (BiLSTM) and bidirectional encoder representations from transformers (BERT).

1 Introduction

Linguistic research on sentiment recognition involves two approaches: from the perspective of analysing the occurrence of emotional words and from the perspective of the entire document. The first attempt is usually a consequence of the creation of the sentiment lexicon, e.g. manual annotation of the WordNet (Baccianella et al., 2010). The second results from the analysis of the specific text content in which we see that the sentiment of a word or phrase changes under the influence of the surrounding context (Taboada et al., 2008). This change may vary depending on the domain of the text. As a research material for our research we have chosen online customer reviews from four domains:

- *S – School* – students' reviews on the lecturers[1],
- *M – Medicine* – patients' opinions on doctors[2],
- *H – Hotels* – customer reviews of hotels[3],
- *P – Products* – buyers' opinions on products[4].

In the introduction we focus mainly on the influence of discourse on the classification of the document sentiment. We only briefly present an approach based on the analysis of emotional words. The rest of the article concerns the description of the corpus used in our analysis, guidelines for the description of the text with sentiment for annotators, the results of the pilot stage of annotation, the proper annotation and experiments with automatic recognition of the text polarity.

2 Related work

One approach for recognising polarity of text is to use a dictionary of emotional words – sentiment lexicons, e.g. WordNet annotated with polarity (Kamps et al., 2004; Takamura et al., 2005) and emotions (Janz et al., 2017). Usually the task is to determine the number of occurrence of such words with a specific polarity in the text or use a simple bag of words method (Wang and Manning, 2012). Such a solution has a number of limitations: simple methods cannot cope with irony, sarcasm, negation and more complex text structures that modify the sound of the words that make them up (Wallace et al., 2015).

The second method of analysing text polarity is examination of the sentence level with evaluating each sentence in isolation. This procedure can be supported by the external corpus of labelled sentences (Pang and Lee, 2004; Wilson et al., 2005). Nevertheless, even within one sentence we can sometimes observe several features of the analysed entity and each of them can be assessed dif-

[1] https://polwro.pl/
[2] https://www.znanylekarz.pl/
[3] https://pl.tripadvisor.com/
[4] https://www.ceneo.pl/

ferently. It is advantageous in opinion mining to achieve both an overall opinion and specific information on the reviewed entity and its aspects. It gives us not only a general opinion on the product but allows us to notice a detailed view on the quality of the product.

This year, the first results of the Sentimenti[5] project were published, which aimed to create methods of analysing the content written on the Internet in terms of emotions expressed by the authors of the texts and the emotional impact of the readers. Within the project, a large database has been created, in which 30,000 lexical units from plWordNet database and 7,000 texts were evaluated, most of which are consumer reviews from the domain of hotels and medicine. The elements were evaluated by 20,000 unique Polish respondents in the Computer Assisted Personal Interview survey and more than 50 marks were obtained for each element, which gives more than 1.8 million annotations. Within each mark, polarisation of the element, stimulation and basic emotions aroused by the recipients are determined. The first results concerning the automatic recognition of sound and emotions for this set are presented in (Kocoń et al., 2019). Our article is based on this work in the development of experiments and we are researching texts from similar domains, but using more complex classification methods as described in Section 4. The annotation guidelines for linguists in the task of sentiment analysis on two levels were also developed: text level and sentence level (presented in Section 3).

2.1 From word level to discourse in text polarity analysis

A discourse perspective in sentiment analysis is an attempt to address limitations of previous methods (e.g. problems with negation, focusing on adjectives). It used findings of Rhetorical Structure Theory (Mann and Thompson, 1988). The attempt bears in mind local and global orientation in the text, discourse structure or topicality (Taboada et al., 2008). It allows the researcher to extract the most important sentences from the text in the perspective of the entire discourse context: nucleus satellite method (Wang et al., 2012). The relevance of the sentences is evaluated in relation to the main topic and the analysis omits some less important parts of the text. In Section 3 we present how the genre structure of a customer review affects the text sentiment polarity. It is an enhancement of the discourse perspective in sentiment analysis.

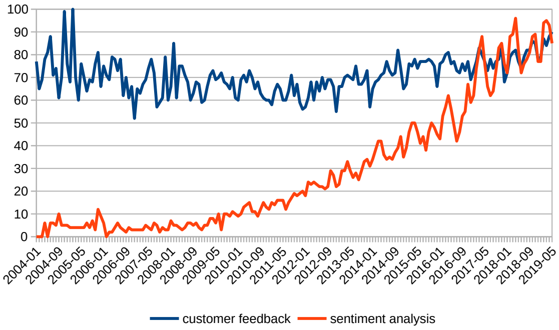


Figure 1: Google Trends (trends.google.com) data showing interest in time for search terms "customer feedback" and "sentiment analysis". On the vertical axis 100 means biggest search term popularity.

2.2 Recognition of sentiment

Sentiment analysis and opinion mining has become an interesting topic for many researches and private companies with constant growth of interest in recent years (see Figure 1), that coincides with the *big data revolution* (Kitchin, 2014). Properly evaluated data can be widely used in fields such as market analysis, public relations and product or customer feedback. Main difficulty in retrieval of those information is non-organised data, unstructured in pre-defined manner. Recently deep neural networks show relatively good performance among all available methods of processing such information (Glorot et al., 2011). Possibility of retrieving data from different sources like social networks (Pak and Paroubek, 2010), publicly available discussion boards or various marketing platforms connected with proper annotations on training data set can provide not only simple positive, negative or neutral classification but lead to accurate fine-grained sentiment prediction (Guzman and Maalej, 2014). In Section 4 we present our approach to solve this task using models based on fastText (Joulin et al., 2017), BiLSTM (Zhou et al., 2016) and BERT (Devlin et al., 2018).

3 Annotation model

Our research on sentiment analysis of customer reviews was conducted in 2018 within CLARIN-PL and it consisted of the pilot stage and the main stage. The preliminary part of analysis involved 3,000 students' opinions about lectures. It is an

[5] https://sentimenti.com/

authentic material provided online by students as a final assessment of the course in each subject. Each text was manually annotated by two annotators: a psychologist and a linguist, who worked accordingly to the general guidelines. In the pilot project we decided to deal with sentiment annotation of the whole text. In the sentiment annotation we used the same set of tags which is applied in *plWordNet 3.0 emo* (Zaśko-Zielińska et al., 2015; Janz et al., 2017) to lexical units: [+m] – strong positive, [+s] – weak positive, [-m] – strong negative, [-s] – weak negative, [amb] – ambiguous, [0] – neutral.

We used *amb* tag but we understood it differently. In annotation of lexical units in WordNet with sentiment *amb* indicated the possibility that units can be positive or negative in various contexts. Hence, in text sentiment analysis we assumed that *amb* denotes *ambiguous* polarity, thus the entire text cannot be clearly described by using neither positive nor negative annotation.

In the annotation we focused primarily on the strategic places in the text. In a customer review these places are the opening and closing sentences, namely the text frame. The beginning consists of the general opinion of the author on the subject of the evaluation and the end includes the author's recommendation to the recipients. The annotators created their first general evaluation based on these two segments. In the body of the review, the authors have only subtly changed these opinions. Regardless of the modification of the main opinion in the text, we did not use the *amb* tag when the text frame was unambiguously positive. The text frame polarity was influenced not only by lexical content but also nonverbal elements, e.g. emoticons or multiplication of punctuation marks.

3.1 An attempt to annotate aspects

The analysis of the content of customer reviews in our pilot project consisted of two stages: the selection of text blocks describing separate aspects and their annotation. Some parts of the text were not of an argumentative nature that could justify the author's decision to polarise the text. They included: advice (e.g. how to sign up for lectures) or general information on lectures, duration of classes, etc.

The main stage of our project was conducted based on text corpus consisting of consumer reviews (80% of texts) and texts from the corresponding domain with high probability of neutral polarity (20% of texts). We observed that the value of inter-annotator agreement in aspect annotation task was very low, below 15% of Positive Specific Agreement, PSA (Hripcsak and Rothschild, 2005).

3.2 New annotation guidelines

In the main stage of the project we decided to annotate the sentiment for the whole text (a *meta* level) and the *sentence* level. We assumed that this strategy allows to establish the acceptable value of PSA. We followed the rule that the *meta* annotation results partially from sentence annotations, however the frame polarity is the main factor for the final meta annotation. We have prepared the following rules of annotation, regardless of whether the entire text or sentence is annotated:

- SP – strong positive – entirely positive;
- WP – weak positive – generally positive, but there are some negative aspects;
- 0 – neutral;
- WN – weak negative – generally negative, but there are some positive aspects;
- SN – strong negative – entirely negative;
- AMB – ambiguous – there are both positive and negative aspects in the text that are balanced in terms of relevance.

Table 1 shows the value of Positive Specific Agreement (Hripcsak and Rothschild, 2005) obtained for a random sample of 111 documents from *Medicine* category.

4 Multi-level sentiment recognition

We selected three different classifiers for the recognition tasks:

- logistic regression (fastText) providing a baseline for text classification (Joulin et al., 2017)
- bidirectional long short-term memory recurrent network in two variants:
  - using word vector representations only
  - using the same vectors extended with general polarity information from sentiment dictionary described in Section 4.1

L	Type	Only A	A & B	Only B	PSA
M	SN	1	33	4	93%
	WN	2	2	2	50%
	0	0	24	0	100%
	AMB	1	2	3	50%
	WP	4	0	0	0%
	SP	0	31	2	97%
	sum	8	92	11	91%
S	SN	10	217	36	90%
	WN	11	1	0	15%
	0	36	273	17	91%
	AMB	2	7	14	47%
	WP	12	0	1	0%
	SP	6	194	8	97%
	sum	77	692	76	90%

Table 1: Annotation agreement between two experts (A and B) at the level (L) of text (meta – M) and sentence (S) for a sample of 111 documents using Positive Specific Agreement metric, PSA (Hripcsak and Rothschild, 2005).

- bidirectional encoder representations from transformers (BERT) with addition of sequence classification layer

We trained logistic regression model using pre-trained vectors for Polish language (Kocoń and Gawor, 2018). This approach is much faster in both training and testing than deep learning classifiers (Joulin et al., 2017), however, it has disadvantage which comes from not sharing parameters by features and classes, therefore overall result can be highly influenced by keywords with bigger class relativity.

BiLSTM on the other side takes into consideration not just words but full text fragment and basing on learnt patterns predicts potential outcome. Texts are divided into tokens and converted to corresponding word embedding vectors generated by fastText (Bojanowski et al., 2017), in this form it is possible to use it as input for neural network. Dimension of used vectors is equal to 300, therefore it must be reflected in the input shape. As a loss function for training a categorical crossentropy was chosen. Model prepared for the task consists of the following layers:

- Gaussian noise layer with standard deviation of 0.01 accepting as input shape up to 128 words with vector matrix for each word of size 300, therefore overall input shape is (128, 300)
- Bidirectional layer with LSTM instances (consisting of 1,024 hidden units using hyperbolic tangent activation method) merged with concatenation
- Dropout layer with dropout ratio equal to 0.2
- Dense layer with number of outputs representing number of all possible labels (6 in our task) using normalised exponential function (softmax) activation

BERT was designed to provide pre-trained deep bidirectional representations conditioning left and right context (Devlin et al., 2018), therefore it achieves best performance on text fragments instead of single sentences. It's architecture allows to fine-tune these representations by adding one additional output layer which suits needs of specified task. For our task as a pre-trained model `BERT-Base, Multilingual Cased`[6] was selected, which consists of 104 languages and 110M parameters, and `BertForSequenceClassification`[7] as a BERT classifier extended for multi-class classification.

4.1 Embedding vector extension

Basing on the data accommodated in plWordNet emo (Zaśko-Zielińska et al., 2015) we prepared the dictionary for all annotated lexical units and all possible levels of sentiment. Due to the lack of word sense disambiguation method, we grouped the sentiment annotations by lemmas. The final dictionary consists of a set of lemmas with assigned numbers representing the proportions of individual sentiment annotations, summing up to 1, e.g. for a lemma *akademicki* (Eng. *academic*) there were 11 annotations: 3 neutral, 4 generally negative, 3 generally positive and 1 entirely positive. Therefore arbitrary values for word "akademicki" are:

- entirely positive = 0.0909
- generally positive = 0.2727
- neutral = 0.2727
- generally negative = 0.3636
- entirely negative = 0.0000

[6] https://github.com/google-research/bert

[7] https://github.com/huggingface/pytorch-pretrained-BERT

- ambivalent = 0.0000

Using the described dictionary we have proposed additional variant of BiLSTM classifier with a word embedding vector extended with the values of sentiment for the lemma of the word from a prepared sentiment dictionary. Lemmas were retrieved during a preparation of the input data using WCRFT part-of-speech tagger (Radziszewski, 2013). Therefore, in this approach the input word vector dimension was extended with 6 values representing sentiment of the word. The final dimension of the word embedding increased from 300 to 306.

5 Evaluation

As in article (Kocoń et al., 2019), three variants of evaluation of the sentiment classification methods were prepared. The basic variant is a single domain in which the classifier is trained, tuned and tested on a set of texts from one domain. The next variant includes an analysis of the ability of the classifier to model the sentiment of the text on a level independent of the domain of the text. For this purpose, we take all available texts except the texts from the selected domain. Then the texts are divided into a training and a validation set. Testing of the model takes place on a test set from a selected domain, not taken into account at the stage of preparing the training and validation set. The third test variant allows to examine the classifiers in order to generalise the task of sentiment analysis in all available domains. For this purpose, texts from all domains are treated as one set, which is randomly divided into train, validation and test sets. Summary of the different types of evaluation:

- *SD – Single Domain* – evaluation sets created using elements from the same domain,

- *DO – Domain Out* – train/dev sets created using elements from 3 domains, test set from the remaining domain,

- *MD – Mixed Domains* – evaluation sets randomly selected from elements belonging to all domains.

Due to the fact that the data are annotated both at the level of the whole text and at the level of each sentence, a *sentence* or *text* may be an *element* in the above list. We use *SDT*, *DOT*, and *MDT* for *text* evaluation types and *SDS*, *DOS*, and *MDS* for *sentence* evaluation types. We use also prefixes of domains (*Hotels, Medicine, School, Products*) as suffixes for *SD** and *DO** variants, e.g. SDS-H is a single domain evaluation type performed on sentences within *hotels* domain, whereas DOT-M is a domain-out evaluation type performed on texts trained on texts outside *medicine* domain and tested on texts from that domain.

Table 2 shows the number of texts and sentences annotated by linguists for all evaluation types, with division into the number of elements within training, validation and test sets. Linguists annotated a total of 8,450 texts from four domains (hotels, medicine, products, school) and 35,789 sentences from two domains (hotels, medicine). The distribution of labels within each domain for texts and sentences is presented in Table 3. Average annotated text length in each domain are as follows: 788 characters in hotels, 802 in medicine, 781 in products and 442 in school.

Type	Domain	Train	Dev	Test	SUM
SDT	Hotels	2534	316	316	3166
	Medicine	2650	330	330	3310
	Products	790	98	98	986
	School	792	98	98	988
DOT	!Hotels	4756	528	-	5284
	!Medicine	4635	514	-	5149
	!Products	6727	746	-	7473
	!School	6725	746	-	7471
MDT	All	6771	846	845	8462
SDS	Hotels	12434	1554	1553	15541
	Medicine	16200	2024	2024	20248
DOS	!Hotels	16200	2024	-	18224
	!Medicine	12434	1554	-	13988
MDS	All	28581	3572	3571	35724

Table 2: The number of texts/sentences for each evaluation type in train/dev/test sets.

Type	Domain	SP	WP	0	WN	SN	AMB
SDT	Hotels	25.80	10.77	11.24	05.87	38.68	07.64
	Medicine	29.48	02.87	23.98	02.33	36.94	04.41
	Products	22.70	15.40	00.20	08.31	44.28	09.12
	School	46.92	26.19	00.20	07.99	10.11	08.59
SDS	Hotels	34.58	00.01	18.72	00.01	44.31	02.38
	Medicine	24.78	00.31	40.68	00.46	32.63	01.14

Table 3: The percentage of annotated elements in a given domain (SDT – single domain texts, SDS – single domain sentences).

6 Results

Table 4 presents the values of F1-score for each label (columns 3-8), global F1-score (column 9), micro-AUC and macro-AUC (columns 10-11) for all evaluation types related to the texts. In case of evaluation for a single domain for each label, fastText (using Logistic Regression) outperformed other classifiers in 13 out of 21 distinguishable cases. There are 12 cases for which the best score is not higher than F1=0.4. These are highly underrepresented labels, for which the part of the total annotations within the domain is less than 10% (see Table 3). The best results are obtained for *strong positive* and *strong negative* cases. Intermediate labels (*weak* and *ambiguous* variants) are much more difficult to be recognised correctly. In these cases deep neural networks outperform logisitic regression in 6 out of 11 cases. BERT classifier performs much better (13 out of 23 cases) in cross-domain knowledge transfer (DOT and MDT). For these evaluation types only 6 times fastText was better. These observations are consistent with the results of article (Kocoń et al., 2019) for *valence* dimensions.

Table 5 presents results corresponding to those presented in Table 4, but this time for sentence-level annotations. Looking at Table 3, the number of sentences marked as *weakly positive* or *weakly negative* is close to zero. These labels are not being recognised by any classifier. For other labels, regardless of the type of evaluation, the best results are mainly obtained using deep learning methods (label-specific F1-score: all 20 cases; general metrics: 12 out of 15 cases).

7 Conclusions and further steps

The automatic annotation of emotions has both a scientific and an applied value. Modern business is interested in the opinions, emotions and values associated with brands and products. Retailers and merchants collect huge amounts of customer feedback from the store and online. Moreover, the relationship departments monitor the impact of their campaigns and need to know if it was positive and affecting customers. In this context, the results of monitoring feedback, reactions and emotions are of great value as they fuel decisions and behaviors (Tversky and Kahneman, 1989). However, most of the existing solutions are still limited to manual annotations and simplified analysis methods.

Type	Classifier	SP	WP	0	WN	SN	AMB	F1	micro	macro
SDT-H	C1	**80.00**	**30.51**	93.98	00.00	**83.33**	**36.84**	**73.50**	90.87	70.20
	C2	71.11	25.00	00.00	04.76	72.44	00.00	53.00	77.14	66.13
	C3	72.82	22.95	94.12	**14.81**	81.98	27.27	68.45	89.83	72.44
	C4	71.22	10.26	**96.39**	00.00	78.16	00.00	68.45	**91.30**	**72.41**
SDT-M	C1	81.05	**15.38**	**96.39**	00.00	**80.63**	00.00	**78.55**	93.44	66.39
	C2	78.69	11.11	95.71	**14.29**	80.31	06.67	77.04	**94.02**	**71.81**
	C3	**81.93**	13.33	95.71	13.33	80.43	**07.41**	78.55	91.81	69.33
	C4	00.00	00.00	95.65	00.00	62.33	00.00	58.01	89.42	59.73
SDT-P	C1	**62.86**	27.59	00.00	**36.36**	**84.68**	16.67	**65.66**	**86.76**	**63.58**
	C2	28.57	**30.30**	00.00	00.00	69.16	**26.67**	49.49	78.37	53.46
	C3	25.00	00.00	00.00	00.00	67.26	00.00	43.43	77.64	51.46
	C4	00.00	00.00	00.00	00.00	69.74	00.00	53.54	80.40	42.60
SDT-S	C1	**79.61**	**52.63**	00.00	00.00	**50.00**	00.00	**61.62**	83.80	**62.33**
	C2	72.22	33.33	00.00	00.00	27.27	**36.36**	52.53	80.39	54.40
	C3	75.68	34.04	00.00	00.00	11.76	00.00	51.52	79.71	54.67
	C4	68.87	00.00	00.00	00.00	00.00	00.00	52.53	**83.88**	49.55
DOT-H	C1	70.91	23.08	**95.24**	00.00	**83.49**	**21.05**	**69.09**	88.21	66.34
	C2	72.73	17.02	91.76	**15.38**	78.76	16.00	65.30	88.31	71.21
	C2	73.94	19.67	88.89	10.00	75.11	16.00	62.46	87.41	70.59
	C4	**75.53**	**34.09**	90.67	00.00	82.76	00.00	68.14	**91.47**	**72.70**
DOT-M	C1	72.51	08.70	86.67	17.39	75.29	00.00	69.18	86.13	68.37
	C2	**73.17**	**22.22**	85.14	**28.57**	76.79	**16.33**	68.28	89.99	71.46
	C3	46.01	03.48	00.00	00.00	00.00	00.00	22.36	59.76	51.52
	C4	72.11	10.71	**90.32**	16.67	**85.06**	12.90	**72.51**	**90.88**	**73.14**
DOT-P	C1	60.61	47.06	00.00	**50.00**	76.00	00.00	59.60	89.71	70.29
	C2	63.16	26.09	00.00	22.22	84.91	11.76	62.63	83.19	61.73
	C3	54.55	28.57	00.00	00.00	85.71	00.00	59.18	83.81	63.25
	C4	**70.97**	**62.07**	00.00	00.00	**92.73**	**28.57**	**73.74**	**90.77**	**71.38**
DOT-S	C1	68.00	12.50	00.00	**13.33**	36.36	00.00	43.43	**90.12**	**68.67**
	C2	73.58	18.75	00.00	00.00	37.84	33.33	**51.52**	79.35	64.91
	C3	**75.25**	24.24	00.00	00.00	**41.03**	22.22	**51.52**	73.71	60.23
	C4	70.83	**29.27**	00.00	00.00	34.48	**37.50**	48.48	75.64	59.62
MDT	C1	83.20	40.27	**97.14**	10.91	85.28	17.72	76.83	88.92	68.04
	C2	81.21	41.03	96.75	09.68	83.36	21.57	74.35	92.90	74.79
	C3	81.82	00.00	96.39	**10.96**	80.75	**27.64**	72.70	87.67	74.19
	C4	**86.12**	**50.00**	94.65	00.00	**86.87**	22.86	**77.78**	**95.78**	**78.85**

Table 4: F1-scores for text-oriented evaluation. Training sets for evaluation types are the same as in Table 2 rows 1-9. Classifiers: C1 - fastText, C2 - BiLSTM, C3 - BiLSTM with word embeddings extended using polarity dictionary, C4 - BERT. Evaluation types are explained in Section 5.

Type	Classifier	SP	WP	0	WN	SN	AMB	F1	micro	macro
SDS-H	C1	85.64	00.00	77.54	00.00	83.59	16.44	81.60	94.39	65.19
	C2	86.53	00.00	82.15	00.00	**88.73**	31.71	85.20	**97.88**	**70.58**
	C3	**88.89**	00.00	**82.58**	00.00	88.09	35.71	**85.91**	97.54	69.46
	C4	87.66	00.00	82.47	00.00	87.99	**42.86**	85.39	97.26	70.32
SDS-M	C1	70.68	00.00	76.36	00.00	70.14	15.38	71.85	89.45	63.76
	C2	**78.01**	00.00	80.35	00.00	**75.30**	07.14	77.19	**95.54**	74.21
	C3	72.86	**18.18**	78.88	00.00	74.66	**25.64**	75.06	94.76	72.98
	C4	76.79	00.00	**81.08**	00.00	75.25	00.00	**77.33**	94.99	**74.76**
DOS-H	C1	60.11	00.00	48.83	00.00	61.30	00.00	55.53	**89.46**	63.81
	C2	**69.56**	00.00	54.45	00.00	67.98	06.15	62.87	87.99	**70.85**
	C3	72.05	00.00	**56.16**	00.00	**68.98**	**06.35**	**64.93**	88.48	69.50
	C4	65.46	00.00	51.74	00.00	60.85	00.00	58.04	86.23	68.04
DOS-M	C1	53.08	00.00	63.29	00.00	64.45	20.69	61.19	**94.49**	**65.29**
	C2	58.30	00.00	**67.40**	00.00	**69.37**	**37.84**	**65.98**	90.43	64.14
	C3	59.54	00.00	66.11	00.00	68.58	20.69	65.28	89.50	60.60
	C4	**61.10**	00.00	65.95	00.00	67.84	16.67	65.09	89.35	59.50
MDS	C1	77.14	00.00	76.43	00.00	76.06	17.82	75.25	89.60	61.60
	C2	**84.37**	00.00	82.30	00.00	82.78	**35.09**	82.11	96.66	**75.11**
	C3	76.00	00.00	77.29	00.00	76.54	16.22	75.42	94.98	70.77
	C4	84.14	00.00	**83.39**	00.00	**83.52**	25.50	**82.28**	**96.83**	74.25

Table 5: F1-scores for sentence-oriented evaluation. Training sets for evaluation types are the same as in Table 2 rows 10-14. Classifiers: C1 - fastText, C2 - BiLSTM, C3 - BiLSTM with word embeddings extended using polarity dictionary, C4 - BERT. Evaluation types are explained in Section 5.

BERT's performance is below the expectations of this advanced method. Looking at both tables (4 and 5), BERT's results are the best in 19 out of 69 label-specific cases, which is exactly as many as fastText was. BiLSTM outperformed other methods in 31 cases. Adding an external sentiment dictionary helped only in 14 label-specific cases. BERT dominance is observed in DOT and MDT cases, especially when analysing general metric values, where the predominance of the method is visible in 11 out of 15 cases. The advantage is repeated for MDS but not for DOS. MDT case is the most promising in terms of the further use of the recognition method in applications such as brand monitoring or early crisis detection. Figure 2 shows the ROC curves (Meistrell, 1990) for this case. The values of the general F1, micro AUC and macro AUC are the highest for the BERT method (see Table 2).

We plan to publish the data created as part of the presented works on an open license soon. We also intend to test the contextualized embedding that we are currently building using the ELMo deep word representations method (Peters et al., 2018), with the use of the large KGR10 corpus presented in work (Kocoń et al., 2019). We also want to train the basic BERT model with the use of KGR10 to investigate whether it will improve the quality of sentiment recognition. It is also very interesting to use the propagation of sentiment annotation in WordNet (Kocoń et al., 2018a,b), to increase the coverage of the sentiment dictionary and to potentially improve the recognition quality as well. This objective can be achieved by other complex methods such as OpenAI GPT-2 (Radford et al., 2019) and domain dictionaries construction methods utilising WordNet (Kocoń and Marcińczuk, 2016).

Acknowledgements

Co-financed by the Polish Ministry of Education and Science, CLARIN-PL Project and by the National Centre for Research and Development, Poland, grant no POIR.01.01.01-00-0472/16 – *Sentimenti*[8].

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical

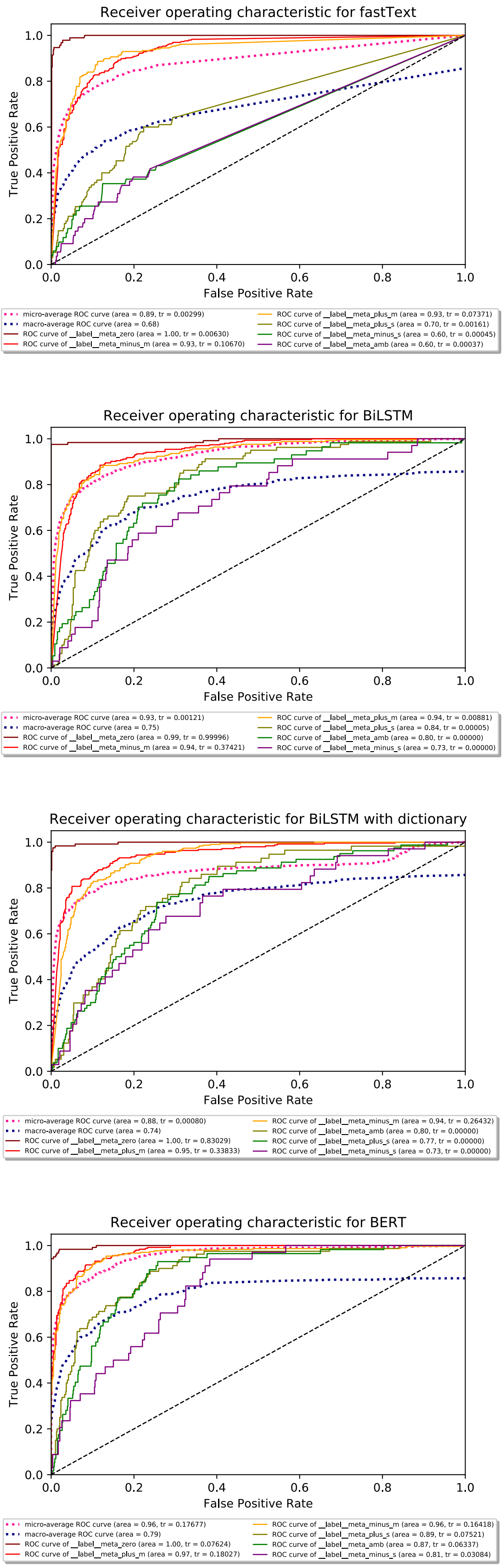


Figure 2: Receiver Operating Characteristic curves of all used classifiers for Mixed Domain Text setting.

[8] http://w3a.pl/projekty/

resource for sentiment analysis and opinion mining. In *LREC*. volume 10, pages 2200–2204.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 513–520.

Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*. IEEE, pages 153–162.

George Hripcsak and Adam S. Rothschild. 2005. Technical Brief: Agreement, the F-Measure, and Reliability in Information Retrieval. *JAMIA* 12(3):296–298. https://doi.org/10.1197/jamia.M1733.

Arkadiusz Janz, Jan Kocoń, Maciej Piasecki, and Monika Zaśko-Zielińska. 2017. plWordNet as a Basis for Large Emotive Lexicons of Polish. In *LTC'17 8th Language and Technology Conference*. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu, Poznań, Poland.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 427–431.

Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. 2004. Using wordnet to measure semantic orientations of adjectives. In *LREC*. Citeseer, volume 4, pages 1115–1118.

Rob Kitchin. 2014. *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage.

Jan Kocoń and Michał Gawor. 2018. Evaluating KGR10 Polish word embeddings in the recognition of temporal expressions using BiLSTM-CRF. *Schedae Informaticae* 27. https://doi.org/10.4467/20838476SI.18.008.10413.

Jan Kocoń, Arkadiusz Janz, and Maciej Piasecki. 2018a. Classifier-based Polarity Propagation in a Wordnet. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*.

Jan Kocoń, Arkadiusz Janz, and Maciej Piasecki. 2018b. Context-sensitive Sentiment Propagation in WordNet. In *Proceedings of the 9th International Global Wordnet Conference (GWC'18)*.

Jan Kocoń, Arkadiusz Janz, Miłkowski Piotr, Monika Riegel, Małgorzata Wierzba, Artur Marchewka, Agnieszka Czoska, Damian Grimling, Barbara Konat, Konrad Juszczyk, Katarzyna Klessa, and Maciej Piasecki. 2019. Recognition of emotions, polarity and arousal in large-scale multi-domain text reviews. In Zygmunt Vetulani and Patrick Paroubek, editors, *Human Language Technologies as a Challenge for Computer Science and Linguistics*, Wydawnictwo Nauka i Innowacje, Poznań, Poland, pages 274–280.

Jan Kocoń and Michał Marcińczuk. 2016. Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents. In *Text, Speech and Dialogue, Proceedings of the 19th International Conference TSD 2016*. Springer, Brno, Czech Republic, volume 9924 of *Lecture Notes in Artificial Intelligence*.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse* 8(3):243–281.

Michael L Meistrell. 1990. Evaluation of neural network performance by receiver operating characteristic (roc) analysis: examples from the biotechnology domain. *Computer Methods and Programs in Biomedicine* 32(1):73–80.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*. volume 10, pages 1320–1326.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* page 8.

Adam Radziszewski. 2013. A tiered CRF tagger for Polish. In *Intelligent tools for building a scientific information platform*, Springer, pages 215–230.

Maite Taboada, Kimberly Voll, and Julian Brooke. 2008. Extracting sentiment as a function of discourse structure and topicality. *Simon Fraser Univeristy School of Computing Science Technical Report* .

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 133–140.

Amos Tversky and Daniel Kahneman. 1989. Rational choice and the framing of decisions. In *Multiple Criteria Decision Making and Risk Analysis Using Microcomputers*, Springer, pages 81–126.

Byron C Wallace, Eugene Charniak, et al. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. volume 1, pages 1035–1044.

Fei Wang, Yunfang Wu, and Likun Qiu. 2012. Exploiting discourse relations for sentiment analysis. *Proceedings of COLING 2012: Posters* pages 1311–1320.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. Association for Computational Linguistics, pages 90–94.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Monika Zaśko-Zielińska, Maciej Piasecki, and Stan Szpakowicz. 2015. A large wordnet-based sentiment lexicon for polish. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. pages 721–730.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 207–212.

A Qualitative Evaluation Framework for Paraphrase Identification

Venelin Kovatchev[1,3], M. Antònia Martí[1,3], Maria Salamó[2,3], Javier Beltran[1,3]
[1]Facultat de Filología, Universitat de Barcelona
[2] Facultat de Matemàtiques i Informàtica, Universitat de Barcelona
[3]Universitat de Barcelona Institute of Complex Systems
Gran Vía de les Corts Catalanes, 585, 08007 Barcelona, Spain
{vkovatchev, amarti, maria.salamo, javier.beltran}@ub.edu

Abstract

In this paper, we present a new approach for the evaluation, error analysis, and interpretation of supervised and unsupervised Paraphrase Identification (PI) systems. Our evaluation framework makes use of a PI corpus annotated with linguistic phenomena to provide a better understanding and interpretation of the performance of various PI systems. Our approach allows for a qualitative evaluation and comparison of the PI models using human interpretable categories. It does not require modification of the training objective of the systems and does not place additional burden on the developers. We replicate several popular supervised and unsupervised PI systems. Using our evaluation framework we show that: 1) Each system performs differently with respect to a set of linguistic phenomena and makes qualitatively different kinds of errors; 2) Some linguistic phenomena are more challenging than others across all systems.

1 Introduction

In this paper we propose a new approach to evaluation, error analysis and interpretation in the task of Paraphrase Identification (PI). Typically, PI is defined as comparing two texts of arbitrary size in order to determine whether they have approximately the same meaning (Dolan et al., 2004). The two texts in 1a and 1b are considered paraphrases, while the two texts at 2a and 2b are non-paraphrases.[1] In 1a and 1b there is a change in the wording (*"magistrate"* - *"judge"*) and the syntactic structure (*"was ordered"* - *"ordered"*) but the meaning of the sentences is unchanged. In 2a and 2b there are significant differences in the quantities (*"5%"* - *"4.7%"* and *"$27.45"* - *"$27.54"*).

1a A federal magistrate in Fort Lauderdale ordered him held without bail.

1b He was ordered held without bail Wednesday by a federal judge in Fort Lauderdale, Fla.

2a Microsoft fell **5 percent** before the open to **$27.45** from Thursday's close of $28.91.

2b Shares in Microsoft slipped **4.7 percent** in after-hours trade to **$27.54** from a Nasdaq close of $28.91.

The task of PI can be framed as a binary classification problem. The performance of the different PI systems is reported using the Accuracy and F1 score measures. However this form of evaluation does not facilitate the interpretation and error analysis of the participating systems. Given the Deep Learning nature of most of the state-of-the-art systems and the complexity of the PI task, we argue that better means for evaluation, interpretation, and error analysis are needed. We propose a new evaluation methodology to address this gap in the field. We demonstrate our methodology on the ETPC corpus (Kovatchev et al., 2018a) - a recently published corpus, annotated with detailed linguistic phenomena involved in paraphrasing.

We replicate several popular state-of-the-art Supervised and Unsupervised PI Systems and demonstrate the advantages of our evaluation methodology by analyzing and comparing their performance. We show that while the systems obtain similar quantitative results (Accuracy and F1), they perform differently with respect to a set of human interpretable linguistic categories and make qualitatively different kinds of errors. We also show that some of the categories are more challenging than others across all evaluated systems.

2 Related Work

The systems that compete on PI range from using hand-crafted features and Machine Learning algorithms (Fernando and Stevenson, 2008; Madnani

[1] Examples are from the MRPC corpus (Dolan et al., 2004)

et al., 2012; Ji and Eisenstein, 2013) to end-to-end Deep Learning models (He et al., 2015; He and Lin, 2016; Wang et al., 2016; Lan and Xu, 2018a; Kiros et al., 2015; Conneau et al., 2017). The PI systems are typically divided in two groups: Supervised PI systems and Unsupervised PI systems.

"Supervised PI systems" (He et al., 2015; He and Lin, 2016; Wang et al., 2016; Lan and Xu, 2018a) are explicitly trained for the PI task on a PI corpora. "Unsupervised PI systems" in the PI field is a term used for systems that use a general purpose sentence representations such as Mikolov et al. (2013); Pennington et al. (2014); Kiros et al. (2015); Conneau et al. (2017). To predict the paraphrasing relation, they can compare the sentence representations of the candidate paraphrases directly (ex.: cosine of the angle), and use a PI corpus to learn a threshold. Alternatively they can use the representations as features in a classifier.

The complexity of paraphrasing has been emphasized by many researchers (Bhagat and Hovy, 2013; Vila et al., 2014; Benikova and Zesch, 2017). Similar observations have been made for Textual Entailment (Sammons et al., 2010; Cabrio and Magnini, 2014). Gold et al. (2019) study the interactions between paraphrasing and entailment.

Despite the complexity of the phenomena, the popular PI corpora (Dolan et al., 2004; Ganitkevitch et al., 2013; Iyer et al., 2017; Lan et al., 2017) are annotated in a binary manner. In part it is due to lack of annotation tools capable of fine-grained annotation of relations. WARP-Text (Kovatchev et al., 2018b) fills this gap in the NLP toolbox.

The simplified corpus format poses a problem with respect to the quality of the PI task and the ways it can be evaluated. The vast majority of the state-of-the-art systems in PI provide no or very little error analysis. This makes it difficult to interpret the actual capabilities of a system and its applicability to other corpora and tasks.

Some researchers have approached the problem of non-interpretability by evaluating the same architecture on multiple datasets and multiple tasks. Lan and Xu (2018b) apply this approach to Supervised PI systems, while Aldarmaki and Diab (2018) use it for evaluating Unsupervised PI systems and general sentence representation models.

Linzen et al. (2016) demonstrate how by modifying the task definition and the evaluation the capabilities of a Deep Learning system can be determined implicitly. The main advantage of such an approach is that it only requires modification and additional annotation of the corpus. It does not place any additional burden on the developers of the systems and can be applied to multiple systems without additional cost.

We follow a similar line of research and propose a new evaluation that uses ETPC (Kovatchev et al., 2018a): a PI corpus with a multi-layer annotation of various linguistic phenomena. Our methodology uses the corpus annotation to provide much more feedback to the competing systems and to evaluate and compare them qualitatively.

3 Qualitative Evaluation Framework

3.1 The ETPC Corpus

ETPC (Kovatchev et al., 2018a) is a re-annotated version of the MRPC corpus. It contains 5,801 text pairs. Each text pair in ETPC has two separate layers of annotation. The first layer contains the traditional binary label (paraphrase or non-paraphrase) of every text pair. The second layer contains the annotation of 27 *"atomic"* linguistic phenomena involved in paraphrasing, according to the authors of the corpus. All phenomena are linguistically motivated and humanly interpretable.

3a A federal **magistrate** in Fort Lauderdale ordered him held without bail.

3b He was ordered held without bail Wednesday by a federal **judge** in Fort Lauderdale, Fla.

We illustrate the annotation with examples 3a and 3b. At the binary level, this pair is annotated as "paraphrases". At the "atomic" level, ETPC contains the annotation of multiple phenomena, such as the *"same polarity substitution (habitual)"* of "magistrate" and "judge" (marked **bold**) or the *"diathesis alternation"* of *"...ordered him held"* and *"he was ordered by..."* (marked underline).

For the full set of phenomena, the linguistic reasoning behind them, their frequency in the corpus, real examples from the pairs, and the annotation guidelines, please refer to Kovatchev et al. (2018a).

3.2 Evaluation Methodology

We use the corpus to evaluate the capabilities of the different PI systems implicitly. That means, the training objective of the systems remains unchanged: they are required to correctly predict the value of the binary label at the first annotation layer. However, when we analyze and evaluate the performance of the systems, we make use of both the binary and the atomic annotation layers. Our evaluation framework is created to address our main research question (RQ 1):

RQ 1 Does the performance of a PI system on each candidate-paraphrase pair depend on the different phenomena involved in that pair?

We evaluate the performance of the systems in terms of their *"overall performance"* (Accuracy and F1) and *"phenomena performance"*.

"Phenomena performance" is a novelty of our approach and allows for qualitative analysis and comparison. To calculate *"phenomena performance"*, we create 27 subsets of the test set, one for each linguistic phenomenon. Each of the subsets consists of all text pairs that contain the corresponding phenomenon[2]. Then, we use each of the 27 subsets as a test set and we calculate the binary classification Accuracy (paraphrase or non-paraphrase) for each subset. This score indicates how well the system performs in cases that include one specific phenomenon. We compare the performance of the different phenomena and also compare them with the *"overall performance"*.

Prior to running the experiments we verified that: 1) the relative distribution of the phenomena in paraphrases and in non-paraphrases is very similar; and 2) there is no significant correlation (Pearson r <0.1) between the distributions of the individual phenomena. These findings show that the sub-tasks are non-trivial: 1) the binary labels of the pairs cannot be directly inferred by the presence or absence of phenomena; and 2) the different subsets of the test set are relatively independent and the performance on them cannot be trivially reduced to overlap and phenomena co-occurrence.

The *"overall performance"* and *"phenomena performance"* of a system compose its *"performance profile"*. With it we aim to address the rest of our research questions (RQs):

RQ 2 Which are the strong and weak sides of each individual system?

RQ 3 Are there any significant differences between the *"performance profiles"* of the systems?

RQ 4 Are there phenomena on which all systems perform well (or poorly)?

4 PI Systems

To demonstrate the advantages of our evaluation framework, we have replicated several popular Supervised and Unsupervised PI systems. We have selected the systems based on three criteria: popularity, architecture, and performance. The systems that we chose are popular and widely used not only in PI, but also in other tasks. The systems use a wide variety of different ML architectures and/or different features. Finally, the systems obtain comparable quantitative results on the PI task. They have also been reported to obtain good results on the MRPC corpus which is the same size as ETPC. The choice of system allows us to best demonstrate the limitations of the classical quantitative evaluation and the advantages of the proposed qualitative evaluation.

To ensure comparability, all systems have been trained and evaluated on the same computer and the same corpus. We have used the configurations recommended in the original papers where available. During the replication we did not do a full grid-search as we want to replicate and thereby contribute to generalizable research and systems. As such, the quantitative results that we obtain may differ from the performance reported in the original papers, especially for the Supervised systems. However, the results are sufficient for the objective of this paper: to demonstrate the advantages of the proposed evaluation framework.

We compare the performance of five Supervised and five Unsupervised systems on the PI task, including one Supervised and one Unsupervised baseline systems. We also include Google BERT (Devlin et al., 2018) for reference.

The **Supervised PI systems** include:

[S1] Machine translation evaluation metrics as hand-crafted features in a Random Forest classifier. Similar to Madnani et al. (2012) *(baseline)*

[S2] A replication of the convolutional network similarity model of He et al. (2015)

[S3] A replication of the lexical composition and decomposition system of Wang et al. (2016)

[2] i.e. The "diathesis alternation" subset contains all pairs that contain the "diathesis alternation" phenomenon (such as the example pair 3a–3b). Some of the pairs can also contain multiple phenomena: the example pair 3a–3b contains both *"same polarity substitution (habitual)"* and *"diathesis alternation"*. Therefore pair 3a–3b will be added both to the *"same polarity substitution (habitual)"* and to the *"diathesis alternation"* phenomena subsets. Consequentially, the sum of all subsets exceeds the size of the test set.

[S4] A replication of the pairwise word interaction modeling with deep neural network system by He and Lin (2016)

[S5] A character level neural network model by Lan and Xu (2018a)

The **Unsupervised PI systems** include:

[S6] A binary Bag-of-Word sentence representation (baseline)

[S7] Average over sentence of pre-trained Word2Vec word embeddings (Mikolov et al., 2013)

[S8] Average over sentence of pre-trained Glove word embeddings (Pennington et al., 2014)

[S9] InferSent sentence embeddings (Conneau et al., 2017)

[S10] Skip-Thought sentence embeddings (Kiros et al., 2015)

In the unsupervised setup we first represent each of the two sentences under the corresponding model. Then we obtain a feature vector by concatenating the absolute distance and the element-wise multiplication of the two representations. The feature vector is then fed into a logistic regression classifier to predict the textual relation. This setup has been used in multiple PI papers, more recently by Aldarmaki and Diab (2018). While the vector representations of BERT are unsupervised, they are fine-tuned on the dataset. Therefore we put them in a separate category (System #11).

ID	System Description	Acc	F1
	SUPERVISED SYSTEMS		
1	MTE features (baseline)	.74	.819
2	He et al. (2015)	.75	.826
3	Wang et al. (2016)	**.76**	**.833**
4	He and Lin (2016)	.76	.827
5	Lan and Xu (2018a)	.70	.800
	UNSUPERVISED SYSTEMS		
6	Bag-of-Words (baseline)	.68	.790
7	Word2Vec (average)	.70	.805
8	GLOVE (average)	.72	.808
9	InferSent	**.75**	**.826**
10	Skip-Thought	.73	.816
11	Google BERT	.84	.889

Table 1: Overall Performance of the Evaluated Systems

5 Results

5.1 Overall Performance

Table 1 shows the *"overall performance"* of the systems on the 1725 text pairs in the test set. Looking at the table, we can observe several regularities. First, the deep systems outperform the baselines. Second, the baselines that we choose are competitive and obtain high results. Since both baselines make their predictions based on lexical similarity and overlap, we can conclude that the dataset is biased towards those phenomena. Third, the supervised systems generally outperform the unsupervised ones, but without running a full grid-search the difference is relatively small. And finally, we can identify the best performing systems: **S3** (Wang et al., 2016) for the supervised and **S9** (Conneau et al., 2017) for the unsupervised. BERT largely outperforms all other systems.

The *"overall performance"* provides a good overview of the task and allows for a quantitative comparison of the different systems. However, it also has several limitations.

It does not provide much insight into the workings of the systems and does not facilitate error analysis. In order to study and improve the performance of a system, a developer has to look at every correct and incorrect predictions and search for custom defined patterns. The *"overall performance"* is also not very informative for a comparison between the systems. For example **S3** (Wang et al., 2016) and **S4** (He and Lin, 2016) obtain the same Accuracy score and only differ by 0.06 F1 score. With only looking at the quantitative evaluation it is unclear which of these systems would generalize better on a new dataset.

5.2 Full Performance Profile

Table 2 shows the full *"performance profile"* of **S3** (Wang et al., 2016), the supervised system that performed best in terms of *"overall performance"*. Table 2 shows a large variation of the performance of **S3** on the different phenomena. The accuracy ranges from .33 to 1.0. We also report the statistical significance of the difference between the correct and incorrect predictions for each phenomena and the correct and incorrect predictions for the full test set, using the Mann–Whitney U-test[3] (Mann and Whitney, 1947).

Ten of the phenomena show significant difference from the overall performance at p <0.1. Note

[3] The Mann–Whitney U-test is a non-parametric equivalence of T-test. The U-Test does not assume normal distribution of the data and is better suited for small samples.

OVERALL PERFORMANCE		
Overall Accuracy	.76	
Overall F1	.833	
PHENOMENA PERFORMANCE		
Phenomenon	**Acc**	**p**
Morphology-based changes		
Inflectional changes	.79	.21
Modal verb changes	**.90**	**.01**
Derivational changes	.72	.22
Lexicon-based changes		
Spelling changes	**.88**	**.01**
Same polarity sub. (habitual)	.78	.18
Same polarity sub. (contextual)	.75	.37
Same polarity sub. (named ent.)	.73	.14
Change of format	.75	.44
Lexico-syntactic based changes		
Opp. polarity sub. (habitual)	1.0	na
Opp. polarity sub. (context.)	.68	.14
Synthetic/analytic substitution	.77	.39
Converse substitution	**.92**	**.07**
Syntax-based changes		
Diathesis alternation	.83	.12
Negation switching	.33	na
Ellipsis	**.64**	**.07**
Coordination changes	.77	.47
Subordination and nesting	**.86**	**.01**
Discourse-based changes		
Punctuation changes	**.87**	**.01**
Direct/indirect style	.76	.5
Syntax/discourse structure	**.83**	**.05**
Other changes		
Addition/Deletion	**.70**	**.05**
Change of order	**.81**	**.04**
Contains negation	.78	.32
Semantic (General Inferences)	.80	.21
Extremes		
Identity	.77	.29
Non-Paraphrase	**.81**	**.04**
Entailment	.76	.5

Table 2: Performance profile of Wang et al. (2016)

that eight of them are also significant at p <0.05. The statistical significance of *"Opposite polarity substitution (habitual)"*, and *"Negation Switching"* cannot be verified due to the relatively low frequency of the phenomena in the test set.

The demonstrated variance in phenomena performance and its statistical significance address **RQ 1**: we show that the performance of a PI system on each candidate-paraphrase pair depends on the different phenomena involved in that pair or at least there is a strong observable relation between the performance and the phenomena.

The individual *"performance profile"* also addresses **RQ 2**. The profile is humanly interpretable, and we can clearly see how the system performs on various sub-tasks at different linguistic levels. The qualitative evaluation shows that **S3** performs better when it has to deal with: 1) surface phenomena such as *"spelling changes"*, *"punctuation changes"*, and *"change of order"*; 2) dictionary related phenomena such as *"opposite polarity substitution (habitual)"*, *"converse substitution"*, and *"modal verb changes"*. **S3** performs worse when facing phenomena such as *"negation switching"*, *"ellipsis"*, *"opposite polarity substitution (contextual)"*, and *"addition/deletion"*.

5.3 Comparing Performance Profiles

Table 3 shows the full performance profiles of all systems. The systems are identified by their IDs, as shown in Table 1. In addition to providing a better error analysis for every individual system, the *"performance profiles"* of the different systems can be used to compare them qualitatively. This comparison is much more informative than the *"overall performance"* comparison shown in Table 1. Using the *"performance profile"*, we can quickly compare the strong and weak sides of the different systems.

When looking at the *"overall performance"*, we already pointed out that **S3** (Wang et al., 2016) and **S4** (He and Lin, 2016) have almost identical quantitative results: 0.76 accuracy, 0.833 F1 for **S3** against 0.76 accuracy, 0.827 F1 for **S4**. However, when we compare their *"phenomena performance"* it is evident that, while these systems make approximately the same number of correct and incorrect predictions, the actual predictions and errors can vary.

Looking at the accuracy, we can see that **S3** performs better on phenomena such as *"Converse substitution"*, *"Diathesis alternation"*, and *"Non-Paraphrase"*, while **S4** performs better on *"Change of format"*, *"Opposite polarity substitution (contextual)"*, and *"Ellipsis"*.

We performed McNemar paired test comparing

PHENOMENON	**PARAPHRASE IDENTIFICATION SYSTEMS**										
	SUPERVISED					UNSUPERVISED					
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
OVERALL ACC.	.74	.75	.76	.76	.70	.68	.70	.72	.75	.73	.84
Inflectional	.77	.76	.79	.79	.75	.79	.75	.76	.78	.80	.84
Modal verb	.84	.89	.90	.89	.91	.92	.89	.84	.81	.89	.92
Derivational	.80	.83	.72	.73	.84	.80	.88	.86	.80	.77	.87
Spelling	.85	.83	.88	.90	.89	.85	.89	.88	.85	.89	.94
Same pol. sub. (hab.)	.74	.77	.78	.76	.76	.76	.76	.75	.76	.76	.85
Same pol. sub. (con.)	.74	.74	.75	.74	.70	.71	.71	.71	.73	.73	.81
Same pol. sub. (NE)	.74	.72	.73	.75	.64	.67	.65	.70	.73	.66	.80
Change of format	.80	.79	.75	.84	.85	.82	.81	.80	.80	.71	.91
Opp. pol. sub. (hab.)	1.0	1.0	1.0	.50	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Opp. pol. sub. (con.)	.77	.84	.68	.84	.52	.84	.61	.77	.65	.52	.71
Synthetic/analytic sub.	.73	.73	.77	.77	.74	.70	.72	.71	.73	.74	.83
Converse substitution	.93	.93	.92	.86	.93	.86	.79	.79	.93	.79	.86
Diathesis alternation	.77	.85	.83	.77	.83	.89	.85	.83	.84	.81	.85
Negation switching	1.0	.67	.33	.33	.33	.67	.33	.67	.33	.67	.33
Ellipsis	.77	.71	.64	.74	.80	.65	.81	.74	.61	.71	.81
Coordination	.92	.92	.77	.92	.77	.92	.85	.85	.92	.92	.92
Subord. & nesting	.83	.84	.86	.84	.81	.81	.85	.86	.80	.85	.93
Punctuation	.88	.90	.87	.87	.86	.87	.89	.89	.89	.88	.93
Direct/indirect style	.84	.84	.76	.80	.76	.80	.80	.84	.80	.80	.92
Syntax/disc. struct.	.80	.83	.83	.81	.78	.81	.80	.80	.76	.78	.82
Addition/Deletion	.69	.68	.70	.72	.67	.64	.65	.66	.70	.67	.82
Change of order	.82	.83	.81	.81	.77	.82	.82	.82	.83	.84	.89
Contains negation	.78	.74	.78	.79	.78	.72	.74	.78	.75	.76	.85
Semantic (Inferences)	.80	.89	.80	.81	.88	.90	.90	.92	.76	.79	.90
Identity	.74	.75	.77	.77	.73	.72	.73	.73	.76	.74	.85
Non-Paraphrase	.76	.77	.81	.75	.71	.55	.67	.68	.77	.79	.88
Entailment	.80	.80	.76	.76	.88	.80	.84	.88	.92	.88	.76

Table 3: Performance profiles of all systems

the errors of the two systems for each phenomena. Table 4 shows some of the more interesting results. Four of the phenomena with largest difference in accuracy show significant difference with p <0.1. These differences in performance are substantial, considering that the two systems have nearly identical quantitative performance.

Phenomenon	**#3**	**#4**	**p**
Format	.75	.84	.09
Opp. Pol. Sub (con.)	.68	.84	.06
Ellipsis	.64	.74	.08
Non-Paraphrase	.81	.75	.07

Table 4: Difference in phenomena performance between S3 (Wang et al., 2016) and S4 (He and Lin, 2016)

Phenomenon	**#3**	**#5**	**p**
Derivational	.72	.84	.03
Same Pol. Sub (con.)	.75	.70	.02
Same Pol. Sub (NE)	.73	.64	.01
Format	.75	.85	.03
Opp. Pol. Sub (con.)	.68	.52	.10
Ellipsis	.64	.80	.10
Addition/Deletion	.70	.67	.02
Identity	.77	.73	.01
Non-Paraphrase	.81	.71	.01
Entailment	.76	.88	.08

Table 5: Difference in phenomena performance: S3 (Wang et al., 2016) and S5 (Lan and Xu, 2018a)

We performed the same test on systems with

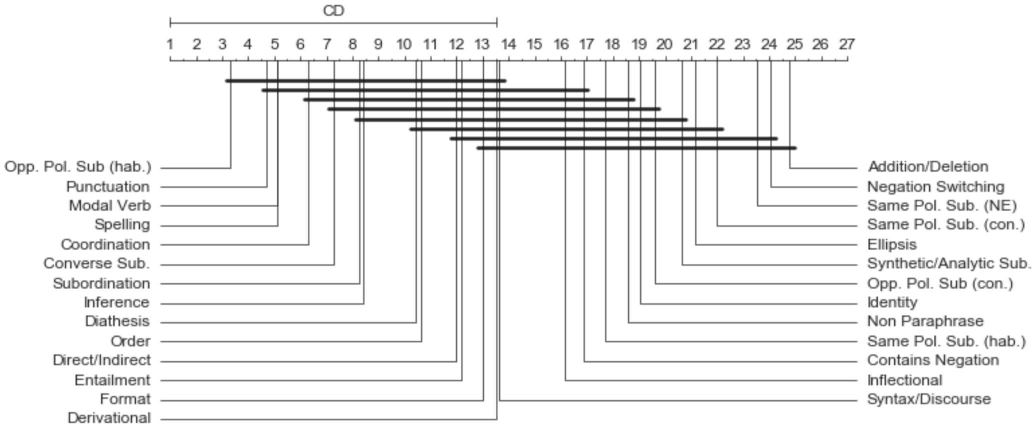


Figure 1: Critical Difference diagram of the average ranks by phenomena

a larger quantitative difference. Table 5 shows the comparison between **S3** and **S5** (Lan and Xu, 2018a). Ten of the phenomena show significant difference with p <0.1 and seven with p <0.05. These results answer our **RQ 3**: we show that there are significant differences between the *"performance profiles"* of the different systems.

5.4 Comparing Performance by Phenomena

The *"phenomena performance"* of the individual systems clearly differ among them, but they also show noticeable tendencies. Looking at the performance by phenomena, it is evident that certain phenomena consistently obtain lower than average accuracy across multiple systems while other phenomena consistently obtain higher than average accuracy.

In order to quantify these observations and to confirm that there is a statistical significance we performed Friedman-Nemenyi test (Demšar, 2006). For each system, we ranked the performance by phenomena from 1 to 27, accounting for ties. We calculated the significance of the difference in ranking between the phenomena using the Friedman test (Friedman, 1940) and obtained a Chi-Square value of 198, which rejects the null hypothesis with p <0.01. Once we had checked for the non-randomness of our results, we computed the Nemenyi test (Nemenyi, 1963) to find out which phenomena were significantly different. In our case, we compute the two-tailed Nemenyi test for k = 27 phenomena and N = 11 systems. The Critical Difference (CD) for these values is 12.5 at p <0.05.

Figure 1 shows the Nemenyi test with the CD value. Each phenomenon is plotted with its average rank across the 11 evaluated systems. The horizontal lines connect phenomena which rank is within CD of each other. Phenomena which are not connected by a horizontal line have significantly different ranking. We can observe that each phenomenon is significantly different from at least half of the other phenomena.

We can observe that some phenomena, such as *"opposite polarity substitution (habitual)"*, *"punctuation changes"*, *"spelling"*, *"modal verb changes"*, and *"coordination changes"* are statistically much easier according to our evaluation, as they are consistently among the best performing phenomena across all systems. Other phenomena, such as *"negation switching"*, *"addition/deletion"*, *"same polarity substitution (named entity)"*, *"opposite polarity substitution (contextual)"*, and *"ellipsis"* are statistically much harder, as they are consistently among the worst performing phenomena across all systems. With the exception of *"negation switching"* and *"opposite polarity substitution (habitual)"*, these phenomena occur in the corpus with sufficient frequency. These results answer our **RQ 4**: we show that there are phenomena which are easier or harder for the majority of the evaluated systems.

6 Discussion

In Section 3.2 we described our evaluation methodology and posed four research questions. The experiments that we performed and the analysis of the results answered all four of them. We briefly discuss the implications of the findings.

By addressing **RQ 1**, we showed that the performance of a system can differ significantly based on the phenomena involved in each candidate-paraphrase pair. By addressing **RQ 4**, we showed that some phenomena are consistently easier or harder across the majority of the systems. These findings empirically prove the complexity of paraphrasing and the task of PI. The results justify the distinction between the qualitatively different linguistic phenomena involved in paraphrasing and demonstrate that framing PI as a binary classification problem is an oversimplification.

By addressing **RQ 2**, we showed that each system has strong and weak sides, which can be identified and interpreted via its *"performance profile"*. This information can be very valuable when analyzing the errors made by the system or when reusing it on another task. Given the Deep architecture of the systems, such a detailed interpretation is hard to obtain via other means and metrics. By addressing **RQ 3**, we showed that two systems can differ significantly in their performance on candidate-paraphrase pairs involving particular phenomenon. These differences can be seen even in systems that have almost identical quantitative (Acc and F1) performance on the full test set. These findings justify the need for a qualitative evaluation framework for PI. The traditional binary evaluation metrics do not account for the difference in phenomena performance. They do not provide enough information for the analysis or for the comparison of different PI systems. Our proposed framework shows promising results.

Our findings demonstrate the limitations of the traditional PI task definition and datasets and the way PI systems are typically interpreted and evaluated. We show the advantages of a qualitative evaluation framework and emphasize the need to further research and improve the PI task. The *"performance profile"* also enables the direct empirical comparison of related phenomena such as *"same polarity substitution (habitual)"* and *"(contextual)"* or *"contains negation"* and *"negation switching"*. These comparisons, however, fall outside of the scope of this paper.

Our evaluation framework is not specific to the ETPC corpus or the typology behind it. The framework can be applied to other corpora and tasks, provided they have a similar format. While ETPC is the largest corpus annotated with paraphrase types to date, it has its limitations as some interesting paraphrase types (ex.: *"negation switching"*) do not appear with a sufficient frequency. We release the code for the creation and analysis of the *"performance profile"* [4].

7 Conclusions and Future Work

We present a new methodology for evaluation, interpretation, and comparison of different Paraphrase Identification systems. The methodology only requires at evaluation time a corpus annotated with detailed semantic relations. The training corpus does not need any additional annotation. The evaluation also does not require any additional effort from the systems' developers. Our methodology has clear advantages over using simple quantitative measures (Accuracy and F1 Score): 1) It allows for a better interpretation and error analysis on the individual systems; 2) It allows for a better qualitative comparison between the different systems; and 3) It identifies phenomena which are easy/hard to solve for multiple systems and may require further research.

We demonstrate the methodology by evaluating and comparing several of the state-of-the-art systems in PI. The results show that there is a statistically significant relationship between the phenomena involved in each candidate-paraphrase pair and the performance of the different systems. We show the strong and weak sides of each system using human-interpretable categories and we also identify phenomena which are statistically easier or harder across all systems.

As a future work, we intend to study phenomena that are hard for the majority of the systems and proposing ways to improve the performance on those phenomena. We also plan to apply the evaluation methodology to more tasks and systems that require a detailed semantic evaluation, and further test it with transfer learning experiments.

Acknowledgements

We would like to thank Darina Gold, Tobias Horsmann, Michael Wojatzki, and Torsten Zesch for their support and suggestions, and the anonymous reviewers for their feedback and comments.

This work has been funded by Spanish Ministery of Science, Innovation, and Universities Project PGC2018-096212-B-C33, by the CLiC research group (2017 SGR 341), and by the APIF grant of the first author.

[4] https://github.com/JavierBJ/paraphrase_eval

References

Hanan Aldarmaki and Mona Diab. 2018. Evaluation of unsupervised compositional representations. In *Proceedings of COLING 2018*.

Darina Benikova and Torsten Zesch. 2017. Same same, but different: Compositionality of paraphrase granularity levels. In *Proceedings of RANLP 2017*.

Rahul Bhagat and Eduard H. Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Elena Cabrio and Bernardo Magnini. 2014. Decomposing semantic inferences.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*, pages 350–356, Geneva, Switzerland. COLING.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. *Computational Linguistics UK (CLUK 2008) 11th Annual Research Colloqium.*

M. Friedman. 1940. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764. The Association for Computational Linguistics.

Darina Gold, Venelin Kovatchev, and Torsten Zesch. 2019. Annotating and analyzing the interactions between meaning relations. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 26–36, Florence, Italy. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. Association for Computational Linguistics.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2017. First quora dataset release: Question pairs.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 891–896.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Venelin Kovatchev, M. Antònia Martí, and Maria Salamó. 2018a. Etpc - a paraphrase identification corpus annotated with extended paraphrase typology and negation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).*

Venelin Kovatchev, M. Antònia Martí, and Maria Salamó. 2018b. WARP-text: a web-based tool for annotating relationships between pairs of texts. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 132–136, Santa Fe, New Mexico. Association for Computational Linguistics.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1224–1234.

Wuwei Lan and Wei Xu. 2018a. Character-based neural networks for sentence pair modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Wuwei Lan and Wei Xu. 2018b. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of COLING 2018.*

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 182–190, Stroudsburg, PA, USA. Association for Computational Linguistics.

H. B. Mann and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

P.B. Nemenyi. 1963. *Distribution-free Multiple Comparisons*. Ph.D. thesis, Princeton University.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Mark Sammons, V. G. Vinod Vydiswaran, and Dan Roth. 2010. "ask not what textual entailment can do for you...". In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1199–1208.

M. Vila, M. A. Martí, and H. Rodríguez. 2014. "is this a paraphrase? what kind? paraphrase boundaries and typology. ". pages 205–218.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *CoRR*, abs/1602.07019.

Study on Unsupervised Statistical Machine Translation for Backtranslation

Anush Kumar[1], Nihal V. Nayak[2], Aditya Chandra[1] and Mydhili K. Nair[1]
[1]Department of Information Science and Engineering, Ramaiah Institute of Technology
[2]Department of Computer Science, Brown University
anush070@gmail.com, nihalnayak@brown.edu,
adihyachndrt@gmail.com, mydhili.nair@msrit.edu

Abstract

Machine Translation systems have drastically improved over the years for several language pairs. Monolingual data is often used to generate synthetic sentences to augment the training data which has shown to improve the performance of machine translation models. In our paper, we make use of an Unsupervised Statistical Machine Translation (USMT) to generate synthetic sentences. Our study compares the performance improvements in Neural Machine Translation model when using synthetic sentences from supervised and unsupervised Machine Translation models. Our approach of using USMT for backtranslation shows promise in low resource conditions and achieves an improvement of 3.2 BLEU score over the Neural Machine Translation model.

1 Introduction

Neural Machine Translation systems with encoder-decoder architecture have significantly improved the state-of-the-art for several language pairs (Bahdanau et al., 2015; Vaswani et al., 2017). A majority of the systems in WMT18 used a Transformer approach with varying number of encoder-decoder layers (Bojar et al., 2018).

Supervised Neural Machine Translation systems are data-hungry as they require huge amounts of aligned parallel corpora (Koehn and Knowles, 2017). Additionally, obtaining parallel corpora is expensive and requires expert knowledge in both - source and target languages. To overcome this bottleneck, recent research has focused on unsupervised machine translation where only monolingual corpus is required, eliminating the need for a bilingual parallel corpora. The approaches in unsupervised machine translation have shown promise (Lample et al., 2018; Artetxe et al., 2018, 2019). Nonetheless, the performance of traditional Neural Machine Translation is still better than Unsupervised Machine Translation.

Often monolingual data is used to further improve the performance of a Neural Machine Translation model (Sennrich et al., 2016a; Currey et al., 2017). Backtranslation is one such popular method in which monolingual data is utilized to improve the model performance. In this technique, a model is initially trained in one of the directions (say target to source) and the trained model is used to translate a monolingual corpora to obtain synthetic sentences in the source language. These synthetic sentences are then included in the training set and a new model is trained from source to target. We take advantage of Unsupervised Machine Translation model for backtranslation (i.e. to generate synthetic sentences).

In our paper, we use Unsupervised Machine Translation Model (USMT) to generate the synthetic sentences for Russian-English language pair. Our aim is to improve the performance of the machine translation model using synthetic sentences from USMT model. Additionally, we look provide information on the settings and scenarios which benefit from USMT model and NMT model based backtranslation. To the best of our knowledge, there has been no study on using unsupervised machine translation models for backtranslation.

Our experiments indicated an improvement of 3.2 BLEU points for Russian to English language pair in a low resource setting while using synthetic sentences generated from an USMT model. However, we observed that NMT based backtranslation is superior when sufficient data is available and significantly improves the overall model performance.

Our paper is organized as follows - In the next section, we discuss related works in the field that have motivated us to carry out this study. Next, we

Proceedings of Recent Advances in Natural Language Processing, pages 578–582,
Varna, Bulgaria, Sep 2–4 2019.

Dataset	Type	Domain	No. of Sentences
News-Commentary-v14 (Ru-En)	Parallel	News	290866
Newscrawl 2018 (Ru)	Monolingual	News	8669559
Newscrawl 2017 (Ru)	Monolingual	News	8233907
Newscrawl 2018 (En)	Monolingual	News	18113311

Table 1: Statistics about the data

describe the datasets that we used for our experiments. Next, we describe our experimental setup for carrying out the experiments. We then analyze the results for NMT, USMT and other models on Russian - English dataset. We conclude the paper with discussion on future work.

2 Related Works

In this section, we will describe the main ideas and experiments using backtranslation and monolingual data.

Backtranslation was popularized by Sennrich et al. (2016a) where they improved the state-of-the-art in several language pairs. They trained a target to source machine translation model on the aligned parallel corpus. The model was later used to translate target-side monolingual sentences to the source language. These new synthetic sentences were added to the training set and a new model is trained.

On similar lines, Zhang and Zong (2016) use the source-side monolingual data to train the NMT model. They build a baseline NMT model and then use that model to generate the synthetic parallel data. They experiment on self-training as well as multitask learning.

He et al. (2016) introduced a dual learning mechanism for neural machine translation. They train translation models in both directions and use monolingual data to provide feedback on the quality of the translation. Their main contribution was to treat machine translation as a reinforcement learning problem.

Hoang et al. (2018) proposed a simple technique. They do show that iterative backtranslation improves the performance of the system on large datasets. However, they observe that the improvements in low resource datasets were not significant.

As mentioned earlier, obtaining aligned parallel corpora is expensive and cumbersome. Recent efforts in Unsupervised Machine Translation indicate that it is possible to develop a competitive system using only monolingual corpus (Lample et al., 2018; Artetxe et al., 2018).

In our study, we use an unsupervised statistical machine translation system based on Artetxe et al. (2018) [1]. In their paper, they describe a novel method to build a statistical machine translation model using monolingual data without any parallel data. The main idea is to learn word embeddings for each language independently and use linear transformations to bring them to shared space. These embeddings are used to generate the phrase table and then, the SMT is fine-tuned on a synthetic training set. In our work, we make use of this model to generate more synthetic data to be added to the parallel corpora.

3 Data

We perform our experiments on the Russian - English language pair. For training the supervised model (both the back-translated model and re-trained model), we use the Russian - English parallel corpus from WMT 19 [2]. To train our unsupervised model, we used monolingual corpora for Russian [3] and English [4] from Newscrawl 2017 and 2018 datasets. For English, we consider only the Newscrawl 2018 as the monolingual data where as for Russian we combine both Newscrawl 2017 and 2018 for the monolingual data.

We provide more details regarding the data in Table 1.

3.1 Preprocessing

For normalizing punctuation in the parallel corpora, we use the default scripts provided by Moses [5]. We then perform true-casing followed byte-

[1] We use this method as it did not require huge amounts of resource to train compared to other methods such as Artetxe et al. (2019)

[2] http://www.statmt.org/wmt19/index.html

[3] http://data.statmt.org/news-crawl/ru/

[4] http://data.statmt.org/news-crawl/en/

[5] https://github.com/moses-smt/mosesdecoder/tree/master/scripts

pair encoding while training the NMT [6] (Sennrich et al., 2016b).

3.2 Postprocessing

We remove byte pair encodings, detrucase and detokenize all the translated sentences before validating the predictions against the test set.

4 Experimental Setup

We describe our experimental setup in this section.

For our unsupervised statistical machine translation model, we use Monoses (Artetxe et al., 2018). We use the defaults for training the unsupervised model from the Monoses code. We train the model on both Russian to English (ru-en) and English to Russian (en-ru) monolingual corpus.

Monoses frameworks trains the model in 8 steps. Steps 1-7 involves training the word embeddings and bringing them to a shared space to build an initial phrase table. In step 8, 2M sentences are generated through backtranslation in both the directions. The phrase table is fine-tuned for 3 iterations using the synthetic sentences to obtain the final model.

We use OpenNMT for training the supervised machine translation models (Klein et al., 2017). We use the default architecture in OpenNMT, which includes an LSTM layer for encoding and another LSTM layer for decoding. Furthermore, we do not use the pretrained word embeddings. For both NMT and retraining the NMT model, we use the same architecture with default values.

To train the USMT model and NMT models, we used a system with 4x vCPUS, NVIDIA Tesla K80 GPU and 61 GB of RAM. The USMT model took about 2 weeks to complete training where as the NMT model usually took about 4-5 hours.

5 Experiments

Our primary aim is to show that we can use an unsupervised machine translation model to improve the performance of NMT systems. At the same time, we want to have a fair validation and investigate numerous scenarios where our approach performs well.

Therefore, in this experiment, we use monolingual data to generate synthetic sentences using both NMT and USMT separately and then, each of them is used to augment the training data for the Neural Machine Translation model. For example, we use back-translated Russian monolingual corpus to generate synthetic English sentences. These English sentences are added to the training corpus (with varying training corpora sizes) with English as the source and Russian as the target. We then train an NMT model from scratch and report our performance on an unseen test set.

In the case USMT, we backtranslate the monolingual corpus using the USMT model and augment the sampled training data. The training data is used to build the NMT model.

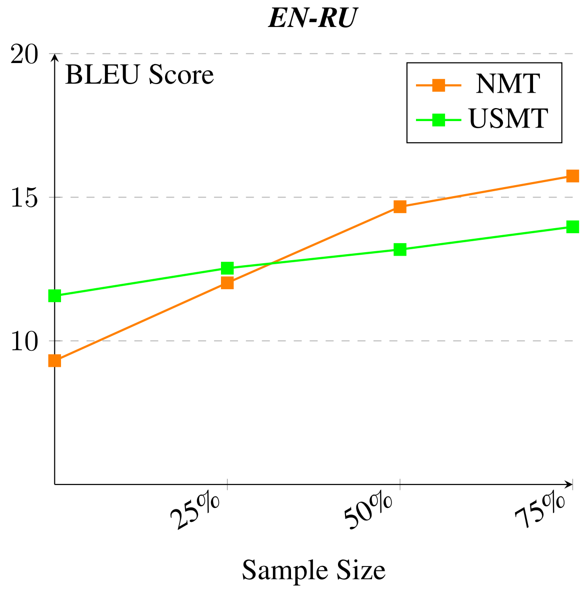


Figure 1: BLEU scores for NMT backtranslation and USMT backtranslation for EN-RU

6 Results

We perform all the experiments mentioned in the Table 2. Each row has a key which corresponds to the type of training corpora. The training corpora includes Only Parallel corpora, Only Monolingual Corpora and a sampled combination of the Parallel Corpora and the backtranslated Monolingual Corpora.

We obtain the baseline NMT model using the parallel corpus without any additional synthetic sentences. As mentioned earlier, we train the parallel corpora using the default encoder-decoder architecture from OpenNMT. Furthermore, we train a baseline USMT model using the monolingual corpora of English and Russian. We report the results in Table 2.

From the baselines of NMT and USMT, it is very clear that NMT outperforms USMT when

[6] https://github.com/rsennrich/subword-nmt/tree/master/subword_nmt

	ru-en		en-ru	
Training Corpus	**NMT**	**USMT**	**NMT**	**USMT**
Only Parallel Corpora	**17.65**	-	**15.54**	-
Only Monolingual Corpora	-	**15.58**	-	**8.07**
10% (∼29K) + Backtranslated Corpora	13.17	**16.34**	9.31	**11.57**
25% (∼72.5K) + Backtranslated Corpora	14.81	**17.17**	12.02	**12.63**
50% (∼145K) + Backtranslated Corpora	**19.63**	18.46	**14.67**	13.18
75% (∼217K)+ Backtranslated Corpora	**20.17**	19.34	**15.74**	13.97

Table 2: The BLEU scores for the different experiments. The training corpus with x% indicates the percentage of aligned training pairs randomly sampled from the parallel corpora.

there is sufficient data available.

To train the NMT backtranslation models, we sample the parallel corpora in batches of 10%, 25%, 50%, and 75% and train the NMT backtranslation model i.e. NMT model in the reverse direction (target to source). The synthetic data is generated by translating the monolingual corpora in English to Russian and Russian to English respectively. In Table 2, we refer to the synthetic sentences as Backtranslated Corpora. These sentences are combined with the sampled parallel corpora and retrained in the correct direction. In the case of USMT, we directly translate the monolingual corpora and include these synthetic sentences as a part of the training for NMT model in the correct direction.

Our results indicate the following - It can be seen that in critically low resource scenarios, the USMT backtranslated model performs better than the NMT backtranslated model (By 2.4 to 3.2 BLEU points for Russian to English and 0.61 to 2.26 BLEU points for English to Russian). However, the performance of the NMT system dramatically increases with the availability of parallel data. This shows that USMT as a backtranslation model works well mostly in low resource settings.

We can infer that the quality of the backtranslation model has a significant impact on the performance of the model. Additionally, we can see that the NMT model with small amount parallel data combined USMT model improves over the USMT baseline performance.

7 Discussion

In our future experiments, we would like to investigate the effect on lexical properties such as Named Entities and numbers in the predictions. We would also like to experiment our approach with newer techniques from Unsupervised Machine Translation (Artetxe et al., 2019). Additionally, we would like to extend our approach to other languages to comprehensively test our hypothesis.

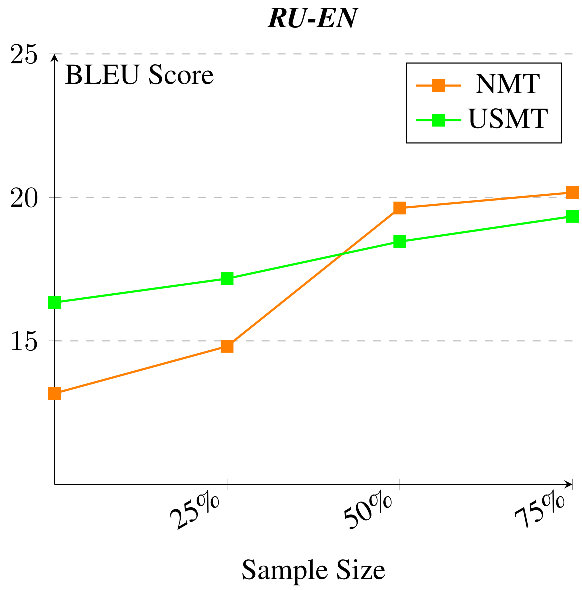


Figure 2: BLEU scores for NMT backtranslation and USMT backtranslation for RU-EN

Our experiments show that Unsupervised Statistical Machine Translation models can be used as a means of obtaining backtranslations to improve the performance of supervised machine translation models. We also note that the improved performances due unsupervised machine translation models are restricted to low resource scenarios. The performance of NMT model with NMT backtranslated sentences is superior when compared to the NMT model with USMT backtranslated sentences. In conclusion, our study helps in identifying the settings which benefit from USMT and NMT backtranslation models.

8 Code

To facilitate reconstruction of our paper, we are releasing the code - https://github.com/anush6/USMT_For_Backtranslation

9 Acknowledgement

We would like to thank our anonymous reviewers for their helpful feedback and comments.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2019. An effective approach to unsupervised machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Florence, Italy. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the EMNLP 2017 Second Conference on Machine Translation (WMT17)*, Copenhagen, Denmark.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 820–828. Curran Associates, Inc.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas. Association for Computational Linguistics.

Towards Functionally Similar Corpus Resources for Translation

Maria Kunilovskaya
University of Tyumen
Tyumen, Russia
m.a.kunilovskaya@utmn.ru

Serge Sharoff
University of Leeds
Leeds, UK
s.sharoff@leeds.ac.uk

Abstract

The paper describes a computational approach to produce functionally comparable monolingual corpus resources for translation studies and contrastive analysis. We exploit a text-external approach, based on a set of Functional Text Dimensions to model text functions, so that each text can be represented as a vector in a multidimensional space of text functions. These vectors can be used to find reasonably homogeneous subsets of functionally similar texts across different corpora. Our models for predicting text functions are based on recurrent neural networks and traditional feature-based machine learning approaches. In addition to using the categories of the British National Corpus as our test case, we investigated the functional comparability of the English parts from the two parallel corpora: CroCo (English-German) and RusLTC (English-Russian) and applied our models to define functionally similar clusters in them. Our results show that the Functional Text Dimensions provide a useful description for text categories, while allowing a more flexible representation for texts with hybrid functions.

1 Introduction

Comparable corpora are an important prerequisite for translation studies (TS) and contrastive analysis. One wants to make sure that the corpus resources used to explore differences between languages or aspects of translational specificity in several target languages (TL) are comparable in the first place.

One of the common approaches to corpus comparability is to define it as the domain similarity and to rely on the vocabulary overlap as the measure of comparability. A brief summary of possible interpretations of the concept and comparability measures can be found in Li et al. (2018). The authors give a domain-based definition to cross-linguistically comparable corpora: "*document sets in different languages that cover similar topics*". While lexical similarity is an important factor in linguistic variation, we would argue that it does not capture all the translationally relevant features of texts. Neumann (2013), Kruger and Van Rooy (2010) and Delaere (2015) have also highlighted the importance of register and genre in studying translations by showing that different registers produce different types of translationese. Moreover, functional theories within translation studies (TS) insist that what matters in translation is functional adequacy. The target text (TT) is expected to fulfill the same communicative functions as the source text (ST) and meet the TL conventions expected in the situation of TL communication of the message (Nord, 2006). Both Reiss and Vermeer (1984) and Neubert (1985) build their theory of translation around genres or text types, while Shveitzer (1973) underlines the impact of the text functions hierarchy on the translator's linguistic choices.

The above suggests that translational comparability of corpus resources should take into account social and situational constrains of the communication and the the speaker's purpose along with the text topic. The functional and communicative variation of texts is usually interpreted through the concepts of register and genre. For the purposes of this research, we will accept the distinction between the two suggested by Lee (2001). Register, as a text-**internal** view with respect to text categorization, refers to the lexicogrammatic choices made by the author. This notion reflects the differences in the linguistic make-up of texts and it relies on frequencies of lexicogrammatic features such as passive voice, relative clauses or personal pronouns. It is assumed that the observed linguistic variation captures the possible combinations of field, tenor and mode, the most prominent factors of communication (suggested by Halliday (1985)).

On the other hand, genres are understood as conventionally recognizable text categories that can be established on a number of **external** criteria, referring to the function of the text and its situational constrains. According to Lee (2001), most existing corpora rely on the text-external approach to text categorization and the choice of parameters behind it is guided by practical considerations in each case. It has been shown how little comparability there is between the genre classifications used to annotate different corpora (Sharoff, 2018). TS researchers interested in register variation find that existing corpora provide "limited background information on the genres ... and how they were defined" and choose to set up annotation tasks to reorganize the corpora (Delaere, 2015).

One translationally relevant common footing to compare texts from corpora with divergent or absent genre annotation is to rely on their function. On the one hand, text function is an important factor in translation, as texts aimed at informing the reader are translated differently from texts aimed at promoting goods and services (Lapshinova-Koltunski and Vela, 2015). On the other hand, text functions can be used to produce continuous rather than discrete text descriptions and account for hybrid texts. In this research we explore the potential of Functional Text Dimensions (FTD), hand-annotated for English and Russian (Sharoff, 2018) to produce text representations and to build functionally comparable corpora for TS research.

The aim of the present study is solve a practical task of creating research corpora for the study of translational tendencies in English-German and English-Russian translation. To this end, we develop a method to build a reasonably big and functionally homogeneous intersection of the three text collections: CroCo, an English-German parallel corpus (Hansen-Schirra et al., 2006), and the students and professional collections from RusLTC, a English-Russian parallel corpus (Kutuzov and Kunilovskaya, 2014). Our major motivation for this research is to find a way to reconcile the diverging genre annotations that exist in these corpora (see Table 4). We want to reduce the probability that the differences observed in the translations are down to the differences between the sources and are not genuine translational or cultural effects.

The rest of the paper is structured as follows. Section 2 has a brief overview of the topological approach to the text characterization and the research related to corpus similarity and genre classification. In Section 3 we describe our approaches to FTDs modelling and report the results of the intrinsic evaluation of the models. A selection of BNC genres is used to evaluate the models against an independent judgment and to test the clustering approaches to be used in the real-life task (Section 4). Section 5 presents a study that showcases the application of the functional vectors to computing the most similar parts of the two corpora. In Section 6 we aggregate the analytic results and highlight important findings.

2 Related Research

The practical needs to describe and compare corpora have made 'corpusometry' a prominent area of research in corpus linguistics. Below we outline the two major approaches to measuring similarity and describing the corpora contents. The first one is based on lexical features and yields a thematic description of corpus texts. It is one of the most prominent methods of measuring similarity between texts and/or building comparable corpora. For example, Kilgarriff and Salkie (1996) put forward a corpus homogeneity/similarity measure based on calculating χ^2 statistics from frequency lists or N keywords. A lexical approach to estimate the corpus composition is taken by Sharoff (2013). This research compared the results of clustering and topic modelling as ways to represent a corpus content using keywords statistics. In Li et al. (2018), the authors compared the performance of several bilingual vocabulary overlap measures on a specifically designed corpus with known comparability levels and found that frequencies of words with a simple Presence/Absence weighting scheme outperformed other approaches.

Another approach to measuring corpora has to do with calculating frequencies of a range of lexicogrammatic features (tense forms, modals) that allegedly reflect linguistically relevant parameters of the communicative situations. This text-internal approach to the text categorization can be best exemplified by Biber's work (Biber, 1988). He used several dozens of hand-picked text-internal features to place a text along each of the six dimensions (e.g. involved vs informational production or abstract vs non-abstract information). Biber's multidimensional approach to describing text variation has been criticized for lack of interpretability and, more importantly, for being loosely related to any external and situational factors, which cab be a socially more important reality for text categorization than linguistic features. The latter can throw together texts that are perceived as belonging to different genres (Lee, 2001). The attempts to classify genres, particularly, as annotated in the BNC and limited to a selection of major 'tried and tested' three or four top-level categories, have shown that the 67 Biber's features can be an overkill for a task like that. Lijffijt and Nevalainen (2017) report over 90% classification accuracy for the BNC four major genres on just pairs of surface features (such as frequencies of nouns and pronouns, values of type-to-token ratio and sentence length). The results from Kilgarriff and Salkie (1996); Xiao and McEnery (2005) indicate that the most frequent words can cope with the four major BNC categories as well. More specifically, Xiao and McEnery (2005) show that keyword analysis can be used to replicate Biber's results. In effect they analyze differences in the frequencies of mostly functional words that are key to genre identification. In a setting similar to Biber's, Diwersy et al. (2014) use 29 lexicogrammatic features and mildly-supervised machine learning methods to tease apart genres annotated in CroCo. The visualizations they provide indicate that they have managed to clearly separate only fiction and instruction of the eight genres in their experiment.

This demonstrates that describing genres needs a sophisticated approach that takes into account a multitude of criteria such as topic and situated linguistic properties. This research continues the investigation of the functional aspect of genre shaped in Sharoff's Functional Text Dimensions (Sharoff, 2018). Sharoff's work establishes a text-external framework to capture human perception of the texts functionality (as distance to a functional prototype) and to link it to any text-internal representations, with the aim of predicting the functional setup of unknown texts. This work is particularly relevant to our task for three reasons: (1) it provides a solid theoretically grounded approach for comparing texts coming from different or unknown sources and for producing comparative descriptions for the corpora at hand, (2) it is focused on functional and communicative parameters of texts that are particularly important in TS, (3) this framework, like Biber's, provides a flexible way to represent texts functionality along a few dimensions instead of squeezing texts into the atomic genre labels. In effect, FTD framework is a way to produce functional text vectors that position each individual text in a multidimensional functional space and help to account for variation within and across text categories.

3 Modelling: Setup and Results

The annotated data from the FTD project was used to learn models that predicted 10-dimensional vectors for the English texts in our research corpora. Further on, we used these vectors to compare texts and to get functionally similar subcorpora for a subsequent TS research (Section 5).

The annotated data for English consists of 1624 chunks of texts that count about 2 mln tokens from two different sources: 5gthe Pentaglossal corpus (Forsyth and Sharoff, 2014) and ukWac (Baroni et al., 2009). We used the annotations for the 10 most prominent FTD described in Sharoff (2018). Each dimension received a score on a 4-point Likert scale that reflects the proximity of a text to the suggested functional prototype. The inter-annotator agreement is reported at Krippendorff's α >0.76. We refer the reader to Sharoff (2018) for more details on the FTD framework.

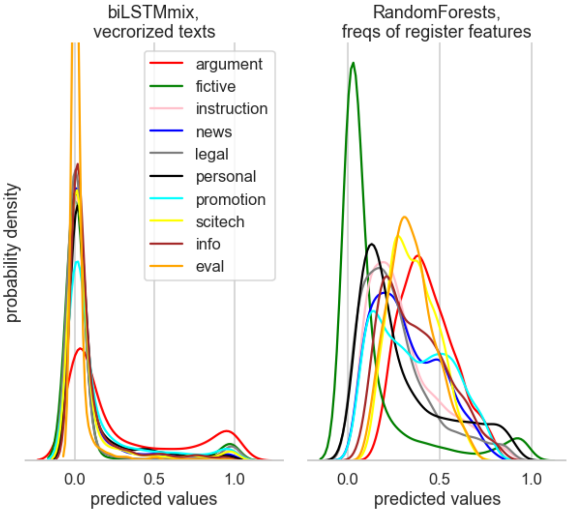
We used two modelling approaches to learn functional vectors from the annotated dataset: a multi-label task in a deep neural network architecture and a set of binary classifiers in a traditional machine learning setting. The respective models produced two types of functional vectors, which demonstrated comparable performance in several evaluation settings. This paper investigates the differences between, and adequacy of, these two types of functional vectors.

In the neural networks scenario, we used a bidirectional LSTM with an attention layer and two types of text input. Firstly, texts received mixed token-PoS representations suggested by Baroni and Bernardini (2006), biLSTMmix throughout this paper and in Table 1). The 1500 most frequent words were kept in their surface form, while the rest of the tokens were converted into their PoS. For example, a sentence “It was published in 1931 by one of New York’s major publishers.” was transformed into “It was VERB in [#] by one of PROPN PROPN major NOUN.” The embeddings for PoS were initialized as random vectors and trained in the Embedding layer. Secondly, we used lemmatised texts, with stop words filtered out (biLSTMlex in Table 1). For both scenarios we used pre-trained word embeddings of size 300, trained on the English Wikipedia and CommonCrawl data, using the skip-gram model, from the WebVectores database (Kutuzov et al., 2017). The preliminary experiments showed that cross entropy as the loss function with the Adam optimizer performed best (Kingma and Ba, 2014). We trained the models for 10 epochs. In the ML case, we reformulated the task as the binary classification task and learnt a classifier for each FTD. To this end, we binarized the existing human annotations by converting ‘0.5’ score to 0 and ‘2’ to 1. To get the real-valued functional vectors we used the probabilities returned for the positive class for each FTD on the assumption that the model would return higher probabilities for texts with a clearer functional makeup. We experimented with features (TF-IDF and Biber’s 67 text-internal register features) and with different algorithms (Support vector machines (SVM), RandomForest (RF), Logistic Regression (LogReg)). SVM and RF results below pertain to the experiments with the grid search optimized for macro F1 score. TF-IDF representation proved to be inferior to the Biber’s features and was excluded from the results below. We added a dummy classifier which randomly predicts a class with respect to the class distribution as a baseline. For register feature extraction (the Biber’s features) we used MAT for English (Nini, 2015).

To use a comparable performance metrics for the two learning approaches, the annotations and the models predictions were transformed into multi-hot vectors.

In Table 1 we report the standard measures averaged over 10 FTDs on the 10-fold cross validation for the six experiments. We accounted for the severe class imbalances in all training settings by using ‘class_weight=balanced’ option, stratified (multi-label) split with cross-validation and, at the evaluation stage, by choosing macro-averaging which penalizes model errors equally regardless of class distributions.

Figure 1. Distribution of predictions for the modeling approaches



From the statistics in Table 1, it follows that the deep learning approach is more accurate in determining the text functionality than the classical algorithms, and the mixed representations work best.

The difference between the models performance, however, is quite slim: it is in the second decimal digit only. A brief glance at the values of the functional vectors components (i.e. values predicted for each FTD) returned by the models reveals the differences in how the models arrive at the same overall result. Figure 1 shows the probability density for the values produced by the best performing models in each learning setup.

Figure 1 demonstrates that biLSTM, unlike the traditional ML algorithms, tends to predict near-zero values, with up to 7-11% of the training texts receiving values smaller than 0.2 on the strongest ‘dominant’ dimension.

In the next section we will show how the predictions of these two models correlate with the experiment-independent judgment we can suggest.

4 Evaluation on BNC Categories

4.1 Genre Classification

To test the functional vectors on the data outside the annotated dataset, we constructed a corpus with the ‘known’ genre composition. To this end, we followed Lee’s scheme for the BNC text

	FTD perspective			FTD minority class	Samples perspective	
	P	R	F1	F1	F1	neg_humming_loss
biLSTMmix	.804	.767	**.776**	.609	.640	.902
biLSTMlex	.787	.747	.757	.576	.596	.895
RF	.732	.756	.723	.532	.517	.846
SVM	.667	.531	.510	.095	.059	.844
LogReg	.664	.734	.659	.480	.527	.753
dummy	.504	.504	.504	.169	.134	.737

Table 1. Results of FTD modelling experiments

categories (Lee, 2001) to select the genres that are, in our opinion, most functionally distinct. The genres that use domain as the major underlying principle were deliberately excluded (religious texts, subcategories of academic texts). Table 2 has the basis parameters of the resulting BNC subcorpus. To find out how well the functional vectors reflect the structure of this functionally-motivated BNC subcorpus, we classified the six genres on the functional vectors produced by our best-performing models and compared their performance to several alternative representations: the raw statistics for Biber's features and log likelihood values for the 446 most common keywords. For brevity, in Table 3 we report the macro-averaged 10-fold cross-validated results only for RandomForest. Several other algorithms (SVM, LR) return approximately the same results.

genre	texts	words
academic(sci)	43	1.3M
editorial	12	115K
fiction(prose)	431	19M
instruct	15	492K
non-acad(sci)	62	2.8M
reportage	87	3.6M
Total	650	30M

Table 2. Basis statistics on the six functionally distinct categories from BNC

From Table 3 it follows that the models learnt on Biber's features coped with the selected BNC genres better than any other representations. However, the best performing pair of surface features from Lijffijt and Nevalainen (2017) — frequencies of nouns and pronouns, which return the reproducible result of over 90% accuracy on the four 'tried and tested' registers of English, — fail in the face of the more fine-grained categories.

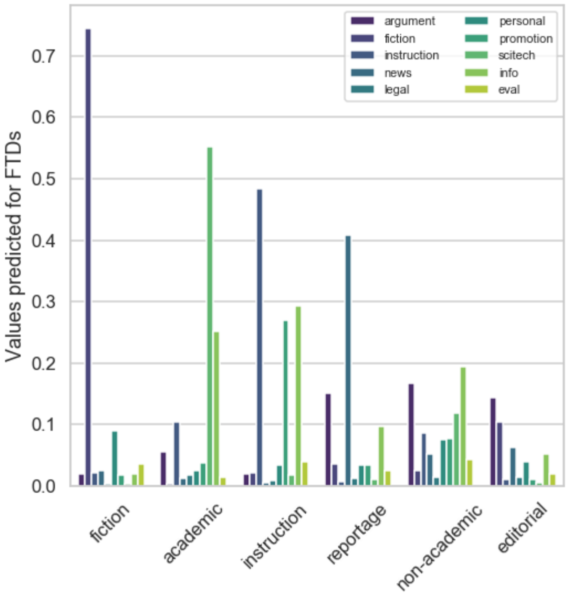
	P	R	F1
biLSTMmix	.84	.75	.79
biLSTMlex	.88	.66	.69
RF	.88	.90	**.89**
Baselines			
67 Biber's features	.93	.88	**.90**
Nouns+Pronouns	.76	.74	.74
keywords	.91	.79	.84

Table 3. BNC classification results

The analysis of the interrelations between the BNC genre labels and the predicted dominant functions suggest that only two categories can be easily mapped to the list of FTDs by all models: fiction and academic texts. The most problematic genres for all models are non-academic writings and editorials. For these texts the models either return no score above the 0.2 threshold on any of the dimensions or similar (relatively low) scores on several dimensions, especially on argumentative, evaluative, informational and personal. Editorials and non-academic texts stand out as functionally hybrid: in Figure 2, which shows the distribution of the FTD values predicted by biLSTM across the six genres, they do not have a functional focus, but integrate several text functions. Their hybrid status is evident from the more uniform distribution of average values for FTDs and from the diversity of text dominants predicted for these genres as well as from the higher percentage of the strong second function in the vectors.

The analysis of the predicted dominant functions against the actual genre labels shows that the best overall fit for BNC is produced by the RandomForest-based model, not the least because it does not produce vectors consisting of very low values only, which results in failure to define texts at all, given the accepted threshold of 0.2 necessary to signal a function. For all genre categories (except editorials) 60-95% of texts can be referred

Figure 2. Average values on the 10 FTDs by BNC genres predicted by biLSTMmix model



to the true genre category following the strongest prediction, if we map genres to FTDs as follows: fiction : fictive, academic : scitech, reportage : news, instruction : instruction, non-academic : argumentative. However, reducing a functional vector to just the strongest component would be unfair to the functionally hybrid texts that fall under the genre labels of non-academic and editorial in our BNC slice.

4.2 Testing the Clustering Method on BNC

The ultimate goal of this work is to produce a functional intersection of two corpora, i.e. to find functionally comparable texts in several sets. In this section we apply two clustering techniques to the BNC selection to determine which text representation and clustering approach is better at matching the annotated genres as class labels.

In the first clustering scenario we ran Affinity Propagation on a square matrix of pair-wise correlations pre-computed as euclidean similarities for the 650 BNC texts. This approach is attractive because it does not require the value of k, which is difficult to deduce in the real application context. We searched through the combinations of parameters to get the highest score for the Adjusted Rand Index (ARI), a clustering metric, which returns the proportion of text pairs that were assigned to the correct cluster, given the gold standard classes. The best clustering solution, with ARI=0.92 and 4 clusters on our BNC selection, was returned for both biLSTMmix and RandomForest vectors at damping=0.9 and preference=-12. The clusters, quite predictably, were built around (1) fiction, (2) reportage (3) non-academic + editorial (4) academic + instructions. Vectors learnt on lemmatized embeddings (biLSTMlex) were not able to converge to this solution.

An alternative clustering technique used in this research was KMeans algorithm with the Elbow Criterion method to determine k. The latter is based on measuring the sum of the squared distances between each member of the cluster and its centroid. k is defined as the number of clusters after which this value drops abruptly. However, this method needs to be applied with regard to the task at hand and some understanding of the data. For BNC k was automatically put at 2, because of the imbalance in our collection towards fiction: more than half of the texts were fiction. The best KMeans result (ARI=0.92) was registered on the RandomForest model vectors for k=5. It was superior to the best biLSTM result in that it separated the instruction cluster. Clustering on Biber's features and keywords did not achieve ARI of more than 0.2 for either Affinity Propagation or KMeans.

5 Case Study: CroCo and RusLTC

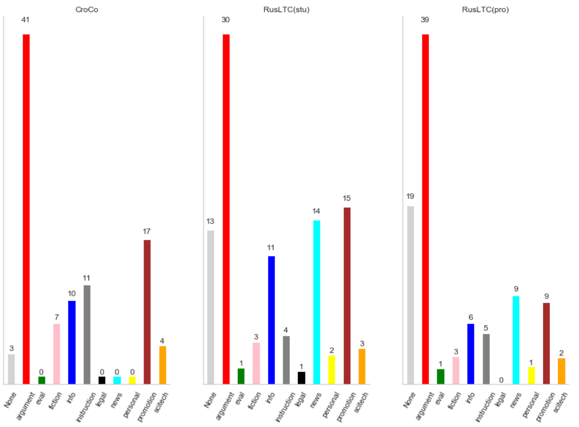
In this section we report the results of a case study where we used the functional vectors to get comparable functional clusters from several text collection. Our data comes from the English parts of the three parallel corpora: RusLTC, including student and professional translations subcorpora that have different English sources, and the CroCo corpus. As can be seen from Table 4, the three text collections vary in size, have diverging genre setup, and there is no way to tell whether the same categories include the same texts. In this work CroCo was chosen as the normative corpus, i.e. the starting point for the comparison and clustering operations.

The first step in solving our practical task with KMeans was to determine k. K-value was identified as n+1, where n is the number of the most populous groups formed by the texts with a specific FTD as the dominant function (see Figure 3, which show the ratio of texts with a specific dominant function). For the tree corpora in our experiment it seems possible to set k to 5 or 6. Our ex-

	CroCo EN>GE	**RusLTC(stu)** EN>RU	**RusLTC(pro)** EN>RU
Tokens	240 K	213 K	1.2 M
Texts	110	360	517
Genres		Acad(12)	Media(417)
	Essay(29)	Adverts(12)	Popsci (100)
	Fiction(10)	Educat(58)	
	Instr(10)	Essay(131)	
	Business(13)	Fiction(12)	
	Popsci(11)	Info(143)	
	Speech(14)	Interview(3)	
	Tourist(11)	Letters(3)	
	Web(12)	Speech(12)	
		Tech(15)	

Table 4. Parameters of the parallel corpora

Figure 3. Ratio of texts by the dominant FTD in the research corpora as predicted by biLSTM



periments showed that the difference was not critical for both types of functional vectors: it did not affect the makeup of the most populous cluster in CroCo. The clusters we received for the normative corpus (CroCo) were not at odds with the existing genre annotation (see Appendix). Both models succeed in grouping together instructions and fiction. The difference between the clusterings is in how the models interpret hybrid texts such as popular scientific texts and what aspects of texts they focus as secondary functions. biLSTM, which was learnt on the vectorized patterns of 1500 most frequent words and PoS for other tokens, produces vectors that highlight the fictional, narrative nature of pop-sci, throwing these texts together with fiction (Cluster3), while the classifiers learnt on frequencies of lexicogrammatic features (including inter alia lists of amplifiers and downtoners, private, public and suasive verbs (Quirk et al., 1985)) prioritize the informational and scientific component of pop-sci and group it with tourist informational leaflets (Cluster2).

Taking into account the size and homogeneity of the CroCo clusters, it makes sense to target Cluster 1 in finding functionally similar subsets from RusLTC(stu) and RusLTC(pro). These two collections were clustered with KMeans, the centriods for each cluster were calculated and compared to the CroCo centroids using Euclidean similarity measure. The most similar subsets of the two corpora are the clusters with most similar centroids. In determining k for RusLTC, we looked for a reasonable balance between the similarity and homogeneity scores. For RusLTC(stu) k = 8 and for RusLTC(pro) k = 10 return the best combination of the two.

To triangulate the results from KMeans, we compared them to the results for Affinity Propagation with the parameters tested on the BNC selection. The algorithm returned clusters which shared 85-98% of the files with the most successful KMeans result for all the experiments.

6 Discussion of Results

The primary goal of this project was to test the applicability of the text vectors learnt from the annotated text functions to the task of producing functionally similar subsets of two arbitrary corpora. This involved decisions on the input text representation, learning approach, clustering method and similarity metric. We have found that, first, the functional vectors learned on sequences, patterns and lexicogrammatic properties of texts were more effective in genre/function detection than those learnt on lexical features. Our results from neural networks modelling demonstrated that the functional properties of texts were better captured by the mixed sequences of the most frequent words and PoS than by lemmatized embeddings with stop words filtered out. The purely lexical features (TF-IDF) and keywords statistics proved inferior to lexicogrammar in the alternative ML setting, too.

However, the patterns of the most frequent words and PoS can be more successful with identifying some functions, but not other. In particular, it seems that the functional representations based on the vectorized texts did not quite capture the evaluative, personal and informational FTDs. This can be explained by two factors: first, these functions can rely on lexical features for their expression and, second, they are often annotated as a second strong dimension, unlike the mutually exclusive (genre-pivotal and relatively easy to predict) FTD such as fiction, instruction, news and scitech. To support this argument: in the classification task on the six hand-picked BNC genre categories, the raw Biber's features performed a bit better than the functional vectors learnt on them, while the biLSTMmix vectors demonstrated even less skill in recognizing our select BNC genres, where the majority of texts are of the easy-to-recognize type. On RusLTC(pro) corpus which consists of mass media texts and popular scientific texts, biLSTMmix returns no reliable predictions for the staggering 19% of texts. This analysis shows that FTD detection can benefit from combining vectorized and statistical register features, which we leave for future work.

Second, though our modelling approaches per se are not directly comparable, because they had different objectives and operated on different text representations, we can evaluate their usefulness for the practical task of predicting functional properties of texts. The inspection of the real-valued vectors indicates that the vectors learnt within the classification task setting overestimated the texts functionality (i.e. produced noisy predictions) and were less adequate in determining the functions hierarchy as manifested in the human scores. The two approaches had very similar overall performance in the intrinsic evaluation and in the BNC genre classification task, though in the real application they produce only partially overlapping clusters. This is probably because the models are focusing different properties of texts that are equally relevant for fulfilling text functions, but are more or less pronounced in individual real texts. It seems reasonable to use the union of the two sets for practical purposes. Besides, the models are different in terms of processing effort required, with the model on Biber's features less easily applicable to big corpora.

Third, the effectiveness of the functional representation was ultimately tested in the BNC clustering task. While for Affinity Propagation on pair-wise similarities the type of functional vectors did not matter, the better-performing KMeans proved to be sensitive to the difference in the functional vectors and managed to find a good fit to the BNC genres (5 clusters, ARI=0.92) only for the RandomForest vectors. The functional vectors learnt on embedded mixed representations achieved ARI=0.58 for any k in the range from 4 to 8. Note, however, that any functional vectors were by far better in this task than the baselines: we failed to produce any good clustering results for our BNC selection on the lexical and on the raw register features.

7 Conclusions

This paper presents an approach to deal with a practical issue of constructing functionally comparable corpus resources. We proposed a method to measure functional comparability of the resources at hand and to produce their functionally homogeneous intersection. The method offers a way to verify the researcher's judgments about the corpora comparability which are usually based on pre-existing corpus annotation schemes and researcher's intuition. We show that texts can be described externally via a reference to a number of communicative functions and that the functions are reflected via text-internal linguistic features. We found that functional text representations offer a better clustering result for a corpus with 'known' functions in comparison to keywords and linguistic register features. They can be effectively used to identify functionally homogeneous subsets of texts in a given text collection and to match them to functionally comparable sets from another corpus. The cross-linguistic extension of this research (left for future work) is supposed to equip a researcher with a corpus of non-translations in the TL functionally comparable to the ST. Such a reference corpus would effectively represent the expected TL textual fit (Chesterman, 2004) that is necessary to estimate specificity of translations.

Acknowledgments

The first author has been supported by the Russian Foundation for Basic Research within Project No. 17-06-00107.

References

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing* 21(3):259–274. https://doi.org/10.1093/llc/fqi039.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226.

Douglas Biber. 1988. *Variations Across Speech and Writing*. Cambridge University Press.

Andrew Chesterman. 2004. Hypotheses about translation universals. *Claims, Changes and Challenges in Translation Studies* pages 1–14. https://doi.org/10.1075/btl.50.02che.

Isabelle Delaere. 2015. *Do translations walk the line? Visually exploring translated and non-translated texts in search of norm conformity*. Phd, Ghent University.

Sascha Diwersy, Stefan Evert, and Stella Neumann. 2014. A weakly supervised multivariate approach to the study of language variation. In *Aggregating dialectology, typology, and register analysis. linguistic variation in text and speech*, Walter de Gruyter, Berlin, pages 174–204.

Richard Forsyth and Serge Sharoff. 2014. Document dissimilarity within and across languages: a benchmarking study. *Literary and Linguistic Computing* 29:6–22.

Michael A. K. Halliday. 1985. *An Introduction to Functional Grammar*. Edward Arnold, London.

Silvia Hansen-Schirra, Stella Neumann, and Mihaela Vela. 2006. Multi-dimensional annotation and alignment in an English-German translation corpus. In *Proc 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing at EACL*. Trento, pages 35–42.

Adam Kilgarriff and Raphael Salkie. 1996. Corpus similarity and homogeneity via word frequency. In *Proceedings of Euralex*. volume 96.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Haidee Kruger and Bertus Van Rooy. 2010. The features of non-literary translated language: a pilot study. *Proceedings of Using Corpora in Contrastive and Translation Studies, England, July 2010* .

Andrei Kutuzov, Murhaf Fares, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 58th Conference on Simulation and Modelling*. Linköping University Electronic Press, pages 271–276.

Andrey Kutuzov and Maria Kunilovskaya. 2014. Russian Learner Translator Corpus: Design, Research Potential and Applications. In *Text, Speech and Dialogue: 17th International Conference, TSD 2014, Brno, Czech Republic, September 8-12, 2014, Proceedings*. Springer, volume 8655, page 315.

Ekaterina Lapshinova-Koltunski and Mihaela Vela. 2015. Measuring 'Registerness' in Human and Machine Translation: A Text Classification Approach. *Proceedings of the Second Workshop on Discourse in Machine Translation* (September):122–131.

David Lee. 2001. Genres, registers, text types, domains, and styles: clarifying the concepts and navigating a path through the BNC jungle. *Language Learning and Technology* 5(3):37–72.

Bo Li, Eric Gaussier, and Dan Yang. 2018. Measuring bilingual corpus comparability. *Natural Language Engineering* 24(4):523–549. https://doi.org/10.1017/S1351324917000481.

Jefrey Lijffijt and Terttu Nevalainen. 2017. A simple model for recognizing core genres in the bnc. In *Big and Rich Data in English Corpus Linguistics: Methods and Explorations*, University of Helsinki, VARIENG eSeries, volume 19.

Albrecht Neubert. 1985. *Text and Translation*. Enzyklopdie.

Stella Neumann. 2013. *Contrastive register variation. A quantitative approach to the comparison of English and German*. Mouton de Gruyter, Berlin, Boston.

Andrea Nini. 2015. Multidimensional Analysis Tagger (v. 1.3).

Christiane Nord. 2006. Translating as a purposeful activity: a prospective approach. *TEFLIN Journal* 17(2):131–143.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Katharina Reiss and Hans J Vermeer. 1984. Groundwork for a general theory of translation. *Tubingen: Niemeyer* 101.

Serge Sharoff. 2013. Measuring the distance between comparable corpora between languages. In *Building and Using Comparable Corpora*, Springer, pages 113–130. http://www.tausdata.org/.

Serge Sharoff. 2018. Functional Text Dimensions for annotation of Web corpora. *Corpora* 13(1):65–95.

Alexandr Shveitzer. 1973. *Translation and Linguistics: Informational newspaper and military publicist texts in translation [Perevod i lingvistika: O gazetno-informacionom i voienno-publicisticheskom perevode]*. Voenizdat.

Zhonghua Xiao and Anthony McEnery. 2005. Two Approaches to Genre Analysis: Three Genres in Modern American English. *Journal of English Linguistics* 33:62–82. https://doi.org/10.1177/0075424204273957.

Appendix: Supplementary material

CroCo clusters as determined for the two alternative functional representations against the existing annotation

	biLSTMmix				RandomForest			
	texts	description	homo	genres	texts	description	homo	genres
Cluster 0	12	instr:0.91, promo:0.19, info:0.09	.685	INSTR 9, WEB 3	15	instr:0.71, promo:0.57, info:0.43	.687	INSTR 10, WEB 5
Cluster 1	**43**	argum:0.83, new:0.11, personal:0.08	**.706**	ESSAY 27, SPEECH 12, SHARE 2, POPSCI 1, WEB 1	**47**	argum:0.7, personal:0.62, new:0.54	.626	ESSAY 22, SPEECH 13, SHARE 7, POPSCI 3, FICTION 1, TOU 1
Cluster 2	12	info:0.59, promo:0.19, eval:0.14	.588	TOU 7, WEB 2, POPSCI 1, SPEECH 1, SHARE 1	39	scitech:0.5, new:0.5, info:0.48	.487	TOU 10, POPSCI 8, WEB 7, ESSAY 7, SHARE 6, SPEECH 1
Cluster 3	24	fiction:0.27, scitech:0.13, argum:0.1	.406	FICTION 10, POPSCI 8, WEB 3, ESSAY 2, SPEECH 1	9	fictio:0.85, eval:0.48, personal:0.35	.672	FICTION 9
Cluster 4	19	promo:0.7, info:0.15, argum:0.1	.527	SHARE 10, TOU 4, WEB 3, POPSCI 1, INSTR 1				

Question Similarity in Community Question Answering: A Systematic Exploration of Preprocessing Methods and Models

Florian Kunneman[A], **Thiago Castro Ferreira**[B], **Emiel Krahmer**[B], and **Antal van den Bosch**[A,C]

[A]Centre for Language Studies, Radboud University, The Netherlands
[B]Tilburg center for Cognition and Communication, Tilburg University, The Netherlands
[C]KNAW Meertens Institute, Amsterdam, The Netherlands

Abstract

Community Question Answering forums are popular among Internet users, and a basic problem they encounter is trying to find out if their question has already been posed before. To address this issue, NLP researchers have developed methods to automatically detect question-similarity, which was one of the shared tasks in SemEval. The best performing systems for this task made use of Syntactic Tree Kernels or the SoftCosine metric. However, it remains unclear why these methods seem to work, whether their performance can be improved by better preprocessing methods and what kinds of errors they (and other methods) make. In this paper, we therefore systematically combine and compare these two approaches with the more traditional BM25 and translation-based models. Moreover, we analyze the impact of preprocessing steps (lowercasing, suppression of punctuation and stop words removal) and word meaning similarity based on different distributions (word translation probability, Word2Vec, fastText and ELMo) on the performance of the task. We conduct an error analysis to gain insight into the differences in performance between the system set-ups. The implementation is made publicly available.[1]

1 Introduction

Community Question Answering (CQA) forums, such as Quora[2] and Yahoo Answers[3], are popular outlets to ask questions and receive answers, as well as to browse through questions and answers. Given the large amount of material on these platforms, a basic problem users encounter is trying to find out if (a variant of) their question has already been posed (and possibly answered) before. Given a target question provided by a user, the automatic task of querying and ranking semantically similar, relevant alternative questions in CQA forums is called *question similarity*.

For efficiency reasons, the question similarity task (also known as question relevance) normally works in two ranking steps. Given a target question, the first step consists of retrieving relevant questions using a general information retrieval technique, such as BM25 (Robertson et al., 2009), or a search engine such as Google (Page et al., 1999). The second step, the focus of this work and of most other studies in this field, consists of reranking the most likely candidate questions with a more fine-grained, domain-specific approach. Optionally, the system could also return whether a candidate question is a duplicate of the query.

The reranking task has been included as a benchmark task (Task 3 - Subtask B) in SemEval-2016/2017 (Nakov et al., 2016, 2017). Using the domain of *Qatar Living*[4], it consisted of re-ranking ten candidate questions retrieved by Google for a target question. Several promising approaches were proposed for this challenge, most notably *SimBOW* (Charlet and Damnati, 2017) based on the SoftCosine metric and winner of SemEval-2017, and KeLP (Filice et al., 2016), which is based on Tree Kernels and provided top results for all the subtasks in the challenge. However, little is known about the effects of particular design choices for these models, especially concerning the preprocessing methods and word-similarity metrics. Moreover, we know little about

[1] https://github.com/fkunneman/DiscoSumo/tree/master/ranlp

[2] https://www.quora.com/

[3] https://answers.yahoo.com/

[4] https://www.qatarliving.com

Proceedings of Recent Advances in Natural Language Processing, pages 593–601,
Varna, Bulgaria, Sep 2–4 2019.

how these models perform in comparison to (or combined with) more traditional question similarity techniques.

In this paper we therefore systematically combine and compare the SoftCosine metric and the Syntactic Tree Kernels with the more traditional BM25 and translation-based models. Moreover, we analyze the impact of preprocessing steps (lowercasing, suppression of punctuation and stop words removal) and word-similarity metrics based on different distributions (word translation probability, Word2Vec, fastText and ELMo).

The experiments were mainly conducted on the data of SemEval 2016-2017 - Task 3, based on the Qatar Living corpus. As a secondary goal, we also evaluated our main models in classifying question duplicates on the Quora dataset, so as to assess whether the results that we find apply to different datasets.

Results show that the choice of a preprocessing method and a word-similarity metric have a considerable impact on the final results. We also show that the combination of all the analyzed approaches leads to results competitive with related work in question-similarity.

2 Models

We compare two traditional and two recent approaches in this study: BM25, Translation-Based Language Model (TRLM), SoftCosine and Smoothed Partial Tree Kernels (SPTK - Syntactic Tree Kernels).

BM25 is a fast information retrieval technique (Robertson et al., 2009) used as a search engine in the first step of the shared task by many studies. We used the implementation of BM25 provided by *gensim*[5] as a baseline.

Translation-Based Language Model (TRLM) is a question similarity ranking function, first introduced by Xue et al. (2008). The method combines a language model with a word translation system technique, and is known to obtain better results on the question similarity task than BM25 and only the language model (Jeon et al., 2005). Equation 1 summarizes the TRLM ranking score between questions Q_1 and Q_2:

$$\begin{aligned} TRLM(Q_1, Q_2) = \prod_{w \in Q_1} (1-\sigma)P_{tr}(w|Q_2) + \sigma P_{lm}(w|C) \\ P_{tr}(w|Q_2) = \alpha \sum_{t \in Q_2} Sim(w,t) P_{lm}(t|Q_2) + \\ (1-\alpha)P_{lm}(w|Q_2) \end{aligned} \quad (1)$$

$Sim(w,t)$ denotes a similarity score among words w and t. In the original study, this similarity metric is the word-translation probability $P(w|t)$ obtained by the IBM Translation Model 1 (Brown et al., 1993). Furthermore, C denotes a background corpus to compute unigram probabilities in order to avoid 0 scores.

SoftCosine is the ranking function used by *SimBOW* (Charlet and Damnati, 2017), the winning system of the question similarity re-ranking task of SemEval 2017 (Nakov et al., 2017). The method is similar to a cosine similarity between the tf-idf bag-of-words of the pair of questions, except that it also takes into account word-level similarities as a matrix M. Given X and Y as the respective tf-idf bag-of-words for questions Q_1 and Q_2, Equation 2 summarizes the SoftCosine metric.

$$\begin{aligned} SoftCos(X,Y) = \frac{X^tMY}{\sqrt{X^tMX}\sqrt{Y^tMY}} \\ X^tMX = \sum_{i=1}^{n}\sum_{j=1}^{n} X_i M_{ij} Y_j \\ M_{ij} = max(0, cosine(V_i, V_j))^2 \end{aligned} \quad (2)$$

As $Sim(w,t)$ in Equation 1, M_{ij} represents the similarity between the i-th word of question Q_1 and the j-th one in question Q_2. $cosine$ is the cosine similarity, and V_i and V_j are originally 300-dimension embedding representations of the words, trained on the unannotated part of the Qatar living corpus using Word2Vec (Mikolov et al., 2013) with a context window size of 10.

Smoothed Partial Tree Kernels (SPTK) are the basis of KeLP (Filice et al., 2016), a system introduced by Croce et al. (2011). SPTK applies the kernel trick by computing the similarity of question pairs based on the number of common substructures their parse trees share. The difference with Partial Tree Kernels (PTK) (Moschitti, 2006) is that SPTK also considers word relations.

Besides the different variations of the model, which are well explained in Moschitti (2006) and

[5] https://radimrehurek.com/gensim/summarization/bm25.html

Filice et al. (2016), we designed SPTK in the following form. Equation 3 portrays the notation of the similarity metric among two questions' constituency trees, i.e. T_{Q_1} and T_{Q_2}.

$$TK(T_{Q_1}, T_{Q_2}) = \sum_{n_1 \in N_{T_{Q_1}}} \sum_{n_2 \in N_{T_{Q_2}}} \Delta(n_1, n_2) \quad (3)$$

$N_{T_{Q_1}}$ and $N_{T_{Q_2}}$ are the respective sets of nodes of parse trees T_{Q_1} and T_{Q_2}. $\Delta(n_1, n_2)$ is computed in distinct forms according to three conditions. (1) If the production rules of T_{Q_1} on n_1 and T_{Q_2} on n_2 are different, then $\Delta(n_1, n_2) = 0$. (2) If n_1 and n_2 are similar preterminals, then $\Delta(n_1, n_2) = Sim(w_{n_1}, w_{n_2})$, where Sim is similar to M_{ij} in Equation 2, as well as w_{n_1} and w_{n_2} are the terminal words for n_1 and n_2, respectively. (3) If the production rules of T_{Q_1} on n_1 and T_{Q_2} on n_2 are the same and both are not preterminals, then

$$\Delta(n_1, n_2) = \prod_{j=1}^{child(n_1)} \Delta(child(n_1)_j, child(n_2)_j) \quad (4)$$

So given a pair of constituency tree questions $p = \langle T_{Q_1}, T_{Q_2} \rangle$ to have their relevance scored and a training set of pair trees C, features are extracted in the following way:

$$SPTK(T_{Q_1}, T_{Q_2}) = \{TK(T_{Q_1}, T_{c_1}) + TK(T_{Q_2}, T_{c_2})\} \quad (5)$$

$$\text{where } \langle T_{c1}, T_{c2} \rangle_i \in C$$

The extracted kernel is used in Support Vector Machines Φ, whose output decision function is the relevance score among T_{Q_1} and T_{Q_2}.

Ensemble is the method we propose to combine the relevance scores produced by the previous approaches into a single model. Given questions Q_1 and Q_2 , we trained a Logistic Regression $\phi(Q_1, Q_2)$ with the relevance scores of BM25, TRLM and SoftCosine as features:

$$Ensemble(Q_1, Q_2) = \phi(Q_1, Q_2) \quad (6)$$

After empirically testing different settings, we found that the integration of SPTK in the ensemble method was most effective when interpolating its relevance score separately with the outcome of formula 6. Equation 7 denotes the model:

$$\begin{aligned} EnsSPTK(Q_1, Q_2) = \gamma\phi(Q_1, Q_2) + \\ (1-\gamma)\Phi(SPTK(T_{Q_1}, T_{Q_2})) \end{aligned} \quad (7)$$

We will compare the performance of the ensemble implementation with and without SPTK. For distinction, in the following sections we will refer to the former ensemble method as *Ensemble* and the latter as *EnsSPTK*.

3 Experiments

3.1 Data

Qatar Living We ran our experiments on the data of SemEval 2016-2017 - Task 3 based on the Qatar Living corpus[6]. Its training split consists of 267 target questions, 2,669 related questions (around 10 for each target question), and 26,690 comments (around 10 per related question). The development split and test sets of 2016 and 2017 have 50, 70, and 88 target questions, respectively (with the same proportion of related questions and comments as the training set). Given a target question, each of its related questions, retrieved by Google, was manually annotated as "Perfect Match", "Relevant" or "Irrelevant". The shared-task also provided a large unannotated dataset of Qatar Living, with 189,941 questions and 1,894,456 comments. In the Qatar Living corpus, each question is formed by a subject and a body. For the models BM25, TRLM and SoftCosine, we treat a question combining the subject and body into a single text, whereas we only use the subject for SPTK.

Quora To mitigate duplicate question pages at scale, Quora motivated the development of automated ways of detecting these questions by releasing a dataset with 400,000 pairs of questions together with a label for each entry indicating whether they are semantically identical (i.e., duplicates) or not[7]. We used this dataset to evaluate our most relevant models in the task of detecting question duplicates.

3.2 Settings

For the Translation model (TRLM), C was computed based on the training questions of the dataset used in the evaluation (e.g., Qatar Living or

[6] http://alt.qcri.org/semeval2017/task3/index.php?id=data-and-tools

[7] http://qim.fs.quoracdn.net/quora_duplicate_questions.tsv

Quora). In the Qatar Living corpus, the unannotated part of the data is also used to compute C.

Across the experiments, hyperparameters of the models such as σ and α of TRLM and γ of Ensemble with SPTK were optimized in the development split of the data through Grid Search. Moreover, Support Vector Machines in SPTK and Logistic Regression in Ensemble were implemented based on the Scikit-Learn toolkit (Pedregosa et al., 2011) and had their hyperparameters tuned by cross-validation on the training set.

3.3 Evaluation

In the SemEval shared-task, the question-similarity task was treated as a binary classification task, where the models aim to predict whether a related question is "Perfect Match/Relevant" or "Irrelevant". We evaluate the models using the Mean Average Precision (MAP) as the main metric, and also report F-Score for the classification models. In the Quora dataset, we evaluated the performance of our models in predicting question duplicates also using the F-Score measure.

3.4 Experiment 1: Preprocessing

From the top models for question similarity, little is known about the design process of their preprocessing methods. Filice et al. (2016) do not report on the preprocessing that they applied, and Charlet and Damnati (2017) lowercased the text as well as removed stopwords and punctuation. So in our first experiment, we evaluated BM25, TRLM, SoftCosine, *Ensemble* and *EnsSPTK* with 3 preprocessing methods (and all combinations of them): lowercasing, removal of stopwords[8] and suppression of punctuation. For SPTK we only apply lowercasing, since its constituency trees contain punctuation and stopwords as terminals. The preprocessing methods were applied in the training, development, test and unannotated parts of the data, such that probabilities and word distributions (e.g., word translation probability, Word2Vec, etc.) were affected.

3.5 Experiment 2: Word-Similarity

A central component of all of the evaluated models except BM25 is the use of a word-similarity metric. To evaluate which distribution better captures the similarity between two words for the task, we evaluated all the models using the word-translation probabilities, plus the cosine similarity measure depicted in Equation 2. In the latter, besides Word2Vec representations, we also tested fastText (Bojanowski et al., 2017), a distribution which takes character-level information and tends to overcome spelling variations, and the top layer of ELMo (Peters et al., 2018). To equalize the trials, the data used by the models were lowercased and stripped of stop words and punctuation.

4 Results

The first section of Table 1 lists the MAP of the preprocessing methods in the development part of the corpus for each model. Although the best combination of preprocessing methods differs between models, we see that preprocessing the data is beneficial for the performance of all models, except for SPTK. Between the best results, we see that suppression of punctuation is beneficial for all the models, while the removal of stopwords and lowercasing are detrimental to BM25 and TRLM, respectively.

The lower part of Table 1 lists the performance of each model according to the different word-level similarity metrics. The use of Word translation probabilities appears the under-performing method out of the five, showing the power of the continuous word representations. Surprisingly, we do not observe an improvement of fastText over Word2Vec representations. Even though CQA forums may have very noisy texts, the character-level information that fastText takes into account apparently does not help. Using the top layer of ELMo concatenated with Word2Vec representations leads to the best results in encoding the relation between words, except with TRLM.

Final Results Table 2 lists the results of the models with their best settings in the test sets of SemEval 2016-2017: BM25 with lowercased data without punctuation; TRLM with Word2Vec without stop words and punctuation; SoftCosine with Word2Vec+ELMo, lowercased data without stop words and punctuation; and SPTK with Word2Vec+ELMo and lowercased data. The table also shows the results of the best baseline (e.g., Google) and the winners of the SemEval 2016-2017 challenges. As expected, our best models were the ensemble approaches (e.g., *Ensemble* and *EnsSPTK*), which combine the ranking scores of all the other evaluated approaches and outperform

[8] We used the list of English stopwords provided by the NLTK framework (Bird and Loper, 2004)

Preproc.	**BM25**	**TRLM**	**SoftCosine**	**SPTK**	**Ensemble**	**EnsSPTK**
L.S.P.	68.80	68.43	**72.75**	-	**71.62**	**72.40**
L.S.	67.31	63.25	69.15	-	69.50	71.29
L.P.	**69.95**	68.42	65.33	-	68.70	69.16
S.P.	66.03	**68.65**	68.56	-	68.67	70.37
L.	67.07	66.42	63.68	54.34	67.04	67.41
S.	63.77	64.53	67.01	-	67.85	68.36
P.	65.05	64.38	60.04	-	65.31	66.66
-	63.52	64.95	60.66	**54.44**	63.08	64.31
Metric	**BM25**	**TRLM**	**SoftCosine**	**SPTK**	**Ensemble**	**EnsSPTK**
Translation	-	68.43	70.75	48.10	70.80	70.80
Word2Vec	-	**72.90**	72.75	54.44	71.40	72.64
fastText	-	70.93	71.07	53.49	71.92	71.92
Word2Vec+ELMo	-	71.41	**73.89**	**54.78**	**73.90**	**74.63**
fastText+ELMo	-	70.56	73.43	54.77	73.73	73.73

Table 1: MAP results on the different preprocessing and word-relation metric conditions in the development set. In the first part, *L.*, *S.* and *P.* denote lowercase, stop words removal and punctuation suppression methods respectively.

	2016		**2017**	
Models	**MAP**	**F-1**	**MAP**	**F-1**
Baseline	74.75	-	41.85	-
BM25	73.33	-	44.98	-
TRLM	71.94	-	44.25	-
SoftCosine	74.10	-	45.23	-
SPTK	45.61	21.24^{C}	29.63	33.13^{C}
Ensemble	75.48	66.96^{B}	46.74	**48.74**^{A}
EnsSPTK	75.40	**68.34**^{A}	47.06	**48.72**^{A}
Winner	**76.70**	66.39^{B}	**47.22**	42.37^{B}

Table 2: Final results of the models with their best preprocessing and word-relation settings in the test sets of SemEval 2016-2017. F-Score results were statistically significant with $p < 0.05$ according to the McNemar's test, with A outperforming B and C, and B outperforming C.

	No preproc. (-)	Preproc. (L.S.P.)
Word2Vec	0.50	0.50
Word2Vec+ELMo	0.50	**0.52***

Table 3: F1 scores of our ensemble method with different preprocessing techniques and word similarity measures on the Quora dev set.

the competitive baselines of SemEval 2016 and 2017.

Regarding the comparison between *Ensemble* and *EnsSPTK* in the test set of Semeval 2016, the approach without SPTK (*Ensemble*) is slightly better on re-ranking similar questions, but is significantly worse on classifying duplicates than *EnsSPTK*. The results are different in the test set of Semeval 2017: the latter approach is slightly better than the former on re-ranking similar questions according to the MAP metric, but shows a non-significant difference in classifying duplicates according to the F-1 score metric.

Although the results between our two best approaches are inconclusive, we argue that the inclusion of SPTK in the ensemble is not beneficial due to the trade-off between efficiency and performance. The SPTK approach, mainly its kernel, is computationally expensive and does not considerably improve the performance of the ensemble. For efficiency reasons we elect *Ensemble* as our best approach.

Checking the coefficients of the trained logistic regression model of *Ensemble*, we saw that the BM25 score (with a coefficient of 4.13) is the most relevant feature of the model, shortly followed by the SoftCosine score (with a coefficient of 3.48) and finally by the TRLM one (with a coefficient of 1.1).

In SemEval-2016, the UH-PRHLT model was the winner of the shared-task (Franco-Salvador et al., 2016). This system is based on a range of lexical (cosine similarity, word, noun and n-gram overlap) and semantic (word representations, alignments, knowledge graphs and common frames) features. In turn, our best model, *Ensemble*, with considerably less features, obtains competitive results in terms of MAP. The same pattern is seen for the SemEval-2017 test set: the Ensemble approach obtained competitive results with the winner *SimBOW*, also based on the SoftCosine metric, in terms of MAP, and outperforms it in F-Score.

Quora results Based on the previous results, we also evaluated the performance of our best question-similarity model, *Ensemble*[9], in classifying question *duplicates* on the Quora dataset. Table 3 depicts the results of our ensemble method with and without preprocessing and using two similarity metrics (Word2Vec and Word2Vec+ELMo). The best F-Score was obtained by the version which preprocesses the questions and represents the words with Word2Vec+ELMo. Results were statistically significant with $p < 0.05$ according to the McNemar's test.

5 Error Analysis

To obtain insight into the improvement by preprocessing setting, in Figure 1 we present the percentage of similar questions that were ranked better (placed on a higher position formerly occupied by a non-similar), equally or worse (switched a lower position with a non-similar) for each model-preprocessing combination in the Qatar Living corpus. The graph shows that each preprocessing manipulation results in both improved rankings and worsened rankings. The model that is least affected by the preprocessing steps is BM25, which shows to be a stable baseline. Most gain is seen for the SoftCosine model with all preprocessing steps, where 38% of the duplicates are ranked better and only 9% is ranked lower than a non-duplicate. Regarding the preprocessing steps applied in isolation, lowercasing leads to most changes for TRLM and BM25, while SoftCosine is most affected after removing stopwords.

The changes in performance by similarity metric are also presented in Figure 1. The highest gains are seen for the TRLM model, which yields an improved ranking for over 20% of the duplicates and a poor re-ranking for 12% to 14% when a similarity metric other than alignment is used. The SPTK model is not helped by a different similarity metric, with the most detrimental effect when combining the model with the translation alignment or fastText. The SoftCosine model with the default Word2Vec is also rather robust, with only a slight improvement when applying one of the ELMo metrics. These metrics do affect the rankings considerably, but lead to fairly equal improvements and declines of the ranking quality.

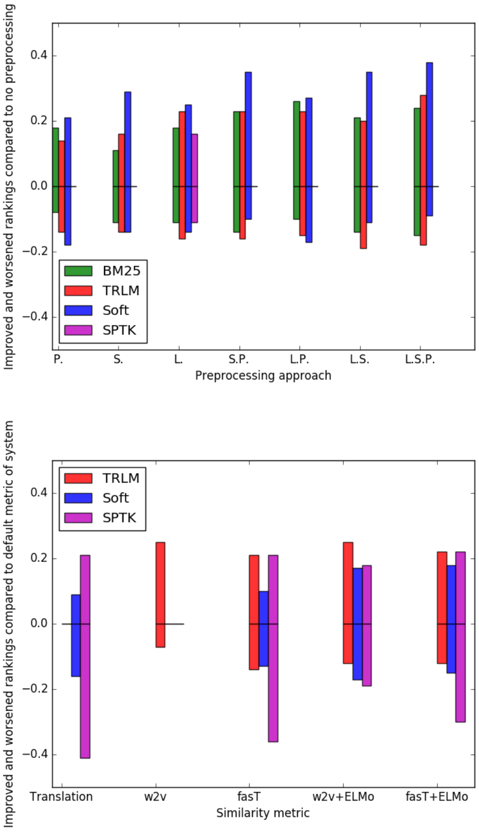


Figure 1: Percentage of similar questions that were ranked better, equally or worse after any of the preprocessing manipulations or similarity metrics combined with each system, in comparison to the standard setting without preprocessing (first graph) or the standard similarity metric for each system (second graph).

The performance patterns presented in Figure 1 show that the SoftCosine metric is affected most by the presence of stopwords. Explicit evidence is presented in Figure 2, which depicts the scores of the SoftCosine settings with and without preprocessing in relation to the number of stopwords in a question-pair. The setting without preprocessing shows a correlation with the number of stopwords: the similarity score goes up as the number of stopwords increases. The setting with preprocessing is, as expected, robust to the number of stopwords. This shows that the SoftCosine metric is considerably affected by the inclusion of stopwords, which hampers performance for the task of question similarity.

In Table 4 we present examples of question pairs in the Qatar Living development set along with their Gold standard label and the preprocessing steps or model that yielded a proper ranking

[9] Given the size of the Quora dataset, computing the kernel trick of SPTK would be intractable.

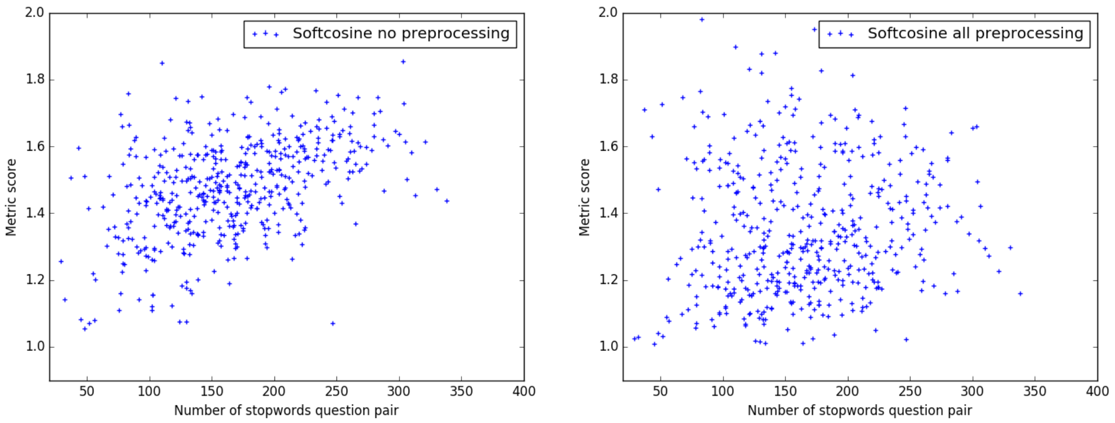


Figure 2: Similarity score in relation to the number of stopwords in a question pair, for the SoftCosine settings with and without preprocessing.

Question ID	Target question (subject - body)	Related question (subject - body)	Gold standard	Best performance
Q314	QLing after working hours - how many of you logon to QL after your working hours?	Surfing QL during office hours - How many hour(s) everyone spend surfing QL during office hours?	Similar	Stopword removal; SoftCosine
Q278	Beach cleaning - I am planning to organize a community service specifically beach cleaning to be carried out by our company staff. Any good suggestion of a beach?	NOT ONE PUBLIC BEACH IN DOHA? - Surrounded by pleasant waters and not ONE public beach! Ridiculous.	Not similar	No lowercasing
Q293	Water theme park in qatar - Hai Friends........ Any one knows the location of watar theme park in qatar; Is it beatiful? childrens have enough ride?? and howmuch fee Thanks	any water theme park in qatar? - Do you know about any water theme park in qatar?	Similar	SPTK

Table 4: Examples of question pairs with particular performance patterns.

for this pair. The first example, with question ID Q314, is of a similar question pair that was most often ranked in a high position by settings that included stopword removal and the SoftCosine model. Stopword removal shows particularly effective for the target question by removing 7 of the 15 words, and SoftCosine best matches 'office hours' to 'working hours'. The second question pair is exemplary of cases where preprocessing is actually detrimental. The related question, not similar to the target question, is partly written in capitals. After lowercasing, the word 'beach' is matched with the target question, which might result in a higher similarity score than questions that are actually similar. The final example, with ID Q293, is particularly well ranked by the SPTK model. On its own, SPTK did not compete with the other models in our study, but the focus on syntactic tree kernels could add a valuable angle to the similarity assessment. In this example, the good assessment by SPTK is likely due to the central phrase 'watar theme park in qatar', which is recurring, albeit with a different spelling, in the related question.

6 Discussion and Conclusion

Until now, careful preprocessing and smart combining of methods have remained understudied in the field of community question answering. Our results highlight that both pay off, yielding state-of-the-art results. Our findings show that lowercasing the input and removing both punctuation and stopwords yields the most robust outcomes, especially for the SoftCosine metric. In addition, representing the meaning of words by means of Word2Vec combined with the top layer of ELMo is the most beneficial word similarity implementation. Combining several metrics implemented with these optimal settings into an ensemble system based on logistic regression yields the best performance in terms of F1-score, being competitive with the winners of the SemEval tasks, and using fewer components.

The error analysis showed that the BM25 model is most stable across different preprocessing metrics, while the SoftCosine model mostly profits from preprocessing. Given the semantic matching that is done as part of SoftCosine and is absent in BM25, we can infer that preprocessing is an important prerequisite for effectively ranking question pairs based on semantic links.

Most of our experimentation was conducted on the Semeval dataset, in which similarity between questions is labeled. We also showed that adjusting preprocessing and word similarity settings leaded to better results in the task of identifying question duplicates, in the Quora dataset. More research is needed to see whether the patterns that we find are dataset-independent.

In future work we aim to compare the optimal models from our current study in a real-world setting, by running A/B testing on a open-domain CQA platform. Through clicks and likes by the users of such a platform, we can obtain insights into the value of these models when applied in the wild with many different question topics.

Acknowledgements

This work is part of the research programme Discussion Thread Summarization for Mobile Devices, which is financed by the Netherlands Organisation for Scientific Research (NWO). We thank the reviewers for their valuable comments.

References

Steven Bird and Edward Loper. 2004. Nltk: The natural language toolkit. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACLdemo '04. https://doi.org/10.3115/1219044.1219075.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146. http://aclweb.org/anthology/Q17-1010.

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, Meredith J Goldsmith, Jan Hajic, Robert L Mercer, and Surya Mohanty. 1993. But dictionaries are data too. In *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, pages 202–205.

Delphine Charlet and Geraldine Damnati. 2017. Simbow at semeval-2017 task 3: Soft-cosine semantic similarity between questions for community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. pages 315–319.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1034–1046.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. pages 1116–1123.

Marc Franco-Salvador, Sudipta Kar, Thamar Solorio, and Paolo Rosso. 2016. Uh-prhlt at semeval-2016 task 3: Combining lexical and semantic-based features for community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, pages 814–821. https://doi.org/10.18653/v1/S16-1126.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, pages 84–90.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 746–751.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*. Springer, pages 318–329.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. Semeval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 27–48. http://www.aclweb.org/anthology/S17-2003.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational

Linguistics, San Diego, California, pages 525–545. http://www.aclweb.org/anthology/S16-1083.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4):333–389.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 475–482.

A Classification-Based Approach to Cognate Detection Combining Orthographic and Semantic Similarity Information

Sofie Labat and Els Lefever
LT3, Language and Translation Technology Team, Ghent University
Groot-Brittanniëlaan 45, 9000 Ghent, Belgium
sofie.labat@gmail.com, els.lefever@ugent.be

Abstract

This paper presents proof-of-concept experiments for combining orthographic and semantic information to distinguish cognates from non-cognates. To this end, a context-independent gold standard is developed by manually labelling English-Dutch pairs of cognates and false friends in bilingual term lists. These annotated cognate pairs are then used to train and evaluate a supervised binary classification system for the automatic detection of cognates. Two types of information sources are incorporated in the classifier: fifteen string similarity metrics capture form similarity between source and target words, while word embeddings model semantic similarity between the words. The experimental results show that even though the system already achieves good results by only incorporating orthographic information, the performance further improves by including semantic information in the form of embeddings.

1 Introduction

In general linguistics, the term *cognate* is defined as a “language or a linguistic form which is historically derived from the same source as another language/form” (Crystal, 2008, page 83). The assumption of common etymology is, however, often disregarded in the literature, because certain research areas such as psycho-linguistics or natural language processing (NLP) tend to shift their focus from diachronic to perceptual relatedness (Shlesinger and Malkiel, 2005; Mitkov et al., 2007; Schepens et al., 2013; Hansen-Schirra et al., 2017). We follow this second strand of research in that we define cognates as words with high formal and semantic cross-lingual similarity. Conversely, *false friends* are words which have similar forms, but which differ in their meaning.

The ability to distinguish cognates from non-cognates (and especially) false friends is an important skill for second language learners. Similarly, source language interference is a problem often experienced by translators that is partly caused by the influence of cognates and false friends. Research in natural language processing can address these bottlenecks by, for instance, developing computer tools that aid second language users.

Nevertheless, most studies have mainly focused on the detection of cognates (Bergsma and Kondrak, 2007; Hauer and Kondrak, 2011; Ciobanu and Dinu, 2014; Rama, 2016), while relatively little attention has been devoted to false friends (Frunza and Inkpen, 2007; Mitkov et al., 2007; Ljubešić and Fišer, 2013; Castro et al., 2018). Mitkov (2007) explains that the main goal of investigation is often the cross-lingual identification of equivalent lexical items, as such knowledge can be integrated in other applications. Automatic cognate detection has indeed proven very useful for NLP, e.g. to boost the performance of automatic alignment between related languages or to compile bilingual lexicons (Smith et al., 2017).

The aim of this research is twofold: (1) we introduce a context-independent gold standard which can be used to classify English-Dutch pairs of cognates and non-cognates (among which false friends); (2) we develop a supervised binary classifier able to identify cognates across the English-Dutch language pair on the basis of orthographic and semantic information. Since our focus lies on the detection of cognates for these proof-of-concept experiments, no distinction is made between false friends and non-equivalent words.

The remainder of this paper is organized as follows. In Section 2, we give a brief overview of

Proceedings of Recent Advances in Natural Language Processing, pages 602–610,
Varna, Bulgaria, Sep 2–4 2019.

the existing research and methodologies to cognate detection. Section 3 describes the data and annotation process used to create the context-independent gold standard for English-Dutch cognate pairs, while Section 4 gives an overview of the experimental setup and the two types of information sources, viz. orthographic and semantic similarity features, that were used. In section 5, we report on the results of our classifier (1) incorporating only orthographic features and (2) combining orthographic and semantic similarity features. Section 6 concludes this paper and gives directions for future research.

2 Related Research

Extensive lists of known cognates and false friends are hard to find and expensive to compose, since they require a considerable amount of time and effort from trained lexicographers (Schepens et al., 2012). Especially for low resource languages, this constitutes a serious issue. Therefore, most NLP research on cognates has mainly focused on the automatic detection of such cognate pairs. In the literature, there are three main methods to identify cognates: orthographic, phonetic and semantic approaches. The oldest approaches to tackle this task involve simple string similarity metrics as the longest common subsequence ratio (Melamed, 1999) or the normalized Levenshtein distance (Levenshtein, 1965). More recently, however, the attention has been drawn to machine learning techniques. For instance, Frunza et al. (2007) combine several orthographic similarity measures to train a machine classifier, while Gomes et al. (2011) design a new similarity metric that is able to learn spelling differences across languages.

Different types of approaches can also be combined to distinguish cognates, e.g. Kondrak et al. (2004) join orthographic and phonetic information to distinguish between similar drug names. In order to capture the phonetic similarity between words, Konrak (2000) further developed a software package, called ALINE, which portrays phonemes as vectors of phonetic features, thus creating a phonetic similarity measure. Nevertheless, Heeringa et al. (2010) find that simple phonetic transcriptions still seem to outperform phonetic similarity metrics that are based on phonetic features. Hence, Schepens et al. (2013) propose to calculate a substitution cost for each pair of phonetically transcribed words by taking the edit distance between them. For this research, we opted to only focus on the orthographic proximity, as sound metrics require an additional phonetic transcription, thus making them less-likely to be applied on large data sets. Moreover, Schepens et al. (2013) find that there is a high consistency between orthographic and phonetic similarity measures for Dutch-English cognate pairs.

Whereas orthographic and phonetic features have often been employed to model the similarity between candidate cognate pairs, semantic information has often been ignored. Mitkov (2007) beliefs that this is another result of the main focus of investigation, which is the identification of cognates rather than distinguishing cognates from false friends. Semantic evidence is, however, an important information source, as it can not only be used to represent the semantic (dis)similarity between word pairs, but it can also further increase the accuracy of cognate detection systems. In his own research, Mitkov (2007) distinguishes between two types of semantic approaches: taxonomic and distributional semantic similarity measures. Whereas the first group relies on the taxonomic structure of a resource such as WordNet (Miller, 1995), the second approach relies on large corpora. The latter methods are based on the Distributional Hypothesis (Harris, 1954), which states that words that appear in similar contexts tend to share similar meanings. The different approaches that leverage this principle are typically divided into two categories: count-based methods, such as Latent Semantic Analysis, and predictive methods, such as neural probabilistic language models, which have gained a lot of popularity in today's NLP community. On the one hand, count-based models count how often a given target word co-occurs with its neighbor words in a large text corpus, after which the resulting counts are mapped to a dense vector for each word. On the other hand, predictive models directly try to predict a word from its neighbors in terms of learned dense embedding vectors (Baroni et al., 2014). Word2vec (Mikolov et al., 2013) is a particularly computationally-efficient and popular example of predictive models for learning word embeddings from raw text. In this research, we will incorporate the more recent fastText word embeddings as implemented by Bojanowski et al. (2017).

3 Data Creation

To train and evaluate the cognate detection system, we created a novel context-independent gold standard by manually labelling English-Dutch pairs of cognates and false friends in bilingual term lists. In this section, we describe how the lists of candidate cognate pairs were compiled on the basis of the Dutch Parallel Corpus (Macken et al., 2011) and how a manual annotation was performed to create a gold standard for English-Dutch cognate pairs.

3.1 List of Candidate Cognate Pairs

To select a list of candidate cognate pairs, unsupervised statistical word alignment using GIZA++ (Och and Ney, 2003) was applied on the Dutch Parallel Corpus (DPC). This high-quality parallel corpus for Dutch, French and English consists of more than ten million words and is sentence-aligned. It contains five different text types and is balanced with respect to text type and translation direction. The automatic word alignment on the English-Dutch part of the DPC resulted in a list containing more than 500,000 translation equivalents. A first selection was performed by applying the Normalized Levenshtein Distance (NLD) (as implemented by Gries (2004)) on this list of translation equivalents and only considering equivalents with a distance smaller than or equal to 0.5. This resulted in a list with 28,503 Dutch-English candidate cognate pairs, which was manually labeled.

3.2 Creation of Gold Standard

To create the gold standard for cognate detection, an extensive set of guidelines was established (Labat et al., 2019). The guidelines propose a clearly defined method for the manual labeling of the following six categories:

1. **Cognate**: words which have a similar form and meaning in all contexts. Conform with our working definition for cognates, the source and target words do not need to be etymologically related.

2. **Partial cognate**: words which have a similar form, but only share the same meaning in some contexts.

3. **False friend**: words which have a similar form but a different meaning.

4. **Proper name**: proper nouns (e.g. persons, companies, cities, countries, etc.) and their derivations (e.g. *American*).

5. **Error**: word alignment errors and compound nouns of which one part is a cognate but the other part is missing in one of the languages (e.g. *peripherals - aansturingsperipherals*).

6. **No standard**: words that do not occur in the dictionary (e.g. *num_connectors*) and numbers (e.g. *adm12006e, VI*).

To decide on the correct label, we adopted a context-independent approach applying the following procedure: (1) for every candidate cognate pair, the dictionary Grote Van Dale[1] (henceforth: VD) was consulted; (2) the English word is looked up in the VD, e.g. *salon*, (3) the Dutch translation is inspected in the VD, e.g. *salon: "nice room"* and *salon: "(room for) gathering of people (e.g. from the literary world)"*.

Based on the previously obtained information, a decision is made: in case all meanings of the Dutch word correspond with the English word, we consider them "cognates", in case only part of the Dutch meanings correspond with the English word, we consider them "partial cognates", in case the words have different meanings, we consider them "false friends". An example of partial cognates is the pair *agent-agent*: the Dutch *agent* refers both to (1) a police man and to (2) a representative (e.g. business representative). As only the second meaning of the Dutch word is expressed by the English *agent*, these words are considered partial cognates.

Two important observations should be made. Firstly, we accorded more fine-grained labels in the gold standard that are described in great detail in the annotation guidelines (Labat et al., 2019). For cognates, a distinction was, for instance, made between cognates of which Part-of-Speech (PoS) and meaning are identical in both languages, cognates that differ in PoS (e.g. *organisatie-organizing*) and cognates that differ in agreement (e.g. *organisatie-organisations*). Secondly, it is important to note that a successful dictionary look-up never overruled the "proper name" annotation.

The resulting gold standard is context-independent. Hence, it can be used for both the development and the evaluation of machine

[1] https://www.vandale.be/

learning models that deal with cognate detection. Besides its applications in natural language processing, the gold standard can also form an important new resource for further research on cognates in linguistics, translation studies or psycho-linguistics.

4 Classification

This section describes the experimental setup and the two types of information sources, viz. orthographic similarity and semantic similarity, that were incorporated for the experiments.

4.1 Experimental Setup

In this paper, cognate detection was approached as a supervised classification task. To this end, we applied Support Vector Machines as implemented in sklearn (Pedregosa et al., 2011).

The data set used for the binary classification experiments consisted of the COGNATE pairs (labels "cognate" and "partial cognate") and NON-COGNATE pairs (labels "error" and "false friend"). The categories of "proper name" and "no standard" were removed from the data set as they are always identical translations and would boost the performance of the system in an artificial way. Table 1 gives an overview of the distribution of the two classes in the gold standard data set.

	Cognate	Non-cognate	Total pairs
GS	9,855	4,763	14,618

Table 1: Distribution of the "cognate" and "non-cognate" class labels in the gold standard (GS).

In order to train and test the system, we performed 5-fold cross-validation for which we fixed our 5 subsamples. Hyperparameter optimisation was performed by means of a 5-fold cross-validation grid search on the training folds, resulting in the following values: $kernel$ = RBF, C = 5, $class\ weight$ = None and $gamma$ = 5.

4.2 Orthographic Similarity Features

Fifteen different string similarity metrics were applied on the candidate cognates to measure the formal relatedness between source and target words. Eleven of these fifteen metrics were also used by Frunza et al. (2007). The following list briefly summarizes the orthographic features implemented:

- **Prefix** divides the length of the shared prefix by the length of the longest cognate in the pair.
- **Dice** (Brew and McKelvie, 1996) divides the number of common bigrams times two by the total number of bigrams in the cognate pair, as in $\frac{2 \times |bigrams(x)| \cap |bigrams(y)|}{|bigrams(x)| + |bigrams(y)|}$.
- **Dice (trigrams)** differs from Dice in that it uses trigrams instead of bigrams.
- **XDice** is a variant of Dice as it uses bigrams that are created out of trigrams by deleting the middle letter in them.
- **XXDice** incorporates the string positions of the bigrams into its metric. Therefore, the denominator is no longer multiplied by two, but by $\frac{2}{1 + (pos(x) - pos(y))^2}$.
- **LCSR** stands for the longest common subsequence ratio, which is two times the length of the longest subsequence over the summed length of both sequences.
- **NLS** or the Normalized Levenshtein Similarity equals one minus the minimum number of edits required to change one string sequence to another.
- **LCSR (bigrams), NLS (bigrams), LCSR (trigrams), and NLS (trigrams)** differ from their standard metrics in that they use, respectively, bigrams and trigrams to calculate their results.
- **Jaccard index** models the length of the intersection of both cognate strings over the length of the union of these strings.
- **Jaro-Winkler similarity** is the complement of the Jaro-Winkler distance. Word pairs that from their beginning correspond to a set prefix length will receive higher scores.
- **Spsim option 1 and Spsim option 2** are the only metrics which require supervised training, in order to learn grapheme mappings between language pairs (Gomes and Pereira Lopes, 2011). They are trained by performing 5-fold cross-validation on the positive instances (i.e. cognates) in the data set. Therefore, we created two different train

sets: option 1 includes cognate pairs which differ in agreement or PoS-tags, while option 2 only includes cognates and partial cognates.

4.3 Semantic Information

Besides features that model formal similarity between word pairs, we also included semantic information in our classifier. We opted for word embeddings, as these have shown to be very effective for various NLP tasks. In addition, word embeddings have not yet been used for the task of cognate identification. For the purpose of this research, we worked with fastText word embeddings that were pre-trained on the Wikipedia corpus with the skip-gram model proposed by Bojanowski et al. (2017). The model was trained with the default parameters and the length of the vector was set to 300. A disadvantage of using text-formatted pre-trained embeddings is that we could not generate embeddings for all words in the gold standard list. As a result, we only obtained word embeddings for 12,433 instances, while we have orthographic information for 14,618 instances. Table 2 gives an overview of the distribution of the two classes in the full and reduced gold standard data sets. The experimental results that we obtained for this subset are presented in Section 5.2.

	Cognate	Non-cognate	Total pairs
Ortho	9,855	4,763	14,618
Semantic	8,935	3,498	12,433

Table 2: Distribution of the “cognate” and “non-cognate” class labels in the full (*Ortho*) and reduced (*Semantic*) gold standard data sets.

We chose to work with fastText embeddings instead of regular Word2Vec embeddings because the former model uses n-grams to train its embeddings. In contrast to the Word2Vec models, fastText can create word embeddings for out-of-vocabulary words, which is especially important for low-frequent words. Although the current research only works with pre-trained word entries, in future research we plan to add out-of-vocabulary words by training word embeddings on domain-specific corpora more similar to the DPC corpus that was used to extract the list of candidate cognate pairs. This way, we hope to construct embeddings for all word pairs in the gold standard list.

Once the results for the Dutch and English monolingual embeddings were loaded, the Dutch embeddings were mapped to the English vector space by means of a pre-trained alignment matrix (Smith et al., 2017). Since the embeddings are then situated in the same vector space, one can easily compute the cosine similarity between the two words of a candidate cognate pair. Subsequently, this cosine similarity was used as a semantic feature for our machine learning system.

5 Experimental Results

This section describes the classification results for two sets of experiments, namely (1) a classifier incorporating fifteen orthographic similarity features and (2) a classifier combining the same set of orthographic similarity features with a semantic feature resulting from computing the cosine similarity between the word embeddings of the cognate pair.

5.1 Experiment 1: Orthographic Features

A first set of experiments was conducted to evaluate the performance of the orthographic similarity features for the task of cognate detection. Table 3 lists the averaged precision, recall and F1-score for all individual orthographic similarity features and their combination.

The results show a very good performance of the classifier combining all orthographic similarity information (average F-score of 84%). Especially precision improves considerably when combining the different orthographic similarity metrics. When looking into the results for the individual features, it is clear that some metrics perform very well in isolation, such as LCSR and NLS, which obtain F-scores of around 85% for the positive class (“Cognates”) with good balance of precision and recall.

To get further insight in the informativeness of the various orthographic features, we also trained a conditional inference tree and random forest on the cognate data set. Figure 1 visualizes the model learned by the conditional inference tree at depth 3. The tree indicates which orthographic metric is the most important for that node in the tree. As can be observed in Figure 1, the longest common subsequence ratio is overall the most influential metric, followed by SpSim (option 1) and the Jaro-Winkler similarity.

In addition to the conditional inference tree, a

	Cognates			**Non-cognates**			**Average score**		
Metric	Prec	Rec	F-score	Prec	Rec	F-score	Prec	Rec	F-score
Prefix	77.43	87.84	82.31	65.17	47.03	54.62	71.30	67.44	68.46
Dice	73.38	91.99	81.63	65.04	30.91	41.84	69.21	61.45	61.73
Dice (3gr)	73.28	91.88	81.53	64.63	30.67	41.59	68.95	61.28	61.56
Jaccard	73.83	91.53	81.73	65.22	32.86	43.69	69.52	62.19	62.71
XDice	70.85	96.26	81.62	70.03	18.08	28.73	70.44	57.17	55.18
XXDice	76.10	92.54	83.52	72.15	39.88	51.35	74.12	66.21	67.43
LCSR	82.15	89.30	85.47	72.65	59.93	65.66	77.40	74.62	75.57
NLS	82.39	86.03	84.24	68.47	61.84	64.95	75.43	73.93	74.59
LCSR (2gr)	76.92	81.28	79.03	56.16	49.52	52.58	66.54	65.40	65.80
NLS (2gr)	76.80	81.02	78.84	55.74	49.31	52.26	66.27	65.17	65.55
LCSR (3gr)	73.28	91.88	81.53	64.63	30.67	41.59	68.95	61.28	61.56
NLS (3gr)	73.34	91.60	81.46	64.16	31.10	41.87	68.75	61.35	61.67
Jaro-Winkler	77.06	90.72	83.33	69.72	44.10	53.99	73.39	67.41	68.66
SpSim (opt.1)	86.01	79.01	82.35	62.83	73.38	67.68	74.42	76.19	75.02
SpSim (opt.2)	83.36	80.37	81.82	62.21	66.76	64.37	72.79	73.56	73.10
Combined	89.33	90.63	89.97	80.63	77.60	78.78	84.68	84.11	84.38

Table 3: Precision (Prec), Recall (Rec) and F1-score for the individual orthographic similarity features and for the classifier combining all features (%).

random forest was trained in order to further investigate the importance of each metric. Since a random forest uses lots of seeds (in our case: 123) in order to decide on the importance of each variable individually, it provides a somewhat more representative, validated picture of the influence of different metrics. An additional Somers' D value was computed for the random forest in order to check the goodness of fit. With a correlation score of 0.9528739, our random forest forms a good model for unseen data. Figure 2 shows that the model agrees with the conditional inference tree in that it also classifies LCSR, SpSim (option 1) and the Jaro-Winkler similarity as important metrics for the identification of cognates. It does, however, provide some additional information, as it shows that the normalized Levenshtein similarity is also very influential for this binary classification task.

5.2 Experiment 2: Orthographic and Semantic Features

In a second set of experiments, we combined all orthographic similarity features with a semantic feature expressing the cosine similarity between the two word embeddings. Table 4 shows the result of the classifiers incorporating (1) only semantic information and (2) a combination of orthographic and semantic similarity information. As this set of experiments is only conducted on that part of the data set for which word embeddings were retrieved, we also added the updated performance scores for all individual orthographic metrics on this reduced data set.

The classification results listed in Table 4 show some interesting findings. First of all, the embeddings in isolation already obtain good classification results for the "Cognates" class (F-score of 89.14%). Second, the classifier combining orthographic and semantic similarity features clearly outperforms the classifier only incorporating orthographic information.

An analysis of the output reveals that the semantic information indeed helps to detect cognate pairs showing less orthographic resemblance (e.g. *east–oost, older–ouderen, widespread–wijdverbreid, asleep–slaap, sweating–zweten, shame–schaamte, belief–geloof, whole–hele, swarm–zwerm, overheated–oververhitte*). In addition, the word embedding information also generates less false negatives. Examples of pairs that were wrongly labeled as cognates by the classifier relying on orthographic information and that are correctly labeled as non-cognates by the combined classifier are: *affects–effecten, unlocking–blokkering, investments–instrument, slit–gesplit, provide–profielen, brazier–branden, might–high, where–wateren*. On the other hand,

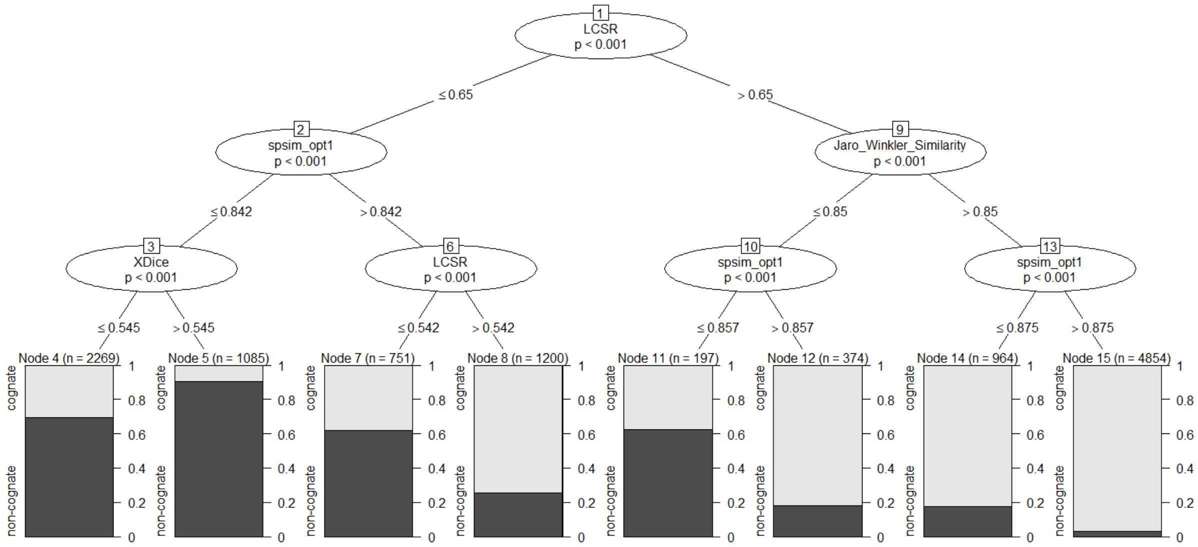


Figure 1: Conditional Inference Tree with depth 3 trained on the orthographic similarity features.

	Cognates			Non-cognates			Average score		
Metric	Prec	Rec	F-score	Prec	Rec	F-score	Prec	Rec	F-score
Prefix	82.30	87.99	85.05	62.74	51.66	56.65	72.52	69.82	70.85
Dice	80.40	91.23	85.47	65.84	43.19	52.15	73.12	67.21	68.81
Dice (3gr)	79.94	91.28	85.23	65.05	41.48	50.65	72.50	66.38	67.94
Jaccard	80.71	90.74	85.43	65.34	44.58	52.98	73.02	67.66	69.20
XDice	76.45	95.94	85.09	70.25	24.50	36.32	73.35	60.22	60.70
XXDice	79.89	94.15	86.43	72.56	39.45	51.09	76.22	66.80	68.76
LCSR	83.79	91.99	87.70	72.73	54.55	62.32	78.26	73.27	75.01
NLS	85.23	88.48	86.83	67.42	60.83	63.95	76.33	74.66	75.39
LCSR (2gr)	78.77	91.57	84.69	63.16	36.94	46.60	70.96	64.25	65.64
NLS (2gr)	79.09	90.57	84.44	61.69	38.82	47.64	70.39	64.69	66.04
LCSR (3gr)	79.94	91.28	85.23	65.05	41.48	50.65	72.49	66.38	67.94
NLS (3gr)	80.04	90.97	85.15	64.57	42.05	50.92	72.30	66.51	68.04
Jaro-Winkler	82.24	91.31	86.54	69.11	49.64	57.76	75.67	70.47	72.15
SpSim (opt.1)	85.58	81.05	83.23	57.40	65.01	60.89	71.49	73.03	72.06
SpSim (opt.2)	80.87	87.42	83.99	59.57	47.02	52.33	70.22	67.22	68.16
Sem	83.56	95.53	89.14	82.00	51.99	63.62	82.78	73.76	76.38
Ortho	89.46	91.23	90.33	76.42	72.54	74.42	82.94	81.88	82.38
Ortho + Sem	92.59	94.65	93.61	85.52	80.64	83.00	89.05	87.64	88.30

Table 4: Precision (Prec), Recall (Rec) and F1-score for the classifiers incorporating the fifteen individual orthographic features, the classifier incorporating only semantic information (*Sem*), the classifier incorporating the combined orthographic information (*Ortho*) and the classifier incorporating both orthographic and semantic similarity features (*Ortho + Sem*).

the combined classifier rarely introduces additional false negatives (seven instances in total, e.g. *lead–leiden, include–inhouden, docker–dokwerker*) or additional false positives (three instances in total: *told–toen, escapologist–escapist, because–bepaalde*).

6 Conclusion and Future Work

This paper presents preliminary experiments for combining orthographic and semantic similarity information for cognate detection. The experimental results already show promising scores for

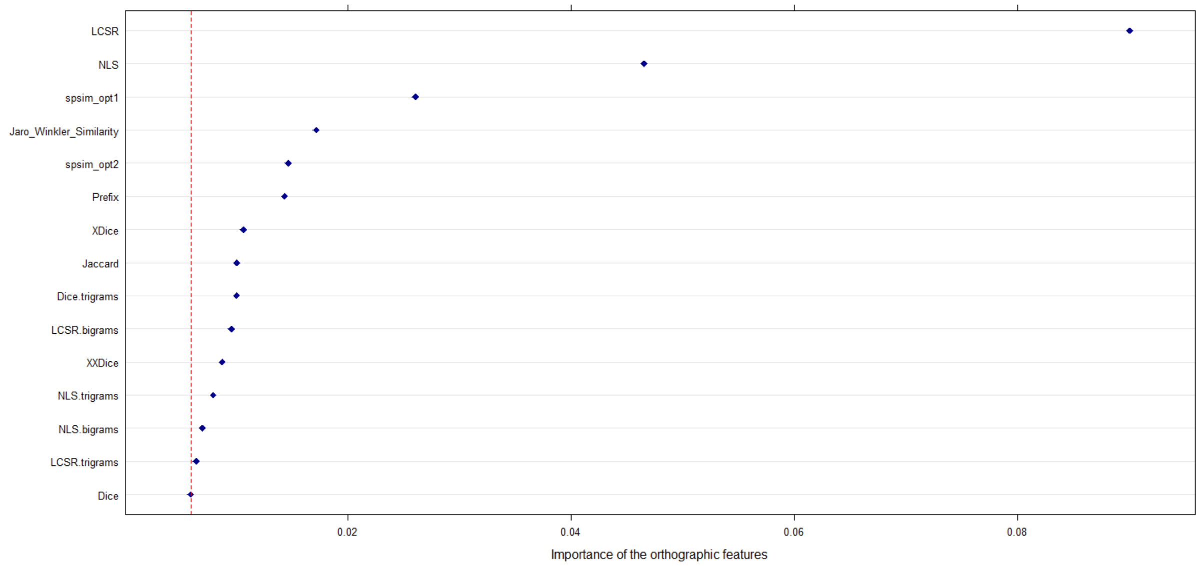


Figure 2: Random Forest indicating the importance the different orthographic similarity features for the current cognate identification task.

a classifier using merely orthographic similarity information. The results, however, revealed that adding semantic information capturing the cosine similarity between the word embeddings of the Dutch and English terms further improves the classification results considerably. As a result, we can conclude that combining orthographic and semantic similarity information is a viable approach to automatic cognate detection.

As we presented proof-of-concept results in this research, there is still a lot of room for future research. Firstly, the implementation of alternative word embeddings is an important direction for future work. We will perform additional experiments with (1) larger and different (e.g. domain-specific) corpora and (2) other embedding approaches to improve the semantic information based on embedding distance. We are confident this will result in high-level quality embeddings for all candidate cognate pairs.

Secondly, it would be interesting to perform multi-class experiments, where a distinction is made between cognates, false friends and non-related word pairs. To this end, a training and evaluation corpus containing cognate candidates in context will be built and manually annotated.

Finally, we plan to compile the corresponding gold standard set for French-Dutch, which is also part of the Dutch Parallel Corpus. This will allow an evaluation of our approach for a different language pair. In addition, this will enable us to perform trilingual machine learning experiments and to gain useful insights into cross-lingual cognate detection.

References

M. Baroni, G. Dinu, and G. Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 238–247.

S. Bergsma and G. Kondrak. 2007. Alignment-Based Discriminative String Similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. pages 656–663.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5:135–146.

C. Brew and D. McKelvie. 1996. Word-pair extraction for lexicography. In *Proceedings of the 2nd International Conference on New Methods in Language Processing*. pages 45–55.

S. Castro, J. Bonanata, and A. Rosá. 2018. A High Coverage Method for Automatic False Friends Detection for Spanish and Portuguese. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. pages 29–36.

A. Ciobanu and L. Dinu. 2014. Automatic Detection of Cognates Using Orthographic Alignment. In

52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference. volume 2, pages 99–105.

D. Crystal. 2008. *A Dictionary of Linguistics and Phonetics*. The language library. Blackwell, 6th edition.

O. Frunza and D. Inkpen. 2007. A tool for detecting French-English cognates and false friends. In *Actes de la 14me confrence sur le Traitement Automatique des Langues Naturelles*. Association pour le Traitement Automatique des Langues, pages 91–100.

L. Gomes and J. G. Pereira Lopes. 2011. Measuring Spelling Similarity for Cognate Identification. In L. Antunes and H. S. Pinto, editors, *Progress in Artificial Intelligence*. Springer, pages 624–633.

S. T. Gries. 2004. Shouldn't It Be Breakfunch? A Quantitative Analysis of Blend Structure in English. *Linguistics* 42(3):639–667.

S. Hansen-Schirra, J. Nitzke, and K. Oster. 2017. Predicting cognate translation. In S. Hansen-Schirra, O. Czulo, and S. Hofmann, editors, *Empirical modelling of translation and interpreting*, Language Science Press, chapter 1, pages 3–22.

Z. S. Harris. 1954. Distributional Structure. *WORD* 10(2-3):146–162.

B. Hauer and G. Kondrak. 2011. Clustering Semantically Equivalent Words into Cognate Sets in Multilingual Lists. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 865–873.

W. Heeringa, J. Nerbonne, and P. Osenova. 2010. Detecting contact effects in pronunciation. In M. Norde, B. de Jonge, and C. Hasselblatt, editors, *Language Contact. New Perspectives.*, Benjamins, pages 131–154.

G. Kondrak. 2000. A New Algorithm for the Alignment of Phonetic Sequences. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. pages 288–295.

G. Kondrak and B. Dorr. 2004. Identification of Confusable Drug Names: A New Approach and Evaluation Methodology. In *Proceedings of the 20th International Conference on Computational Linguistics*. pages 952–958.

S. Labat, L. Vandevoorde, and E. Lefever. 2019. Annotation Guidelines for Labeling English-Dutch Cognate Pairs, version 1.0. Technical report, Ghent University, LT3 15-01.

V. I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR* 163(4):845–848.

N. Ljubešić and D. Fišer. 2013. Identifying False Friends between Closely Related Languages. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*. pages 69–77.

L. Macken, O. De Clercq, and H. Paulussen. 2011. Dutch Parallel Corpus: A Balanced Copyright-Cleared Parallel Corpus. *Meta* 56(2):374–390.

I. D. Melamed. 1999. Bitext Maps and Alignment via Pattern Recognition. *Computational Linguistics* 25(1):107–130.

T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781.

G. A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11):39–41.

R. Mitkov, V. Pekar, D. Blagoev, and A. Mulloni. 2007. Methods for extracting and classifying pairs of cognates and false friends. *Machine Translation* 21(1):29–53.

F.J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1):19–51.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Mller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

T. Rama. 2016. Siamese convolutional networks based on phonetic features for cognate identification. *CoRR* abs/1605.05172.

J. Schepens, T. Dijkstra, and F. Grootjen. 2012. Distributions of cognates in Europe as based on Levenshtein distance. *Bilingualism: Language and Cognition* 15(1):157–166.

J. Schepens, K. Paterson, T. Dijkstra, F. Grootjen, and W. J. B. van Heuven. 2013. Cross-Language Distributions of High Frequency and Phonetically Similar Cognates. *PLOS ONE* 8:1–15.

M. Shlesinger and B. Malkiel. 2005. Comparing Modalities: Cognates as a Case in Point. *Across Languages and Cultures* 6(2):173–193.

S. L. Smith, D. H. P. Turban, S. Hamblin, and N. Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR* abs/1702.03859.

Resolving Pronouns for a Resource-Poor Language, Malayalam Using Resource-Rich Language, Tamil

Sobha Lalitha Devi
AU-KBC Research Centre, Anna University, Chennai
sobha@au-kbc.org

Abstract

In this paper we give in detail how a resource rich language can be used for resolving pronouns for a less resource language. The source language, which is resource rich language in this study, is Tamil and the resource poor language is Malayalam, both belonging to the same language family, Dravidian. The Pronominal resolution developed for Tamil uses CRFs. Our approach is to leverage the Tamil language model to test Malayalam data and the processing required for Malayalam data is detailed. The similarity at the syntactic level between the languages is exploited in identifying the features for developing the Tamil language model. The word form or the lexical item is not considered as a feature for training the CRFs. Evaluation on Malayalam Wikipedia data shows that our approach is correct and the results, though not as good as Tamil, but comparable.

1 Introduction

Natural language processing techniques of the present day require large amounts of manually-annotated data to work. In reality, the required quantity of data is available only for a few languages of major interest. In this work we show how a resource-rich language, Tamil, can be leveraged to resolve anaphora for a related resource-poor language, Malayalam. Both Tamil and Malayalam belong to the same language family, Dravidian. The methods we focus on exploits the similarity at the syntactic level of the languages and anaphora resolution heavily depends on syntactic features. If the resources available in one language (henceforth referred to as source) can be used to facilitate the resolution, such as anaphora, for all the languages related to the language in question (target), the problem of unavailability of resources would be alleviated.

There exists a recent research paradigm, in which the researchers work on algorithms that can rapidly develop machine translation and other tools for an obscure language. This work falls into this paradigm, under the assumption that the language in question has a less obscure sibling. Moreover, the problem is intellectually interesting. While there has been significant research in using resources from another language to build, for example, parsers, there have been very little work on utilizing the close relationship between the languages to produce high quality tools such as anaphora resolution (Nakov, P and Tou Ng,H 2012). In our work we are interested in the following questions:

If two languages are from the same language family and have similarity in syntactic structures and not in lexical form and script

1. Can the language model developed for one language be used for analyzing the other language?
2. How the lexical form difference can be resolved in using the language model?
3. How to overcome the challenges of script variation?

In this work we develop a language model for resolving pronominals in Tamil using CRFs and using the language model test another language, Malayalam. As said earlier, in this work, we are focusing on Dravidian family of languages. Historically, all Dravidian languages have branched out from a common root Proto-Dravidian. Among the Dravidian languages, Tamil is the oldest language. Though there is similarity at the syntactic level, there is no similarity in lexical form or at the script level among the languages. We are motivated by the observation that related languages tend to have similar word order and syntax, but they do not have similar script or orthography. Hence words are not similar.

Tamil has the most resources at all levels of linguistics, right from morphological analyser to discourse parser and Malayalam has the least. About the similarity of the two languages we give in detail in section 2.

The remainder of this article is organized as follows: Section 2 provides a detailed description of the two languages, their linguistic similarity credits and differences, Section 3 presents the pronominal resolution in Tamil. In Section 4 we introduce our proposed approach on how the language model for Tamil can be used for resolving Malayalam pronouns. Section 5 describes the datasets used for evaluation, experiments and analysis of the results and the paper ends with the conclusion in Section 6.

2 How similar the languages Tamil and Malayalam Are?

As mentioned earlier, both the languages belong to the Dravidian family and are relatively free word order languages, inflectional (rich in morphology), agglutinative and verb final. They have Nominative and Dative subjects. The pronouns have inherent gender marking as in English and have the same lexical form "avan" "he", "aval" "she" and "atu" "it" both in Tamil and Malayalam. Though the pronouns have same lexical form and meaning, it can be said that there is no lexical similarity between the two languages. The similarity between two languages can be at three levels, a) writing script, b) the word forms and c) the syntactic structure.

Script Level: The two languages have different writing form, though the base is from Grandha script. Hence no similarity at the script level.
The Word Level: There is no similarity at the lexical level between the two languages. The Sanskritization of Malayalam contributed to have more Sanskrit verbs in Malayalam whereas Tamil retained the Proto –Dravidian verbs. For example, the word for "talk" in Malayalam is "samsarikkuka" "to talk" which has root in Sanskrit, whereas the Tamil equivalent is "pesuu" "to talk", the root in Pro-Dravidian.
The Syntactic Structure Level: There is lot of similarity at the syntactic structure level between the two languages. Since antecedent to anaphor has dependency on the position of the noun, the structural similarity is a positive feature for our goal. The syntactic similarity at Sentence level, Case maker level, pronominal distribution level are explained with examples.
Case marker level: Both the languages have the same number of cases and their distribution is similar. In both the languages, nouns inflected with nominative or dative case become the subject of the sentence (Dative subject is the peculiarity of Indian languages). Accusative case denotes the object.
Clausal sentences: The clause constructions in both the languages follow the same rule. The clauses are formed by nonfinite verbs. The clauses do not have free word order and they have fixed positions. Order of embedding of the subordinate clause is same in both the languages.
Ex:1

(Ma) [innale vanna(vbp) kutti]/Sub-RP-cl
{sita annu}/Main cl

(Ta) [neRu vanta(vbp) pon]/Sub- RP-cl
{sita aakum}/Maincl

[Yesterday came girl]/subcl
{Sita is}/ Maincl

(The girl who came yesterday is Sita)

As can be seen from the above example, the basic syntactic structure is the same in both the languages. The above example is a two clause sentence with a relative participial clause and a main clause. The relative participial clause is formed by the nonfinite verb (vbp). Using the same example we can find the pronominal distribution.
Ex: 2

(Ma) [innale vanna(vbp) aval$_i$ (PRP)
]/Sub-RP-cl {sita$_i$ annu}/Main cl
(Ta) [neRu vanta(vbp) aval$_i$ (PRP)
]/Sub-RP-cl {sita$_i$ aakum}/Maincl
[Yesterday came she$_i$ (PRP)
]/subcl { Sita$_i$ is} / Maincl
(The she who came yesterday is Sita)

In the above example the pronoun "aval" "she" occurs at the same position in both the languages and the antecedent “sita” also occurs at the same position as shown by co-indexing. Consider another example.

Ex:3

(Ma). sithaa$_i$ kadaikku pooyi. aval$_i$ pazham
Vaangicchu(Vpast)

(Ta). sithaa$_i$ kadaikku cenRaal. aval$_i$ pazham
Vaangkinaal(V,past,+F,+Sg).
Sita shop went. She fruit
bought
(Sita$_i$ went to the shop. She$_i$ bought fruit.)

In the above example there are two sentences and pronoun is in one sentence and antecedent is in another. Here you can see the distribution of the pronoun “aval” and where the antecedent “sita” is occurring. Though Tamil has number, gender and person agreement between subject and verb and Malayalam does not have, this cannot be considered as a grammatical feature which can be used for identifying the antecedent of an anaphor. This grammatical variation does not have an impact on the identification of pronoun and antecedent relations. From the above examples we can see that the two languages have the same syntactic structure at the clause and sentence level. We are exploiting this similarity between the two languages to achieve our goal. We find that using this similarity between the languages, the language model of Tamil can be used to resolve pronouns in Malayalam.

3 Pronoun Resolution in Tamil

3.1 Pronouns in Tamil

In this section, we analyse in detail the pronominal expressions in Tamil. Pronouns are the words used as a substitution to nouns, that are already mentioned or that is already known. There are pronouns which do not refer. Pronouns in Tamil have person (1st, 2nd, 3rd person) and number (singular, plural) distinction. Masculine, feminine and neuter gender distinctions are clearly marked in 3rd person pronouns, whereas in 1st and 2nd person pronouns there is no distinction of masculine, feminine and neuter gender. In this work we consider only third person pronouns. Third person pronouns in Tamil have inherent gender and as in English and they are “avan” he, “aval” she and “atu” it. In this work, we resolve 3rd person pronouns. The distribution of pronouns in various syntactic constructions is explained with examples below.

Ex:4

4a. ***maaNavarkaL****$_i$ paLLikku celkiranar.*
Students(N) school(N)+dat go(V)+present+3pl
(Students are going to the school)

4b. ***avarkaL****$_i$ veekamaaka natakkinranar.*
They(PN) fast(ADV) walk(V)+present+3pl
(They are walking fast.)

Considering Ex 4a and Ex.4b, sentence Ex.4b has 3rd person plural pronoun ‘avarkaL’ as the subject and it refers to plural noun ‘maaNavarkaL’ which is the subject in Ex.4a.

Ex:5

5a. ***raamuvum****$_i$* ***giitavum****$_j$ nanparkaL.*
Raamu(N)+INC Gita(N)+INC friends(N)
(Ramu and Gita are friends.)

5b. ***avan****$_i$ ettaam vakuppil padikkiraan.*
He(PN) eight(N) class(N) study(V)+present+3sm
(He studies in eight standard.)

5c. ***avaLum****$_j$ ettaam vakuppil padikkiaal.*
She(PN) eight(N) class(N) study(V)+present+3sf
(She also studies in eight standard.)

In Ex.5b, 3rd person masculine pronoun ‘avan’ occurs as the subject and it refers to the masculine noun ‘raamu’, subject noun in Ex.5a. Similarly, 3rd person feminine pronoun ‘avaL’ in Ex.5c refers to feminine noun ‘giita’ in Ex.3a. ‘atu’, which is a 3rd person neuter pronoun, will also occurs as genitive/possessive case marker. Consider the following example.

3.1.1 Non-anaphoric Pronouns

The pronouns can also occur as generic mentions without having referent. In English it known as‘pleonastic it’.

Ex:6.

atu *oru malaikalam.*
It(PN) one(Qc) rainy_season (N)
(It was a rainy season.)

In Ex.6, the 3rd person neuter pronoun ‘atu’ (it) do not have a referent. Here ‘atu’ is equivalent to pleonastic ‘it’ in English

3.1.2 Corpus annotation

We collected 600 News articles from various online Tamil News wires. The News articles are from Sports, Disaster and General News domains. The anaphoric expressions are annotated along with its antecedents using graphical tool, PAlinkA, a highly customisable tool for Discourse Annotation (Orasan, 2003) which we customized for Tamil. We have used two tags namely, MARKABLE and COREF. The corpus used for training is 54,563 words which are annotated for anaphora –antecedent pairs and testing corpus is 10,912 words. We have calculated the inter-annotator agreement which is the degree of agreement among annotators. We have used Cohen's kappa as the agreement statistics. The kappa coefficient is generally regarded as the statistics of choice for measuring agreement on ratings made on a nominal scale. We got a Kappa score of 0.87. The difference between the annotators were analysed and found the variation in annotation. It occurred in the marking of antecedents for pronominal. This is common in sentences with clausal inversion, and genitive drop.

3.2 Pronoun Resolution System

Early works in anaphora resolution by Hobbs (1978), Carbonell and Brown (1988), Rich and Luper Foy (1988) etc. were mentioned as knowledge intensive approach, where syntactic, semantic information, world knowledge and case frames were used. Centering theory, a discourse based approach for anaphora resolution was presented by Grosz (1977), Joshi and Kuhn (1979). Salience feature based approaches were presented by Lappin and Leass (1994), Kennedy Boguraev (1996) and Sobha et al., (2000). Indicator based, knowledge poor method for anaphora resolution methods were presented by Mitkov (1997, 1998). One of the early works using machine learning technique was Dagan Itai's (1990) unsupervised approach based on co-occurrence words. With the use of machine learning techniques researchers work on anaphora resolution and noun phrase anaphora resolution simultaneously. The other machine learning approaches for anaphora resolution were the following. Aone and Bennett (1995), McCarty and Lahnert (1995), Soon et al., (2001), Ng and Cardia (2002) had used decision tree based classifier. Anaphora resolution using CRFs was presented by McCallum and Wellner (2003) for English, Li et al., (2008) for Chinese and Sobha et al., (2011, 2013) for English and Tamil.

In Indian languages anaphora resolution engines are demonstrated only in few languages such as Hindi, Bengali, Tamil, and Malayalam. Most of the Indian languages do not have parser and other sophisticated pre-processing tools. The earliest work in Indian language, 'Vasisth' was a rule based multilingual anaphora resolution platform by Sobha and Patnaik (1998, 2000, 2002), where the morphological richness of Malayalam and Hindi were exploited without using full-parser. The case marker information is used for identifying subject, object, direct and in-direct object. Prasad and Strube (2000), Uppalapu et al., (2009) and Dekwale et al., (2013) had presented different approaches using Centering theory for Hindi. Sobha et al., (2007) presented a salience factor based with limited shallow parsing of text. Akilandeswari et al., (2013) used CRFs for resolution of third person pronoun. Ram et al., (2013) used Tree CRFs for anaphora resolution for Tamil with features from dependency parsed text. In most of the published works resolution of third person pronoun was considered and it is a non-trivial task.

Pronoun resolution engine does the task of identifying the antecedents of the pronouns. The Pronoun resolution is built using Conditional Random Fields (CRFs) technique. Though CRFs is notable for sequence labelling task, we used this technique to classify the correct anaphor-antecedent pair from the possible candidate NP pairs by presenting the features of the NP pair and by avoiding the transition probability. While training we form positive pairs by pairing anaphoric pronoun and correct antecedent NP and negative pairs by pairing anaphoric pronouns and other NPs which match in person, number and gender (PNG) information with the anaphoric pronoun. These positive and negative pairs are fed to the CRFs engine and the language model is generated. While testing, when an anaphoric pronoun occurs in the sentence, the noun phrases which match in PNG with the pronoun, that occur in the preceding portion of the sentence and the four preceding sentences are collected and paired with the anaphoric pronoun and presented to CRFs engine to identify the correct anaphor-antecedent pair.

3.2.1 Pre-processing

The input document is processed with a sentence splitter and tokeniser to split the document into sentences and the sentences into individual tokens which include words, punctuation markers and symbols. The sentence split and tokenized documents are processed with Syntactic Processing modules. Syntactic processing modules include Morphological analyser, Part-of-Speech tagger and Chunker. These modules are developed in house.

a) **Morphological Analyser:** Morphological analysis processes the word into component morphemes and assigning the correct morpho-syntactic information. The Tamil morphological Analyser is developed using paradigm based approach and implemented using Finite State Automata (FSA) (Vijay Sundar et.al 2010). The words are classified according to their suffix formation and are marked as paradigms. The number of paradigms used in this system is Noun paradigms: 32; Verb Paradigms: 37; Adjective Paradigms: 4; Adverb Paradigms: 1. A root word dictionary with 1, 52,590 root words is used for developing the morphological anlyser. The morphological analyser is tested with 12,923 words and the system processed 12,596 words out of which it correctly tagged 12,305. The Precision is 97.69% and Recall = 97.46%. MA returns all possible parse for a given word.

b) **Part Of Speech Tagger (**POS tagger): Part of Speech tagger disambiguates the multiple parse given by the morphological analyser, using the context in which a word occurs. The Part of speech tagger is developed using the machine learning technique Conditional Random fields (CRF++) (Sobha L, et.al 2010). The features used for machine learning is a set of linguistic suffix features along with statistical suffixes and uses a window of 3 words. We have used 4, 50,000 words, which are tagged using BIS POS tags. The system performs with recall 100% and Average Precision of 95.16%.

c) **Noun and Verb Phrase Chunker:** Chunking is the task of grouping grammatically related words into chunks such as noun phrase, verb phrase, adjectival phrase etc. The system is developed using the machine learning technique, Conditional Random fields(CRF++) (Sobha L et.al 2010). The features used are the POS tag, Word and window of 5 words. Training Corpus is 74,000 words. The recall is 100%. Average Precision of 92.00%.

3.3 Pronoun Resolution Engine

In both training and testing phase, the noun phrases (NP) which match with the PNG of the pronoun are considered. The features are extracted from these NPs. In the training phase the positive and negative pairs are marked and fed to the ML engine for generating a language model. In the testing phase these NPs with its features are input to the language model to identify the antecedent of a pronoun. Here we have not taken the lexical item or the word as a feature. We have used only the grammatical tags as feature. The features selected represent the syntactic position of the anaphor –antecedent occurrence.

3.3.1 Features Selection

The features required for machine learning are identified from shallow parsed input sentences. The features for all possible candidate antecedent and pronoun pairs are obtained by pre-processing the input sentences with morphological analyser, POS tagger, and chunker. The features identified can be classified as positional and syntactic features

Positional Features: The occurrence of the candidate antecedent is noted in the same sentence where the pronoun occurs or in the prior sentences or in prior four sentences from the current sentence.

Syntactic Features: The syntactic arguments of the candidate noun phrases in the sentence are a key feature. The arguments of the noun phrases such as subject, object, indirect object, are obtained from the case suffix affixed with the noun phrase. As mentioned in section 2 the subject of a sentence can be identified by the case marker it takes. We use morphological marking for the above.

a) PoS tag and chunk tag of Candidate NP, case marker tags of the noun.
b) The suffixes which show the gender which gets attached to the verb.

3.3.2 Development of Tamil Language Model

We used 600 Tamil Newspaper articles for building the language model. The preparation of the training data is described below. The raw corpus is processed with sentence splitter and tokeniser. The tokenized corpus is then preprocessed with shallow processing modules, namely, morphological analyser, part-of-speech tagger and chunker. The training data is prepared from this processed corpus. For each pronoun, the Noun phrase (NP) preceding the pronouns and in the NPs in preceding sentence till correct antecedent NP, which match in Person, number and Gender (PNG) are selected for training. The above features are used for CRF for learning. The system was evaluated with data from the web and the result is given below.

Domain	Testing Corpus (Words)	Precision (%)	Recall (%)
News data	10,912	86.2	66.67

Table 1: Pronominal Resolution (CRFs engine)

4. Resolution of Pronouns in Malayalam Corpus using Tamil Language Model

In this section, we present in detail how Malayalam is tested using Tamil language model. Here Malayalam data is pre-processed as per the requirement of the Tamil language model test data. The test data required four grammatical information, i) POS, ii) the case marker, iii) the number gender and person and iv)chunk information. In the introduction we have asked three questions on how to use a language model in source language be used for testing a target language. The three questions are dealt one by one below.

1. Can the language model developed for one language be used for analyzing the other language?

In this study we have used the language model of the source language Tamil. The features used to develop this language model are POS tag, Chunk tag and the case/ suffix tags. The word form was not considered as a feature. Since these are the features used for learning, the test data also should have these information. As said earlier, Malayalam is not a resource rich language and it does not have pre-processing engines such as POS tagger, Chunker and morphological analysers with high accuracy. Hence we developed a very rudimentary preprocessing systems which can give the POS tag, case/suffix tags and chunk tags.

The POS information: The POS and suffix information are assigned to the corpus using a root word dictionary with part of speech information and a suffix dictionary.

The dictionary has the root words which include all types of pronouns and contains nearly 66,000 root words. The grammatical information (POS) such as noun, verb, adjective, pronoun and number gender person (PNG) information are given for a word in the dictionary.

The suffix dictionary contains all possible suffixes which a root word can take. The suffix includes the changes in sandhi when added to the root word. The suffixes are of two types, i) that which gets attached to nouns called the case suffixes and 2) that which gets attached to verbs called the TAM (Tense, Aspect, and Modal). Using this suffix dictionary we can identify the POS of the word even if the word is not present in the root word dictionary. The suffix dictionary has 1,00,000 unique entries.

The noun chunk information is given by a rule based chunker which works on three linguistic rules. The noun phrases alone are required for anaphora resolution. Chunks are identified using the basic linguistic rule for Noun phrases as given below

1. [determiner] [quantifier][intensifier] [classifier][adjective] {Head Noun}. Here Head noun is obligatory and others are all optional
2. NN+NN combination

3. NN with no suffix+NN with no suffix…… +NN with suffix or without suffix.

Using the above rules we identified the noun chunks in Malayalam. The above discussed preprocessing gives an accuracy of 66% for POS and suffix tagging and 63% for chunking.

2. How the lexical form difference can be resolved in using the language model?

The second question we asked is about the lexical form or words which are not similar in both the languages and how this can be resolved. The analysis of Tamil has shown that the syntactic structure of the language has more prominence over the words in the resolution of anaphors. Hence we have taken the syntactic features and did not take word as a feature. The system learned only the structure patterns.

3. How to overcome the challenges of script variation?

Since word feature is not considered the script do not pose any challenges. Still to have the same script we converted the two languages into one form the WX notation. This helped in having the same representation of the languages.

4. Testing with Malayalam Data

We selected 300 articles from Malayalam Wikipedia, which were on different genre and size. The 300 Malayalam documents from Wikipedia has 7600 sentences with 3660 3rd person pronouns. The distribution of the pronouns is presented in Table 2.

Pronoun	Number of Occurrences with its Inflected forms
avan (3rd person masculine singular)	1120
aval (3rd person feminine singular)	840
avar (3rd person honorific)	420
athu (3rd person nuetor singular)	1280
Total	3660

Table 2. Distribution of 3rd person pronouns in the corpus

As discussed in the earlier section, the preprocessing done for Tamil using syntactic module are morphological information, POS and Chunking information. Hence the same pre-processing information is necessary for Malayalam data as well. The documents are initially preprocessed with a sentence splitter and tokenizer. The tokenized documents are pre-processed using the pos, suffix and chunking systems discussed above to enrich the text with syntactic information.

For each pronoun, we identify the possible candidate antecedents. Those noun phrases that occur preceding the pronoun in the current sentences and preceding four sentences, which match in the person, number gender (PNG) with the select pronoun are identified as possible candidates. For these possible candidates we extract the features required for CRFs techniques as explained in the previous section. After extraction of features for selected candidate antecedents, the antecedent is identified by using language model built using Tamil data. The results are encouraging with 67% accuracy which is a respectable score. The errors and evaluation is given in detail in the next section.

5. Experiment and Discussion

The experiment showed that the resolution of the pronouns “avan” he and “aval” she is similar to that of Tamil documents. The issues related to split antecedent is not addressed and hence pronouns which are referring to coordinated nouns were not resolved. The pronoun which is less resolved is the third person neuter pronouns “atu” compared to other pronouns. The third person neuter pronoun usually has more number of possible candidates, which leads to poor resolution. Consider the following example.

Ex:7

avan	joli	ceytha	jolikkarkku
He(PRP)	work(N)	do(V)+past+RP	worker(N)+pl+dat
vellam	kotuthu.		
water(N)	give(V)+past		

(He gave water to the workers who did the work.)

atu	nallatu	aayirunnu
It(PRP)	good(N)	is (Copula V) +past.

(It was good.)

In this example, the pronoun 'atu' refers to 'vellam” ‘water’; in the previous sentence. In the previous sentence there are two possible candidate antecedents 'vellam' and 'joli 'which are 3rd person nouns. The ML engine chooses 'joli', which is in the initial position of the sentence and in the subject position. When the antecedent is in the object position the engine has not identified it properly. The following table gives the results of pronouns.

Type of pronoun	Precisi on (%)	Recal l (%)
avan (3rd person masculine singular)	70.83	69.52
avaL (3rd person feminine singular)	69.34	68.56
avar/avarkal (3rd person plural/honorofic)	65.45	70.34
atu (3rd person nuetor singular)	56.67	65.67
Total	68.45	67.34

Table 3: Evaluation Results

6. Conclusion

In this paper we explained a method to use high resource language to resolve anaphors in less resource language. In this experiment the high resource language is Tamil and the less resource language is Malayalam. The results are encouraging. The model needs to be tested with more data as future work.

References

Carbonell J. G., and Brown R. D. 1988. *Anaphora resolution: A multi- strategy approach*. In: 12th International Conference on Computational Linguistics, 1988, pp. 96-101.

Dagan I., and Itai. A. 1990. *Automatic processing of large corpora for the resolution of anaphora references*. In: 13th conference on Computational linguistics, Vol. 3, Helsinki, Finland, pp.330-332.

Dakwale. P., Mujadia. V., Sharma. D.M. 2013. *A Hybrid Approach for Anaphora Resolution in Hindi.* In: Proc of International Joint Conference on Natural Language Processing, Nagoya, Japan, pp.977–981.

Li., F., Shi., S., Chen., Y., and Lv, X. 2008. *Chinese Pronominal Anaphora Resolution Based on Conditional Random Fields.* In: International Conference on Computer Science and Software Engineering, Washington, DC, USA, pp. 731-734.

Hobbs J. 1978. *Resolving pronoun references*. Lingua 44, pp. 339-352.

Grosz, B. J. 1977. *The representation and use of focus in dialogue understanding. Technical Report* 151, SRI International, 333 Ravenswood Ave, Menlo Park, Ca. 94025.

Joshi A. K., and Kuhn S. 1979. *Centered logic: The role of entity centered sentence representation in natural language inferencing*. In: International Joint Conference on Artificial Intelligence.

Kennedy, C., Boguraev, B. 1996 *Anaphora for Everyone: Pronominal Anaphora Resolution without a*

Parser. In: 16th International Conference on Computational Linguistics COLING'96, Copenhagen, Denmark, pp. 113–118.

Lappin S., and Leass H. J. 1994. *An algorithm for pronominal anaphora resolution*. Computational Linguistics 20 (4), pp. 535-561.

McCallum A., and Wellner. B. 2003. *Toward conditional models of identity uncertainty with application to proper noun coreference*. In Proceedings of the IJCAI Workshop on Information Integration on the Web, pp. 79–84.

McCarthy, J. F. and Lehnert, W. G. 1995. *Using decision trees for coreference resolution*. In C. Mellish (Ed.), Fourteenth International Conference on Artificial Intelligence, pp. 1050-1055

Mitkov R. 1998. *Robust pronoun resolution with limited knowledge*. In: 17th International Conference on Computational Linguistics (COLING' 98/ACL'98), Montreal, Canada, pp. 869-875.

Mitkov, R. 1997. "*Factors in anaphora resolution: they are not the only things that matter. A case study based on two different approaches*". In Proceedings of the ACL'97/EACL'97 workshop on Operational factors in practical, robust anaphora resolution, Madrid, Spain.

Ng V., and Cardie C. 2002. *Improving machine learning approaches to coreference resolution*. In. 40th Annual Meeting of the Association for Computational Linguistics, pp. 104-111.

Prasad R., and Strube,M.,2000. *Discourse Salience and Pronoun Resolution in Hindi*, Penn Working Papers in Linguistics, Vol 6.3, pp. 189-208.

Orasan, C. 2003*, PALinkA: A highly customisable tool for discourse annotation*. SIGDIAL Workshop 2003: 39-43

Preslav Nakov Hwee Tou Ng 2012. *Improving Statistical Machine Translation for a Resource-Poor Language Using Related Resource-Rich Languages*; Journal of Artificial Intelligence Research 44 (2012) 179-222;

Ram, R.V.S. and Sobha Lalitha Devi. 2013.*Pronominal Resolution in Tamil Using Tree CRFs*", In Proceedings of 6th Language and Technology Conference, Human Language Technologies as a challenge for Computer Science and Linguistics - 2013, Poznan, Poland

Rich, E. and LuperFoy S.,1988 *An architecture for anaphora resolution*. In: Proceedings of the Second Conference on Applied Natural Language Processing, Austin, Texas.

Russell, B. 1919, *"On Propositions: What They Are and How They Mean,"* Proceedings of the Aristotelian Society, Supplementary Volume 2: 1–43; also appearing in Collected Papers, Vol. 8

Senapati A., Garain U. 2013. *GuiTAR-based Pronominal Anaphora Resolution in Bengal*. In: Proceedings of 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria pp 126–130.

Sikdar U.K, Ekbal A., Saha S., Uryupina O., Poesio M. 2013. *Adapting a State-of-the-art Anaphora Resolution System for Resource-poor Language*. In proceedings of International Joint Conference on Natural Language Processing, Nagoya, Japan pp 815–821.

Sobha L. and Patnaik B. N. 2000. *Vasisth: An Anaphora Resolution System for Indian Languages*. In Proceedings of International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications, Monastir, Tunisia.

Sobha L. and Patnaik,B.N. 2002. *Vasisth: An anaphora resolution system for Malayalam and Hindi*. In Proceedings of Symposium on Translation Support Systems.

Sobha L. 2007. *Resolution of Pronominals in Tamil*. Computing Theory and Application, The IEEE Computer Society Press, Los Alamitos, CA, pp. 475-79.

Sobha L., Sivaji Bandyopadhyay, Vijay Sundar Ram R., and Akilandeswari A. 2011. *NLP Tool Contest @ICON2011 on Anaphora Resolution in Indian Languages*. In: Proceedings of ICON 2011.

Sobha Lalitha Devi and Pattabhi R K Rao. 2010. *Hybrid Approach for POS Tagging for Relatively Free Word Order Languages.* In Proceedings of Knowledge Sharing Event on Part-Of-Speech Tagging, CIIL, Mysore.

Vijay Sundar Ram and Sobha Lalitha Devi. 2010. *Noun Phrase Chunker Using Finite State Automata for an Agglutinative Language.* In Proceedings of the Tamil Internet – 2010, Coimbatore, India, 218–224.

Soon W. H., Ng, and Lim D. 2001. *A machine learning approach to coreference resolution of noun phrases.* Computational Linguistics 27 (4), pp.521-544.

Taku Kudo. 2005. CRF++, an open source toolkit for CRF, http://crfpp.sourceforge.net .

Uppalapu. B., and Sharma, D.M. 2009. *Pronoun Resolution For Hindi*. In: Proceedings of 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 09), pp. 123-134.

Semantic Role Labeling with Pretrained Language Models for Known and Unknown Predicates

Daniil Larionov
FRC CSC RAS / Moscow, Russia
RUDN University / Moscow, Russia
dslarionov@isa.ru

Artem Shelmanov
Skoltech / Moscow, Russia
FRC CSC RAS / Moscow, Russia
a.shelmanov@skoltech.ru

Elena Chistova
FRC CSC RAS / Moscow, Russia
RUDN University / Moscow, Russia
chistova@isa.ru

Ivan Smirnov
FRC CSC RAS / Moscow, Russia
RUDN University / Moscow, Russia
ivs@isa.ru

Abstract

We build the first full pipeline for semantic role labelling of Russian texts. The pipeline implements predicate identification, argument extraction, argument classification (labeling), and global scoring via integer linear programming. We train supervised neural network models for argument classification using Russian semantically annotated corpus – FrameBank. However, we note that this resource provides annotations only to a very limited set of predicates. We combat the problem of annotation scarcity by introducing two models that rely on different sets of features: one for "known" predicates that are present in the training set and one for "unknown" predicates that are not. We show that the model for "unknown" predicates can alleviate the lack of annotation by using pretrained embeddings. We perform experiments with various types of embeddings including the ones generated by deep pretrained language models: word2vec, FastText, ELMo, BERT, and show that embeddings generated by deep pretrained language models are superior to classical shallow embeddings for argument classification of both "known" and "unknown" predicates.

1 Introduction

Semantic role labeling (SRL) is one of techniques for shallow semantic parsing of natural language texts that produces predicate-argument structures of sentences. Predicates bear the central meaning of a situation expressed by a text. In most semantic theories, predicates are verbs, verbal nouns, and some other verb forms. Arguments are phrases that fill meaning slots of a situation expressed by a predicate and define its essential details. They answer such questions as "who?", "did what?", "to whom?", "with what?", "where?", "when?", etc. It is said that arguments play semantic roles in a situation as roles define meanings of slots. Role meanings and sizes of role inventories vary in different semantic theories and annotated corpora. Converting a text into such shallow semantic structures helps to abstract from syntactic and morphological representations of sentences and is considered to be an important technique for natural language understanding. In (Jurafsky and Martin, 2009), this is demonstrated with the following example for a predicate *break*.

- John [AGENT] broke the window [THEME].
- John [AGENT] broke the window [THEME] with a rock [INSTRUMENT].
- The rock [INSTRUMENT] broke the window [THEME].
- The window [THEME] broke.
- The window [THEME] was broken by John [AGENT].

Note, that despite the surface syntactic representations of these sentences differ, the core predicate-argument structure retains and only adjusts to available situation details.

Semantic role labeling has been shown to be beneficial in a number of tasks, where it is important to compare or query texts by meaning: machine translation (Shi et al., 2016), question answering (Shen and Lapata, 2007), information search (Osipov et al., 2010), information extraction (Bastianelli et al., 2013), sentiment analysis (Marasović and Frank, 2018), and others.

The whole SRL process can be divided in four steps: predicate identification and identification of its frame (disambiguation), argument extraction (for each predicate), argument classification (or labeling of arguments with semantic roles), and global scoring that deals with linguistic constrains. Predicate-argument structures in some notations can be represented as two-level trees, rooted in predicates, with single tokens (nouns, adjectives, pronouns, proper names) as leaves that denote arguments. We adopt this dependency-based notation and treat the problem of semantic role labeling as constructing such trees.

There are two main types of linguistic corpora that are used for training models for SRL: FrameNet-like (Baker et al., 1998) and PropBank-like (Kingsbury and Palmer, 2002). The Russian-language resource that can be used for supervised training is a FrameBank corpus (Lyashevskaya, 2012; Lyashevskaya and Kashkin, 2015). The underlying semantic model of this resource is close to the one FrameNet is based on. The biggest difference from FrameNet besides semantic role inventory lies in the fact that FrameBank does not group several verbs into frames but introduces “frame” structures for each unique verb. The corpus contains partially annotated text samples with predicates, arguments, and their semantic roles.

A notable limitation of this resource is that there are annotations only for a very limited set of predicates. In this work, we combat the problem of annotation scarcity by introducing two classification models that rely on different sets of features: one for “known” predicates that are present in the training set and one for “unknown” predicates that are not seen in the training data. We show that the model for “unknown” predicates can deal with the lack of annotation by using pretrained embeddings. We perform experiments with various types of embeddings including the ones generated by deep pretrained language models: word2vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017), ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), and show that embeddings generated by deep pretrained language models are superior to classical shallow embeddings for semantic role labeling in both cases of “known” and “unknown” predicates.

The contribution of this paper is the following:

- We present and evaluate the first full pipeline for semantic role labeling of Russian texts. The developed models and the code are published online[1].
- We show that pretrained embeddings and language models can alleviate the problem of annotation scarcity for predicates.
- We conduct experiments that demonstrate the superiority of using embeddings generated by pretrained language models compared to shallow embeddings like word2vec and FastText.

The rest of the paper is structured as follows. Section 2 discusses the related work on semantic role labeling for Russian and other languages. Section 3 describes the developed pipeline for semantic role labeling of Russian texts. Section 4 presents the results of the experimental evaluation of the developed pipeline. Section 5 concludes and outlines the future work.

2 Related Work

The data-driven methods for semantic role labeling originate from the work (Gildea and Jurafsky, 2002), in which authors propose a statistical model based on various morpho-syntactic features and train it on the FrameNet corpus. The release of the PropBank corpus sparked a notable interest in SRL among researchers. The consecutive works and numerous shared tasks facilitated creation of elaborated machine learning models based on manually engineered lexico-syntactic features (Xue and Palmer, 2004; Punyakanok et al., 2005; Pradhan et al., 2005).

More recent works on semantic role labeling leverage deep neural networks (Collobert et al., 2011) shifting from feature-engineering to architecture-engineering. Several notable approaches suggest doing semantic role labeling in an end-to-end fashion relying only on raw low-level input consisted of characters or tokens and well-known multilayer recurrent networks (He et al., 2017; Sahin and Steedman, 2018; Marcheggiani et al., 2017). State-of-the-art approaches leverage multitask learning (Strubell et al., 2018) and self-attention techniques (Strubell et al., 2018; Tan et al., 2018). Several recent works also report that although the end-to-end approaches have

[1] https://github.com/IINemo/isanlp_srl_framebank/tree/master

managed to show comparable results, syntactic information still significantly helps semantic parsing (He et al., 2018).

It is worth noting a novel approach to creating annotated resources for semantic role labeling. In (He et al., 2015; FitzGerald et al., 2018), instead of annotating a corpora with a scheme grounded in elaborated linguistic theory, which requires highly qualified annotators, researchers suggest question-answer driven approach to construction of annotated resource based on crowd-sourcing. The recently presented QA-SRL Bank 2.0 (FitzGerald et al., 2018) is a large-scale annotated dataset built by non-experts. The construction of such a resource becomes possible due to simplicity of the annotation scheme, which provides an ability to label predicate-argument relationships using question-answer pairs.

There is a number of works devoted to automatic semantic parsing of Russian texts. In (Sokirko, 2001), a rule-based semantic parser is presented that converts a sentence into a semantic graph. The work does not provide numerical evaluation results, and the generated semantic graph is substantially different from predicate-argument structures produced in SRL. In (Shelmanov and Smirnov, 2014), authors present a rule-based semantic parser for Russian that relies on a dictionary of predicates and a set of morpho-syntactic rules created by human experts. They use this parser to automatically annotate representative corpus for supervised training of a transition-based labeler. In (Kuznetsov, 2015; Kuznetsov, 2016), an SVM-based labeling model is trained on FrameBank corpus. Authors rely on feature-engineering approach and suggest to use syntactic features and clusters of lexis. They also implement integer linear programming inference as a post processing step. These works are based on the pre-release version of the FrameBank corpus and do not provide code for data preparation, modeling, and evaluation. They also do not consider argument extraction and the problem with labeling arguments of “unknown” predicates. In (Shelmanov and Devyatkin, 2017), authors experiment with training a neural network models on the FrameBank corpus and suggest using word2vec embeddings for dealing with scarcity of predicate annotations. However, they implement only an argument classification step but not the full SRL pipeline. It is also worth noting that they performed experiments on gold-standard morphological features (POS tags and morphological characteristics), which does not reflect the real-world scenario. In this work, we additionally suggest using embeddings generated by deep pretrained language models, train models on automatically generated linguistic annotations (morphology / syntactic trees), and provide the full pipeline for semantic role labeling including argument extraction. It is also worth noting the Frame-parser project[2], however, it is in an early stage and only implements argument labeling using an SGD classifier.

3 Pipeline for Semantic Role Labeling

The limitations of the FrameBank corpus do not allow to use end-to-end / sequence labeling methods for SRL. Unlike PropBank, its text samples are annotated only partially, so they are not suitable for straightforward training of a supervised argument extractor or a combined pipeline. Therefore, we split our pipeline into multiple stages, some of which leverage rule-based methods.

The pipeline for semantic role labeling assumes that input texts are preprocessed with a tokenizer, a sentence splitter, a POS-tagger, a lemmatizer, and a syntax parser that produces a dependency tree in a Universal Dependencies format (Nivre et al., 2016). The SRL pipeline consists of the following steps: predicate identification, argument extraction, argument classification, and global scoring.

In the predicate identification step, we mark all verbs and some verb forms according to the given POS-tags of sentence tokens. We do not consider verbal nouns as predicates since they are not present in the FrameBank corpus. In the argument extraction step, for each marked predicate, we try to detect its arguments within a sentence by analyzing its syntax dependency tree with a number of manually constructed rules. The arguments in the pipeline are not spans as stated in CoNLL Shared Tasks 2004, 2005, 2012 (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005; Pradhan et al., 2012), but single tokens (nouns, proper names, or pronouns) as stated in CoNLL Shared Tasks 2008, 2009 (Surdeanu et al., 2008; Hajič et al., 2009).

For the argument classification step, we train two neural models that predict roles of arguments

[2] `https://github.com/lizaku/frame-parsing`

of "known" and "unknown" predicates accordingly. We call a predicate "known" if it appears in a training set within a labeled example, and "unknown" if it does not. During the inference we choose a model by simply checking a presence of a given predicate in a list of predicates appeared in the training corpus. The result of model inference is a set of probabilities for each semantic role in the inventory (above a certain threshold).

In the global scoring step, we enforce the final predicate-argument structure to fulfill the important linguistic constraint: in a single predicate-argument structure, each semantic role can be assigned only once, and each argument can have only one role.

The whole pipeline is schematically depicted in Figure 1.

3.1 Argument Extraction

Base arguments are extracted from a syntax tree of a sentence by rules that take into account POS-tags of tokens and direct syntax dependency links rooted in predicates. Arguments are often also connected to predicates not directly but through simple and complex prepositions that consist of several words. Complex prepositions are detected using the predefined list of triplets <PREP, NOUN, syntactic link>. Name of a syntactic link is used to resolve ambiguity between noun phrases and complex prepositions.

We also take into account tokens that are not related to the predicate directly but are homogeneous to base arguments. The tokens that are linked to the base arguments with a conjunct relation ("conj") are considered as extensions of base arguments and are labeled with the same semantic role as the base argument. The list of predicates is also expanded via adding the syntactic subjects and agents connected to the extracted arguments with a nominal subject ("nsubj") relation, as well as with "name" and "appos" in case it is a person name or a title. The nominal modifier ("nmod") relation is used for nominal dependents and often indicates locations. The adverbial clause modifier ("advcl") helps to find the sequences of participle clauses that have a common subject.

3.2 Argument Classification

For argument classification, we train two feed-forward neural-network models: the model for "known" predicates and the model for "unknown" predicates. The feature set of the model for "known" predicates includes embeddings of argument and predicate lemmas, as well as the following sparse lexical and morpho-syntactic features:

- Various types of morphological characteristics of both an argument and a predicate (case, valency, verb form, etc.)
- Relative position of an argument in a sentence with respect to a predicate.
- Preposition of an argument extracted from a syntax tree (including a complex preposition as a single string).
- Name of a syntactic link that connects an argument token to its parent in the syntax tree.
- Argument and predicate lemmas.

We note that the predicate lemma is one of the essential features for high-quality semantic role labeling, since predicates express a situation in a sentence and determine its meaning slots. Similar morpho-syntactic structures with different predicates can express different meanings. Therefore, the lack of annotation for a predicate in a training set hits hard classifier confidence and overall performance on examples with this "unknown" predicate. We combat this problem by introducing a second model that is trained without predicate lemmas as features, so it should rely on predicate lemma word embeddings and other features instead. This model performs worse than the model for "known" predicates on seen predicates but it is also affected less by an absence of a predicate in a training corpus. This happens because predicate lemma embeddings capture predicate semantics, and similarity between these embeddings can help the model to guess meanings of "unknown" predicates.

In this work, we experiment with various types of word embeddings obtained from shallow models word2vec and FastText, as well as from pretrained language models ELMo and BERT. Recently, it has been shown that pretrained language models provide substantially better generalization to downstream models compared to shallow embeddings built by word2vec or FastText algorithms. This happens because language models can take into account contexts of words, for which they generate an embedding, and capture much longer dependencies in texts. ELMo generates contextual word representations by using a

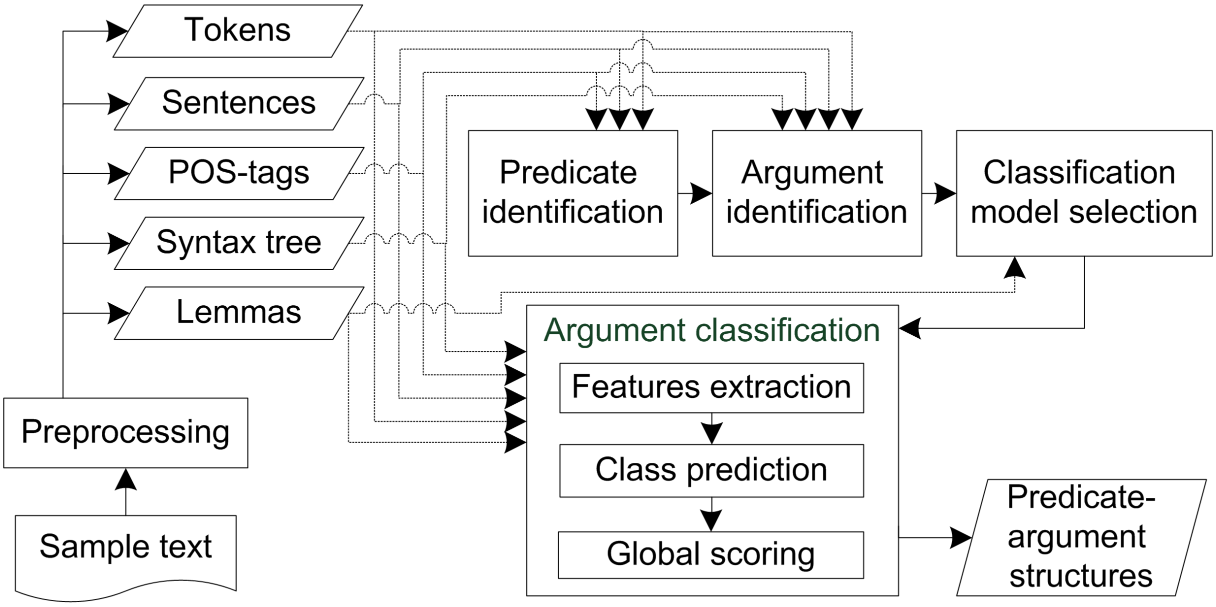


Figure 1: Semantic role labeling pipeline

stack of bidirectional LSTM layers that are trained to predict a following word on the basis of seen context. The output from each layer can be used as a set of features to downstream models. BERT is a masked language model that is trained to predict masked words in a sequence given all other words in the sequence. In addition, it is also trained to simultaneously predict whether two given sentences are consecutive. BERT uses self-attention encoder (Vaswani et al., 2017) that can be much faster than LSTM and can capture longer dependencies across sequences.

The feedforward neural network model for argument classification has three dense layers. Three inputs, namely embedding of a predicate, embedding of an argument, and sparse categorical features are separately passed through the first piecewise layer with ReLU activation. Concatenated outputs of the first layer are then propagated through another ReLU layer and the output layer with softmax activation. Before the activation function, batch normalization is applied on each hidden layer. The network is regularized with dropout. The output of this network is a vector of probabilities for each semantic role in a given inventory.

3.3 Global Scoring

Semantic labels in a single predicate-argument structure are not completely independent from each other. Moreover there is a certain linguistic constraint that requires that there should be no duplicate argument labels, since a meaning slot of a situation can be filled by just a single participant (for the core semantic roles) or a group of homogeneous arguments. In the argument classification step, we use a neural network model to produce a number of probabilities for a list of semantic roles and due to the linguistic constraint, we cannot greedily assign roles with maximum probability. In the global scoring step, we effectively produce the global optimal predicate-argument structure that fulfills the constraint by leveraging an integer linear programming inference (Punyakanok et al., 2004). Formally, it can be described in the following way. Let $x_{ij} \in \{0,1\}$ be a target variable and $x_{ij} = 1$ means an argument j has a semantic role i. Let $p(i,j)$ be a probability of assigning a role i to an argument j estimated by a neural network. Let n and m be the numbers of roles and arguments accordingly. The optimization problem formally:

$$\underset{x_{ij}}{\operatorname{argmax}} \sum_{j}^{m} \sum_{i}^{n} x_{ij} \log p(i,j)$$

$$\sum_{i}^{n} x_{ij} = 1, j = 1..m$$

$$\sum_{j}^{m} x_{ij} = 1, i = 1..n$$

$$x_{ij} \in \{0,1\}, i = 1..n, j = 1..m$$

Table 1: The merging scheme for mixed roles

Original (mixed) role	Destination role
agent – perceiver	perceiver
agent – sbj of mental state	sbj of mental state
result / target	result
location – patient	location
speaker – sbj of psychol. state	sbj of psychol. state

The optimal solution to this problem is used as the final assignment of semantic roles to arguments.

4 Experiments

4.1 Dataset and Preprocessing

FrameBank contains annotated text samples with multiple contextual sentences. Each sentence consists of tokens with their morphological features. We follow preprocessing procedure from (Shelmanov and Devyatkin, 2017) to map annotations to corresponding tokens. We also merged mixed roles annotations from the original dataset into their subsequent roles. See the merging scheme in Table 1.

Unlike (Shelmanov and Devyatkin, 2017), in this work, we do not rely on gold-standard linguistic annotations at all, since the goal of our work is to develop the parser that can process raw texts. To generate linguistic annotations for SRL input, we perform the following linguistic processing steps:

- Tokenization and sentence splitting are performed by NLTK library[3].
- Lemmatization, POS-tagging, and morphological analysis are done by MyStem (Segalovich, 2003).
- Syntax parsing is performed via UDPipe parser (Straka and Straková, 2017) with model trained on SynTagRus (Nivre et al., 2008).

These steps are implemented using the IsaNLP library[4].

The original corpus after preprocessing contains examples for 803 predicates. However, for many predicates there are just few examples, and some semantic roles are also rare. Therefore, we followed (Shelmanov and Devyatkin, 2017) and filtered the dataset keeping only predicates that have at least 10 examples. The filtered dataset retains 643 unique predicates (verbs). We also drop infrequent roles, for which the dataset contains less then 180 examples. The final corpus version contains 52,751 examples for 44 unique semantic roles.

4.2 Embeddings and Pretrained Language Models

In our experiments, we use the following publicly available pretrained word embeddings and language models:

- Word2Vec: RusVectores[5] (Kutuzov and Kuzmenko, 2017). Skip-gram model trained on Russian Wikipedia, dimension: 300.
- FastText: DeepPavlov[6] (Burtsev et al., 2018). Skip-gram model trained on a mixed corpus of Russian Wikipedia and Lenta.ru news texts, dimension: 300.
- ELMo: DeepPavlov. Pretrained on Russian Wikipedia corpus, achieves perplexity of 43.692, 2 layers, dimension: 1024.
- BERT multilingual cased: released by the authors (Devlin et al., 2018). Pretrained on 104 languages, 12 encoder blocks, produces vectors with 768 dimensions.
- RuBERT: DeepPavlov. RuBert is an adaptation of BERT-multilingual with vocabulary enriched with Russian byte-pairs (Arkhipov et al., 2019).

Both ELMo and FastText mitigate out-of-vocabulary problem, so we do not lose any predicates and arguments, while there are some misses for word2vec. BERT models are built upon byte-pair encoding, so we use only the first byte-pair representation for each token as recommended by the authors. It is worth noting that although BERT is a quite large model, it takes only 15 minutes on a single GTX 1080 Ti to encode all examples, compared to 1.5 hours for ELMo.

4.3 Neural Network Hyperparameters

We performed hyperparameter tuning using random search on the task of argument labeling for "known" predicates. The following parameters were selected and used in all further experiments:

[3] https://www.nltk.org/

[4] https://github.com/IINemo/isanlp

[5] http://rusvectores.org/en/

[6] https://deeppavlov.ai/

categorical features layer hidden size – 400, embeddings projection layer hidden size – 100, concatenated vector layer size – 400, dropout – 0.3.

4.4 Experimental Setup

We evaluated the argument extraction step using only the predicate-argument structures labeled in FrameBank and did not take into consideration any other structures in the corpus found by our parser.

For evaluation of argument labeling step, we conducted two experiments using various token representations and dataset splitting schemes.

In the first setup, we evaluate argument classification step on full set of features and test various word representations. Lexical, morphological, and syntax features are encoded in one-hot manner. Macro and micro F1 scores are calculated on a 5-fold cross-validation. Evaluation results are presented in Table 2.

In the second setup, we evaluate the performance of the argument classification models for "unknown" predicates. Thus, we divided the dataset in two parts by leaving 80% of predicates with their examples for training and 20% of predicates for testing. The sets of predicates in training and testing parts do not intersect. The division of predicates was performed at random. For this setup, there was no cross-validation. Instead, we averaged results of 5 models trained with different random seeds. In this experimental setup, we compare models for "unknown" predicates that do not take into account predicate lemma with various types of token embeddings.

To ensure the importance of introducing additional model for "unknown" predicates in our semantic role labeling pipeline and importance of predicate lemma feature, we also evaluate the model for "known" predicates on the test set with "unknown" predicates. The goal of this experiments is to show that the model for "known" predicates overfits on predicate-lemma features and performs worse than models trained specifically for "unknown" predicates, since it is "blind" to its most important feature. We report the results of the model for "known" predicates in this setup only with the embeddings that achieve the best score in the previous experiment. The test results are presented in Table 3.

4.5 Results and Discussion

In the argument extraction step, we achieve 74.48% precision, 85.12% recall, and 79.44% F1-score. We note that many false positives are due to the absence of non-core arguments in our evaluation set. These phrases that bear temporal, locative, and some other types of information are correctly identified by our parser but are considered as mistakes in the evaluation setup resulting in lower precision than it actually is. However, we see that it is the only adequate way to assess extraction quality with partially labeled data.

The results for the argument labeling step presented in Table 2 show that ELMo and BERT outperform all other approaches, including results in (Shelmanov and Devyatkin, 2017), although, unlike them, we do not rely on gold-standard morphological features. In Table 4, we also report the performance per semantic role of the model that uses ELMo word representations.

In many works, BERT outperforms ELMo by a significant margin. However, in our work, there is just an insignificant gap between ELMo and RuBERT. This is probably due to BPE tokenization scheme of BERT, since we take encoded representation only for the first subword unit of each token, with no fine-tuning, leaving a lot of information about words unused.

In the second experimental setup, the gap between RuBERT and ELMo is increased. In this case, the model based on RuBERT shows worse performance than all other approaches. However, there is a certain improvement $\Delta 1.5\%$ of micro F1 score between ELMo and word2Vec-based models. It shows that representations generated by deep pretrained language models can restore semantics of unseen predicates better than shallow models by additionally leveraging the context.

The results of the model for "known" predicates even with the best word representations ELMo are expectedly low. The performance drop compared to the model for "unknown" predicates with the same embeddings is substantial: 10% micro-F1 and more than 8% macro-F1. It is worse than any of other models except RuBERT. This proves that predicate lemma is very important as a feature and SRL pipeline should have two models to process "known" and "unknown" predicates to alleviate the domain shift.

Model	Micro F1	Macro F1
Plain Features Only	76.96 ± 0.67	73.63 ± 0.61
Word2Vec UPOS	79.87 ± 0.34	76.70 ± 0.77
FastText	80.60 ± 0.51	77.39 ± 0.36
ELMo	**83.42** ± 0.60	79.91 ± 0.40
BERT-Multiling	79.04 ± 0.63	75.68 ± 0.72
RuBERT	83.12 ± 0.60	**80.12** ± 0.62

Table 2: Performance of models in the experimental setup with “known” predicates

Model	Micro F1	Macro F1
ELMo (for known pred.)	45.51 ± 0.50	29.31 ± 0.82
Word2Vec UPOS	53.97 ± 0.21	37.29 ± 0.74
ELMo	**55.50** ± 0.51	**37.64** ± 0.41
FastText	49.37 ± 0.43	37.26 ± 0.29
BERT-Multiling	31.81 ± 0.51	21.04 ± 0.13
RuBERT	43.68 ± 0.50	30.84 ± 0.55

Table 3: Performance of models in the experimental setup with “unknown” predicates

5 Conclusion and Future Work

We presented and evaluated the first full pipeline for semantic role labeling of Russian texts. The experiments with various types of embeddings showed that the pretrained language models ELMo and BERT substantially outperform the embeddings obtained with shallow algorithms like word2vec and FastText. We also showed that providing supplementary SRL model for “unknown” predicates can alleviate the problem with annotation scarcity. We note, that in the case of predicting arguments for “uknown” predicates, the deep pretrained language model ELMo also outperformed other types of embeddings. We publish the code of the pipeline that can be used to parse raw Russian texts[7], we also publish the code for model training and experimental evaluation.

In the future work, we are going to apply semi-supervised and unsupervised algorithms to expand the training data and improve the model performance on out-of-domain data.

Acknowledgments

This paper is partially supported by Russian Foundation for Basic Research (project No. 16-29-12937, 17-07-01477).

Class	Precision	Recall	F-score
agent (11.7%)	76.1	83.3	79.5
patient (10.2%)	85.1	88.7	86.9
theme (6.9%)	84.6	71.6	77.6
sbj of psychol. state (6.2%)	86.7	83.9	85.2
goer (5.7%)	82.9	89.2	85.9
cause (4.7%)	86.2	88.6	87.4
speaker (4.5%)	73.5	78.3	75.8
location (4.1%)	87.4	82.5	84.9
content of action (3.6%)	89.1	83.8	86.3
content of thought (3.4%)	74.6	79.7	77.0
content of speech (3.4%)	75.9	69.5	72.6
final destination (3.4%)	70.3	52.0	59.8
result (2.8%)	63.5	54.0	58.4
patient of motion (2.6%)	88.8	80.4	84.4
stimulus (2.4%)	85.1	72.2	78.1
cognizer (2.3%)	85.1	76.9	80.8
addressee (1.8%)	75.7	79.1	77.4
perceiver (1.7%)	90.5	79.0	84.3
counteragent (1.6%)	56.8	65.6	60.9
effector (1.4%)	77.0	81.0	78.9
subject of social attitude (1.1%)	82.2	79.5	80.8
initial point (1.1%)	76.0	80.4	78.1
topic of speech (1.0%)	58.3	81.5	68.0
manner (1.0%)	84.0	69.3	76.0
recipient (1.0%)	82.3	68.0	74.5
goal (0.9%)	80.0	67.7	73.3
field (0.7%)	90.7	91.8	91.3
attribute (0.7%)	83.5	81.5	82.5
source of sound (0.7%)	73.7	69.5	71.6
behaver (0.6%)	84.8	84.4	84.6
situation in focus (0.6%)	88.2	88.3	88.2
counteragent of social attitude (0.6%)	75.0	58.2	65.5
sbj of physiol. reaction (0.6%)	76.0	85.4	80.4
topic of thought (0.6%)	95.9	88.7	92.2
potential patient (0.5%)	89.3	90.9	90.1
status (0.5%)	89.0	78.4	83.3
patient of social attitude (0.5%)	86.1	76.2	80.8
standard (0.5%)	80.2	85.3	82.7
term (0.5%)	87.5	85.7	86.6
attribute of action (0.5%)	92.5	71.2	80.4
causer (0.4%)	72.6	65.2	68.7
initial possessor (0.4%)	83.7	73.5	78.3
potential threat (0.4%)	73.6	82.7	77.9
path (0.3%)	90.3	80.0	84.9

Table 4: The performance of the model based on ELMo embeddings in the experimental setup with “known” predicates by semantic roles. The frequencies of roles in the training corpus are presented in parentheses

References

Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, and Alexey Sorokin. 2019. Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras

[7] `https://github.com/IINemo/isanlp_srl_framebank/tree/master`

Khakhulin, Yuri Kuratov, Denis Kuznetsov, et al. 2018. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122–127.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale qa-srl parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2051–2060.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 473–483.

Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2061–2071.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*, pages 1989–1993.

Andrey Kutuzov and Elizaveta Kuzmenko, 2017. *WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models*, pages 155–161. Springer.

Ilya Kuznetsov. 2015. Semantic role labeling for russian language based on russian framebank. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 333–338. Springer.

Ilya Kuznetsov. 2016. *Automatic semantic role labelling in Russian language, PhD thesis (in Russian)*. Ph.D. thesis, Higher School of Economics.

Olga Lyashevskaya and Egor Kashkin. 2015. Framebank: a database of russian lexical constructions. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 350–360.

Olga Lyashevskaya. 2012. Dictionary of valencies meets corpus annotation: a case of russian framebank. In *Proceedings of the 15th EURALEX International Congress*, volume 15.

Ana Marasović and Anette Frank. 2018. SRL4ORL: Improving opinion role labeling using multi-task learning with semantic role labeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 583–594.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Joakim Nivre, Igor M Boguslavsky, and Leonid L Iomdin. 2008. Parsing the syntagrus treebank of russian. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 641–648.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666.

GS Osipov, IV Smirnov, and IA Tikhomirov. 2010. Relational-situational method for text search and analysis and its applications. *Scientific and Technical Information Processing*, 37(6):432–437.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling 2004*, pages 1346–1352.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *IJCAI*, volume 5, pages 1117–1123.

Gozde Gul Sahin and Mark Steedman. 2018. Character-level models versus morphology in semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 386–396.

Ilya Segalovich. 2003. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *MLMTA*, pages 273–280. Citeseer.

A.O. Shelmanov and D.A. Devyatkin. 2017. Semantic role labeling with neural networks for texts in Russian. In *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue" (2017)*, 16, pages 245–256.

A. O. Shelmanov and I. V. Smirnov. 2014. Methods for semantic role labeling of Russian texts. In *Computational Linguistics and Intellectual Technologies. Papers from the Annual International Conference "Dialogue" (2014)*, 13, pages 607–620.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2245–2254.

Alexey Sokirko. 2001. A short description of dialing project.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

A Structural Approach to Enhancing WordNet with Conceptual Frame Semantics

Ivelina Stoyanova
Institute for Bulgarian Language
Bulgarian Academy of Sciences
iva@dcl.bas.bg

Svetlozara Leseva
Institute for Bulgarian Language
Bulgarian Academy of Sciences
zarka@dcl.bas.bg

Abstract

This paper outlines procedures for enhancing WordNet with conceptual information from FrameNet. The mapping of the two resources is non-trivial. We define a number of techniques for the validation of the consistency of the mapping and the extension of its coverage which make use of the structure of both resources and the systematic relations between synsets in WordNet and between frames in FrameNet, as well as between synsets and frames).

We present a case study on causativity, a relation which provides enhancement complementary to the one using hierarchical relations, by means of linking in a systematic way large parts of the lexicon. We show how consistency checks and denser relations may be implemented on the basis of this relation.

We, then, propose new frames based on causative–inchoative correspondences and in conclusion touch on the possibilities for defining new frames based on the types of specialisation that takes place from parent to child synset.

1 Introduction

The research presented in this paper aims at enhancing WordNet with information about the conceptual structure of verbs based on the mappings between WordNet (WN) and FrameNet (FN). This information includes description of the conceptual elements that receive expression as verbs' arguments and adjuncts as well as the selectional restrictions imposed on these elements.

Our approach relies on the structural features of each of the resources as well as on various kinds of linguistic analysis. The proposed enhancement is directed to: (a) improving the quality of existing mappings; (b) expanding the mappings' coverage; (c) enhancing the description of frames with additional information obtained from WN; (d) proposing structural improvements on the resources based on systemic features (e.g., causativity), including the definition of new conceptual frames; and (e) suggesting further procedures for verification and improvements of precision.

The described methodology is semi-automatic (automatic assignment complemented by manual verification at several stages), but its contribution consists in the devising of a set of procedures to a structured and consistent enrichment of the two resources, which presents an important step towards its automatisation. The enhanced resources will be made available to the research community. As the description of verbs' conceptual structure is largely language independent, the enriched description is applicable cross-linguistically.

2 Prerequisites and Motivation

2.1 Linguistic Resources

WN (Miller, 1995; Fellbaum, 1998) is a large lexical database that represents comprehensively the conceptual and lexical knowledge in the form of a network whose nodes denote cognitive synonyms (synsets) interconnected through a number of conceptual-semantic and lexical relations. The main relation that determines WN's structure (as reflected in the hierarchical treelike organisation of nouns and verbs) is a relation of inheritance of conceptual and lexical features between synsets. The respective pairs of synsets are linked through the relation of hypernymy/hyponymy.

FN (Baker et al., 1998; Baker, 2008) represents lexical and conceptual knowledge couched in the apparatus of frame semantics. Frames are conceptual structures describing particular types of objects, situations, or events along with their components, called frame elements, or FEs (Baker et al., 1998; Baker and Ruppenhofer, 2002; Ruppenhofer et al., 2016). Depending on their status, FEs may be core, peripheral or extra-thematic, cf. Ruppenhofer et al. (2016). For our purposes, we deal particularly with core FEs, which instantiate conceptually necessary components of a frame, and which in their particular configuration make a frame unique and different from other frames. To a lesser degree we touch upon peripheral FEs, which mark notions such as Time, Place, Manner, Means, among others and may be instantiated in any semantically appropriate frame. A Lexical Unit (LU) in FN is a pairing of a word with a meaning; each LU is associated with a frame describing its conceptual semantics.

2.2 Structural Properties of WN and FN

FN frames are related into a netlike structure through a number of frame-to-frame relations part of which also provide a hierarchical organisation. These relations are presented in detail in Leseva and Stoyanova (2019); below, we just sum up those used in the procedures we propose.

Inheritance (Is Inherited by ↔ Inherits from) is the strongest and most prominent relation in FN, which posits a relationship between a more general (parent) frame and a more specific (child) frame so that the child frame elaborates on the parent frame in such a way that each semantic fact about the parent must correspond to an equally or more specific fact about the child (Ruppenhofer et al., 2016, p. 81–82). By definition, *Inheritance* corresponds to the relation of hypernymy/hyponymy in WordNet. In an ideal setting, the hyponyms should be instantiations of the hypernym's frame or of a more specific frame that inherits from the hypernym's frame. For instance, the frame Execution *Inherits from* Killing and is assigned to the WN synset {*execute:1, put to death:1*} – a hyponym of {*kill:1*}, which is assigned the frame Killing.

As the two resources have been developed independently, their relational structure is distinct, one of the major differences being that there are other frame-to-frame relations that to various degrees embody the notion and features of inheritance; we use (some of) them in the definition of the procedures proposed in Section 4.1.

The first one is *Using (Is Used by ↔ Uses)*, a frame-to-frame relation defined as a relationship between two frames where the first one makes reference in a very general kind of way to the structure of a more abstract, schematic frame (Ruppenhofer et al., 2016, p. 83); it may be viewed as a kind of weak inheritance (Petruck, 2015) where only some of the FEs in the parent frame have a corresponding entity in the child frame, and if such exist, they are more specific (Petruck and de Melo, 2012). Thus, the frame Arranging *Uses* the frame Placing and the two frames share the FEs Agent and Theme, while the more specific FE Configuration in the frame Arranging corresponds to the more general FE Goal in the frame Placing; the first frame is exemplified by the synset {*arrange:1, set up:5*} which is a hyponym of {*put:1, set:1, place:1, pose:5*} whose assigned frame is the more general frame Placing.

We consider two more relations although they align with the notion of inheritance only marginally. *Perspective (Is Perspectivized in ↔ Perspective on)* is defined as a relation which indicates that a given situation viewed as neutral may be further specified by means of perspectivised frames representing different possible points-of-view on this neutral situation (Ruppenhofer et al., 2016, p. 82). *Subframe (Has Subframe(s) ↔ Subframe of)* is a relation between a complex frame referring to sequences of states and transitions (each of which can itself be separately described as a frame) and the frames denoting these states or transitions (Ruppenhofer et al., 2016, p. 83–84).

These hierarchical relations are the basis for part of the procedures outlined in Section 4.

2.3 Causativity in WN and FN

Causativity (Is Caused by ↔ Causative of) is a systematic non-inheritance relation where one of the frames represents the causative counterpart of the other, stative or inchoative, frame (Ruppenhofer et al., 2016, p. 85). *Causativity* corresponds straightforwardly to the WN relation *causes*, although this correspondence is exhibited in only a small number of cases (30 pairs) – the relation has not been implemented consistently neither in FN, nor in WN even in clear-cut parts of the lexicon such as those described by the hypernym trees with the roots {*change:1, alter:1, modify:3*} ('cause to change; make different; cause a transformation') and {*change:2*} ('undergo a change;

become different in essence; losing one's or its original nature').

3 Existing Mappings of WN and FN

Previous efforts at linking WN and FN include Shi and Mihalcea (2005), Baker and Fellbaum (2009), WordFrameNet[1] (Laparra and Rigau, 2009, 2010), MapNet[2] (Tonelli and Pighin, 2009), and more enhanced proposals, such as the system Semlink[3] (Palmer, 2009) which brings together WN, FN and VerbNet with PropBank, and its follow-up Semlink+ that brings in mapping to Ontonotes (Palmer et al., 2014). Some procedures for automatically extending the mapping, are presented by Leseva et al. (2018) and a more thorough overview may be found in Leseva and Stoyanova (2019).

Whereas these efforts have resulted in the creation of databases of integrated semantic knowledge, most of them deal with mapping of the units of the original resources to each other – FN LUs and WN synset members (literals), LU definitions and synset glosses, etc. Such a methodology is able to perform mapping in those cases where there is a correspondence between LUs and literals with equivalent or close meaning, but would fail where such correspondence is missing. With 155,287 synonyms in 117,659 synsets and more than 246,577 relations, of which 91,631 are instances of the hypernymy relation[4] as compared with 13,640 LUs and 1,875 frame-to-frame relations[5] in FN, the discrepancy in the size of the data is reflected in the limited coverage of the mappings between synsets and frames. To the best of our knowledge, no further checks and verification have been performed on the mappings, as well.

The approach that we propose in addition to the lexical mapping of units deals with exploring and taking into account the relational structure of the resources (especially the structure of WN), particularly the relation of inheritance which ensures the propagation of conceptual and linguistic features down the trees. We employ features of the relational structure in the definition of procedures for the augmentation of the mapping coverage which are aimed at: (i) discovering existing but unmapped relations between synset members and FN frames; and (ii) transferring frames between synsets through relations of inheritance derived from WN and FN.

4 Enhancing WN Mappings to FN

As noted above, the proposed approach combines the features used in the direct mapping with the structural properties of WN and FN – particularly, the inheritance relations existing between hypernyms and hyponyms in WN and the inheritance (and other similar relations) that determine the hierarchical structure of FN. As shown in Leseva and Stoyanova (2019), although the relations in the two resources have different number and scope, part of them are grounded in similar universal assumptions which leads to partial overlap, depending on their definition and the specificities of the information in the resources.

Apart from the correspondences between FN's *Inheritance* and other relations and the WN hypernymy relation, there are other systematic structural relations which can be applied for the purpose of enhancing the resources. Notable examples are the *Causativity* relation between frames in FN and the *causes* relation defined between causative and stative or inchoative verbs in WN (cf. Section 5).

4.1 Expanding Mappings based on Hierarchical Relations in WordNet

Our work relies on the assumption that in a taxonomic structure such as WN subordinate nodes inherit the properties of their superordinates, i.e. a hyponym elaborates on the meaning of its hypernym and shares its conceptual and linguistic properties. We propose that if a WN synset instantiates a particular FN frame, its hyponyms should (ideally) instantiate the same or a more specific frame which may or may not hold a(n) (inheritance) relation with the more general frame.

This assumption allows us to suggest that in the cases where we are not able to assign a FN frame due to the fact that the coverage of the two resources is non-overlapping and/or other mapping procedures fail, we may resort to assigning the frame of a hypernym to its hyponyms; at worst, the semantic representation will be too general.

There are 14,103 verb synsets in WN, which, unlike nouns that all have a common root, are organised in 566 separate trees. Initially, FN frames

[1] http://adimen.si.ehu.es/web/WordFrameNet

[2] https://hlt-nlp.fbk.eu/technologies/mapnet

[3] https://verbs.colorado.edu/semlink/

[4] https://wordnet.princeton.edu/documentation/wnstats7wn

[5] https://framenet.icsi.berkeley.edu/fndrupal/current_status

have been assigned to a total of 4,306 synsets out of which 264 are root synsets. In order to improve the quality of the existing mappings and to expand the coverage we performed the following steps: (i) manual verification of the frames assigned to root synsets (resulting in 75 corrected mappings); (ii) semi-automatic assignment of valid frames to selected root synsets with a large number of hyponyms (27 roots); (iii) assignment of a hypernym's frame to its hyponyms in the cases where a hyponym is not directly mapped to FN frames, thus obtaining an extended coverage of 13,226 verb synsets with an assigned FN frame; (iv) definition of further procedures to the end of improving the quality of this assignment (section 4.2).

4.2 Selection of Frames based on the FN and the WN Structure

We devised two types of procedures aimed at obtaining a more specific mapping: (i) procedures that make use of the conceptual and lexical information and the relational structure in FN; (ii) procedures employing the conceptual and lexical information and the relational structure in WN.

As noted above, the first step of assigning a FN frame to a WN synset is transferring the frame assigned to the synset's direct or inherited hypernym. The frame so assigned may either appropriately describe the conceptual structure of the literals in the synset, or it may provide a more general description than an optimally informative one. We therefore view this as a default assignment on the basis of which we try to elaborate to the end of discovering a more suitable or specific frame to which to map the synset. When such a frame is found, we validate it manually and assign it to the hyponyms of the synset under discussion, overriding the more general frame as in Example 1.

Example 1. **Synset**: eng-30-00047945-v {*dress:6; clothe:1; enclothe:1; garment:1*} 'provide with clothes or put clothes on'
Assigned FR from hypernym: Undergo_change
Suggested FR: Dressing (transferred automatically to 13 out of 15 hyponyms such as {*corset:1*} 'dress with a corset', {*vest:1*} 'dress with a vest', {*overdress:2*} 'dress too warmly')

Below, we describe the procedures proposed and how they make use of the relational structure of FN and WN and the following components of the description in the two resources, in particular: (i) WN literals (and synsets) and synset-to-synset relations – especially the hypernymy relation, as well as the relations between synsets with a common hypernym (i.e., sister synsets); and (ii) LUs from a particular FN frame (the verbs listed as instantiations of a given frame), the hierarchical frame-to-frame relations: *Inheritance*, *Uses*, *Subframe*, and *Perspective*, as well as the relation between two frames inheriting from the same frame (i.e., sister frames).

For a synset assigned a frame inherited from its hypernym, we apply the following procedures:

(1) **Literal–LU correspondence using FN relations**: We check whether any of the synset literals appears as a LU in: (a) the assigned frame (to confirm its validity); (b) more specific frames the frame under discussion is linked to by means of any of the considered frame-to-frame relations (to make it more precise); (c) the sister frames of the assigned frame.

Example 2. **Synset**: eng-30-00540946-v {*extend:8; expand:4*} 'expand the influence of'
Assigned FR from hypernym: Cause_change
Suggested FR from (1b): Change_event_duration (LU: extend)
Suggested FR from (1c): Cause_expansion (LU: expand)

(2) **Literal–LU correspondence using WN relations**: We check whether any of the synset literals appears as a LU in: (a) any of the frames assigned to its hyponyms; (b) any of the frames related to the frames in (a) through frame-to-frame relations; and (c) any of the frames assigned to its sister synsets.

Example 3. **Synset**: eng-30-00223374-v {*bolster:1; bolster up:1*} 'support and strengthen'
Assigned FR from hypernym: Cause_change
Suggested FR from (2c): Supporting (LU: bolster)

(3) **General literal–LU correspondence**: We check whether any of the synset literals appears as a LU in any other frame in FN.

Example 4. **Synset**: eng-30-00544936-v {*exalt:1*} 'raise in rank, character, or status'
Assigned FR from hypernym: Cause_change
Suggested FR from (3): Judgment (LU: exalt)

(4) **Keywords**: We use keywords (words contained in the FN frame name, plus their derivatives collected from WN through the *eng_derivative* relation), to identify synset literals and/or definitions containing these keywords as candidates to be as-

signed the frame in question.

Example 5. **Synset**: eng-30-00448864-v {*clean out:1; clear out:1*} 'empty completely'
Assigned FR from hypernym: Cause_change
Suggested FR from (4): Emptying (keyword *empty* found in gloss)

(5) **Direct similarity**: We check the similarity between the gloss of a verb synset and FN LU definitions (even when there is no correspondence between literals and LUs) to identify candidate frames for a given verb synset. We separate: (i) suggested frames related to the one assigned from the hypernym, which are given higher priority; (ii) unrelated suggestions.

Example 6. **Synset**: eng-30-02514187-v {*gloss over:1; skate over:1; smooth over:1; slur over:1; skimp over:1*} 'treat hurriedly or avoid dealing with properly'
Assigned FR from hypernym: Intentionally_act
Suggested FR from (5): Avoiding (which Inherits from Intentionally_act); similarity with the definition of LU *skirt.v* 'avoid dealing with'

(6) **Indirect similarity:** We check the similarity between the glosses of synsets derivationally related to the verb under discussion (as well as the glosses of their hypernyms, which are considered their closest semantic generalisation) and FN LU definitions to identify candidate frames for the verb synset. We separate: (i) suggested frames related to the one assigned from the hypernym, which are given higher priority; (ii) unrelated suggestions.

Example 7. **Synset**: eng-30-00831651-v {*warn:1*} 'notify of danger, potential harm, risk'
Assigned frame from hypernym: Telling
Derivationally related synset: eng-30-07224151-n {*warning:1*} 'a message informing of danger'
Suggested FR from (6): Warning (Inherits from Telling); similarity with the gloss of LU *alert.n* 'a message to inform someone of danger; a warning'

Similarity in procedures (5) and (6) is calculated as a cumulative measure based on coinciding terms in the two definitions. Scores of similarity between two words are highest for full match and lower when stemming is applied. Short words (up to length 3) are disregarded and longer words are given more weight. The final score is normalised by the length (in words) of the definitions.

Through these steps 9,341 new suggestions of more specific or other possible frames have been made for 5,661 synsets with automatically transferred hypernym frames – Table 1 shows the distribution of the new suggestions in terms of the types of procedures that have been applied and the distance of the synset from the hypernym whose frame has been inherited.

Procedure	# 1-step transfers	# 2-step transfers	# 3+ step transfers
(1)	516	231	121
(2)	460	41	17
(3)	1,701	859	145
(4)	1,088	612	27
(5)	1,175	526	202
(6)	1,009	417	194
Unique synsets	3,957	1,388	316

Table 1: Distribution of frames suggested for synsets with automatic frame assignments from the hypernym (rows (1)-(6) include multiple suggestions for the same synset)

4.3 Discussion on Evaluation

These suggestions need to be manually verified as so far no reliable fully automated verification procedure has been established. Since the main objective is to discover, or suggest, a more precise frame than the one assigned from the hypernym, which is not necessarily wrong but rather may be too general, such evaluation needs to measure the degree of relevance as opposed to precision. Furthermore, it will be highly dependent on the granularity of the frames and their hierarchical organisation. Designing such a measure and its automatisation, if at all achievable, is beyond the scope of this work.

Suggestions, although non-definitive, provide useful pointers to candidate frames and thus are valuable in assisting the manual selection of frames. Only in 203 cases are there multiple suggestions as a result of the procedures, out of which in 177 cases 5 or more different frames are suggested. There are 1,056 synsets for which a suggestion is confirmed at least 2 times from the repeated application of the same or different procedures, out of which 265 cases are confirmed 5 or more times.

Given the task, human judgment is indispensable, especially for frames assigned to synsets higher in the tree as errors propagate down and may result in multiple wrong assignments.

5 Causativity and Inchoativity as a Systemic Structural Feature

Another direction of expanding the mappings and verifying the information in both FN and WN is by employing systematic semantic relations such as causativity. It is a non-hierarchical relation that links stative (e.g. {*lie:2*} 'be lying, be prostrate; be in a horizontal position') or inchoative ({*lie down:1, lie:7*} 'assume a reclining position') verbs with their causative ({*lay:2, put down:2, repose:5*} 'put in a horizontal position') counterparts. The relation provides enhancement complementary to the one using hierarchical relations described above and links in a systematic way large parts of the lexicon.

A considerable part of causative and non-causative pairs are formed with the same root and are thus morphologically similar or identical, e.g. EN *change – change*; RO *schimba – schimba*; BG *promenyam – promenyam se*, which makes them easier to identify. Nevertheless, as noted above, causativity is not consistently encoded in WN, and neither is it fully implemented in FN where we have spotted a number of instances of inchoative/stative or causative frames lacking a counterpart in the opposite domain. This means that the verbs instantiating them cannot be appropriately described in FN. Respectively, the mapping of literals instantiating non-defined frames will result in failure of assignment or wrong assignment.

Causativity also has an important application in WN and FN data validation and expansion: exploring the assignment of frames from FN to synsets enables us to check the consistency of assignments, by adopting the following logic: (i) in a tree whose root is a causative synset, all the descendants must be assigned a causative frame; (ii) in a tree with an inchoative/stative root all the descendants must be inchoative/stative; (iii) the pairs of causative–non-causative synsets from corresponding trees should be connected to each other through the WN *causes* relation in a consistent way; (iv) the respective pair of causative–non-causative frames assigned to such a pair of synsets should also be related via the *Is Causative of* relation in FN. The opposite signals either wrong assignment of a frame or inconsistency either in the WN data, that is – the encoding of a stative or inchoative verb in a causative tree or vice versa, or in the FN data – missing or wrong relation between frames, undefined frames, etc.

	Causative change:1;	Non-causative change:2;
From FN	241	251
Direct hypernym	910	624
Indirect hypernym	577	469
Total	1,728	1,344
General frame*	719	561
in %	41.6%	41.7%

Table 2: Analysed data with respect to causativity (* assignments of the most general frame Cause_change for the causative and Undergo_change for the non-causative)

Sections 5.1 and 5.2 describe the procedures for exploring pairs of causative–non-causative trees and extracting information enabling the validation of assigned frames, as well as the increase of the density of causativity relations within FN and WN. Section 5.3 deals with the formulation of new causative or stative and/or inchoative frames.

5.1 Analysis and Consistency Checks

We have extracted two separate WN trees from two root synsets connected by the *causes* relation (see Table 2): (1) eng-30-00126264-v {*change:1; alter:1; modify:3*}, assigned the frame Cause_change; and its corresponding non-causative counterpart (2) eng-30-00109660-v {*change:2*}, assigned the frame Undergo_change.

The checks for consistency with regards to (i)-(iv) above, included the following procedures:

(1) Identifying non-causative synsets in the causative tree and causative synsets in the non-causative tree. These mismatches are identified by pattern matching of the gloss or by analysis aimed at establishing whether the manually assigned frame contradicts the position in the tree. 9 such cases have been found in the causative tree (e.g., eng-30-00416880-v {*even:6; even out:2*} 'become even or more even'). Pattern-matching in the non-causative tree proved to be unreliable. It identified 120 cases of 'make' or 'cause' in the gloss, but only a small number of them were causative synsets (e.g., eng-30-00330565-v {*break up:3; disperse:1; scatter:1*} 'cause to separate'). We propose moving each wrongly placed synset (and the subtree rooting from it) to the rel-

evant tree and attaching it to its real hypernym.

Furthermore, there are synsets which combine the causative and the non-causative meaning and thus, create inconsistency in the WN structure. We identify such synsets by pattern matching of the gloss since they usually have glosses such as 'make or become', 'cause or become', 'cause or undergo'. There are 7 cases in the causative (e.g., eng-30-01253468-v {*coarsen:2*} 'make or become coarse or coarser') and 5 in the non-causative tree (e.g., eng-30-00280532-v {*blacken:1; melanize:1*} 'make or become black'). We propose that such synsets are split into two and placed at the respective positions in the relevant trees. This is an optimal solution as these concepts are not necessarily expressed by the same lexeme cross-linguistically, and such a split improves the consistency of WN.

(2) Identifying non-causative frames assigned to synsets in the causative tree and causative frames assigned to synsets in the non-causative tree. A causative frame is identified based on: keywords such as 'cause' or 'make' in its name or its definition; Agent or Cause/Causer FEs in its conceptual structure; its position as the first member in a *Is Causative of* relation, etc. A non-causative frame is identified based on: keywords such as 'become' or 'undergo'; lack of Agent or Cause/Causer FEs in its structure; its position as the second member of an *Is Causative of* relation.

We found 7 non-causative frames in the causative tree (e.g., eng-30-02190188-v {*quieten:3; hush:5; quiet:9; quiesce:1; quiet down:1; pipe down:1*} 'become quiet or quieter', Frame: Becoming_silent) and 61 causative frames in the non-causative tree (e.g., eng-30-00339085-v {*crush:1*} 'break into small pieces', Frame: Cause_to_fragment). These are clearly either errors in the frame assignment or wrongly encoded synsets as discussed in (1).

(3) Identifying synset pairs connected by the *causes* relation in WN where the causative synset is assigned a non-causative frame or vice versa. Section 5.2 deals with the enrichment of the two resources with instances of the causative relation.

5.2 Densifying Causative Relations in WN and FN

The causative tree stemming from {*change:1*} and the non-causative one stemming from {*change:2*} were aligned using the WN *causes* relation, resulting in 47 pairs of corresponding synsets – one in each tree. A set of consistency checks showed that there are no crossing relations (i.e., no instances where for a causative hypernym C_1 and its hyponym C_2, and a non-causative hypernym N_1 and its hyponym N_2, C_1 causes N_2 and C_2 causes N_1).

Further procedures were proposed to discover pairs of corresponding causative and non-causative synsets unrelated through the causative relation. These are based on pattern matching of the definition and/or on measuring similarity, as well as on an analysis of the synsets position in the WN tree structure, the causative relations in which their sisters, hypernym and hyponyms enter, and the frames assigned to them. On the basis of these linguistic features we have identified 673 possible causative relations between pairs of synsets in the two corresponding trees. After manual validation they may be used to create a more dense structure of causative relations in WN, as well as to extend them to frame-to-frame relations in FN.

5.3 Suggesting New Frames

New frames are suggested where a suitable causative or non-causative frame is not defined in FN to match its existing counterpart. The missing one is defined using the conceptual description of the available frame. Consider the synset {*age:3*} 'make older': we assign it the frame *Cause_change* and then try to acquire additional classificatory information and, possibly, to find a more specific frame by applying the remaining procedures. We confirm that the synset's meaning is causative through the keyword procedure (cf. Section 4.2). Another mapping procedure suggests Aging as the corresponding frame. Aging is a non-causative frame denoting the meaning of an entity undergoing a particular kind of change (see Example 8). Since Aging does not have a causative counterpart in FN, we posit such a frame, Cause_to_age. The conceptual structure of stative/inchoative and causative counterparts is distinguished by the presence of a causative subevent in the latter (Van Valin Jr. and LaPolla, 1997, p. 109) which is associated with a causative (Agent or an Agent-like) participant (FE). Thus, in the discussed example Cause_to_age is derived from Aging by enriching the set of Aging's FEs with the frame elements Cause and Agent. In addition, we posit a *Causative_of* relation between Cause_to_age and Aging. In general, causative frames inherit from the abstract frame Transitive_action so we define an *Inheritance* relation between Transitive_action and Cause_to_age. In such a way the newly-defined relation is integrated into the FN relational structure.

Example 8. Frame: Cause_to_age
Core frame elements: Agent/Cause; Entity
FN definition: An Agent or Cause causes an Entity to undergo a change in age typically associated with some deterioration or change in state.
Example synsets: {*age:3*} 'make older'
FN relation: Inherits_from
Frame: Transitive_action
Core frame elements: Agent/Cause; Patient
Frame definition: An Agent or Cause affects a Patient.

FN relation: Is_Causative_of
Frame: Aging
Core frame elements: Entity
Frame definition: An Entity is undergoing a change in age typically associated with some deterioration or change in state.
Example synset: {*senesce:1, age:2, get on:7, mature:5, maturate:2*} 'grow old or older'

The domain of causativity provides an approach at symmetricising large parts of the lexicon both at a horizontal level (same level lexemes in a taxonomical hierarchy) and in depth as the improvements in the higher levels of the lexicon influence the deeper levels as reflected in the procedure of assigning relations by inheritance described in 4.1.

6 Frame Specialisation and Relations

The observations on hierarchical relations, especially on the more populated ones, such as *Inheritance*, *Using* and *See_also*, shed light on the specialisation that takes place from parent to child in the taxonomic (inheritance) hierarchy. The changes in the causativity domain deal with including/excluding FEs that correspond to causative subevents in the event structure. The modifications that occur in the conceptual and semantic structure include, but are not limited to the following:

– **Reducing the number of core frame elements by incorporating** one of them in the verb's meaning, e.g. {*whip:4*} incorporates the peripheral FE Instrument ('whip') of {*strike:1*} in the frame Cause_harm assigned to both;

– **Reducing the scope of the frame** through imposing more strict selectional restrictions on the FEs, e.g. {*drive:1*} (Operate_vehicle) as a hyponym of {*operate:3*} (Operating_a_system) applies only to land vehicles while other verbs in the frame impose different restrictions on the FE Vehicle;

– **Profiling a different FE** from the one profiled by the hypernym, e.g. {*rob:1*} (Robbery) profiles the Victim, while its hypernym {*steal:1; rip off:2, rip:4*} (Theft) profiles the stolen Goods;

– **Inclusion/exclusion of a causative/agentive FEs** corresponding to a causative subevent in the respective pairs of frames, e.g. {*break:5*} (Cause_to_fragment) and {*break:2, separate:10, fall apart:4, come apart:1*} (Breaking_apart).

Some of the types of specialisation are currently being studied as a point of departure for defining more narrow-scope frames that would allow for more precise predictions about the selectional restrictions and the syntactic realisation of FEs.

7 Future Work

A further goal is to enrich FN by extending its lexical coverage on the basis of the expanded mapping to synsets. Verbs which do not have correspondence among LUs (or no correspondence in a given frame) but belong to synsets that have been successfully mapped to FN frames, will be suggested as possible LUs to be included in the respective frame(s).

A venue of ongoing research is to define precise selectional restrictions on FEs and to implement them as semantic relations between a verb synset and a set of noun synonyms that satisfy these restrictions. In such a way we will enrich WN with relations between verbs and nouns corresponding to participants in their conceptual structure, particularly ones realised as arguments and adjuncts.

The developed resource may have a considerable impact on the development of methods for identification of predicate-argument structure in text, which in turn will facilitate the development of new methods for frame verification and consistency checks on FN and WN.

Acknowledgments

This study has been carried out as part of the project *Towards a Semantic Network Enriched with a Variety of Semantic Relations* funded by the National Science Fund of the Republic of Bulgaria under the Fundamental Scientific Research Programme (Grant Agreement 10/3 of 14.12.2016).

References

Collin F. Baker. 2008. FrameNet, present and future. In Jonathan Webster, Nancy Ide, and Alex Chengyu Fang, editors, *The First International Conference on Global Interoperability for Language Resources*. City University, City University, Hong Kong.

Collin F. Baker and Christiane Fellbaum. 2009. WordNet and FrameNet as Complementary Resources for Annotation. In *Proceedings of the Third Linguistic Annotation Workshop (ACL-IJCNLP '09), Association for Computational Linguistics, Stroudsburg, PA, USA*. pages 125–129.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL '98: Proceedings of the Conference. Montreal, Canada*. pages 86–90.

Collin F. Baker and Josef Ruppenhofer. 2002. FrameNet's frames vs. levin's verb classes. In J. Larson and M. Paster, editors, *Proceedings of 28th Annual Meeting of the Berkeley Linguistics Society*. pages 27–38.

Christiane Fellbaum. 1998. A Semantic Network of English Verbs . In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, MIT Press, Cambridge, MA, pages 69–104.

E. Laparra and G. Rigau. 2009. Integrating WordNet and FrameNet using a knowledge-based Word Sense Disambiguation algorithm. In *Proceedings of Recent Advances in Natural Language Processing (RANLP09), Borovets, Bulgaria*. pages 208–213.

Egoitz Laparra and German Rigau. 2010. eXtended WordFrameNet. In *Proceedings of LREC 2010*. pages 1214–1219.

Svetlozara Leseva and Ivelina Stoyanova. 2019. Enhancing Conceptual Description through Resource Linking and Exploration of Semantic Relation. In *Proceedings of 10th Global WordNet Conference, 23 – 27 July 2019, Wroclaw, Poland*. pages 229–238.

Svetlozara Leseva, Ivelina Stoyanova, and Maria Todorova. 2018. Classifying Verbs in WordNet by Harnessing Semantic Resources. In *Proceedings of CLIB 2018, Sofia, Bulgaria*. pages 115–125.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38(11):39–41.

Martha Palmer. 2009. Semlink: Linking PropBank, VerbNet and FrameNet. In *Proceedings of the Generative Lexicon Conference*. 9–15.

Martha Palmer, Claire Bonial, and Diana McCarthy. 2014. SemLink+: FrameNet, VerbNet and Event Ontologies. In *Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929–2014), Baltimore, Maryland USA, June 27, 2014*. Association for Computational Linguistics, pages 13–17.

Miriam R. Petruck. 2015. The Components of FrameNet. http://naacl.org/naacl-hlt-2015/tutorial-framenet-data/FNComponentsMRLP.pdf.

Miriam R. Petruck and Gerard de Melo. 2012. Precedes: A semantic relation in FrameNet. In *Proceedings of the Workshop on Language Resources for Public Security Applications*. pages 45–49.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. Petruck, Christopher R. Johnson, C. F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California.

Lei Shi and Rada Mihalcea. 2005. Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing. CICLing 2005. Lecture Notes in Computer Science*, Springe, Berlin, Heidelbergr, volume 3406.

Sara Tonelli and Daniele Pighin. 2009. New Features for Framenet – Wordnet Mapping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL'09), Boulder, USA*.

Robert D. Van Valin Jr. and Randy J. LaPolla. 1997. *Syntax: Structure, meaning and function* . Cambridge University Press.

Compositional Hyponymy with Positive Operators

Martha Lewis
ILLC, University of Amsterdam
m.a.f.lewis@uva.nl

Abstract

Language is used to describe concepts, and many of these concepts are hierarchical. Moreover, this hierarchy should be compatible with forming phrases and sentences. We use linear-algebraic methods that allow us to encode words as collections of vectors. The representations we use have an ordering, related to subspace inclusion, which we interpret as modelling hierarchical information. The word representations built can be understood within a compositional distributional semantic framework, providing methods for composing words to form phrase and sentence level representations. The resulting representations give competitive results on simple sentence-level entailment datasets.

1 Introduction

Distributional semantics (Harris, 1954; Firth, 1957) is effective and important within the area of computational modelling of language, particularly as regards to synonymy and paraphrasing. Within the field, at least two additional properties are desirable. Firstly, we would like a method by which we can compose vectors to form representations above the word level. Secondly, we would like a notion of lexical entailment, or hyponymy, with which we can capture a sense of the generality of concepts, and the notion of one concept being an instance of another. Furthermore, we would like these two properties to interact nicely with one another, so that the hyponymy relation is not lost when words are composed. In Bankova et al. (2019) the authors provide theory describing a notion of hyponymy that interacts well with compositionality, but do not provide experimental support. In Balkır et al. (2016) the authors suggest a measure of hyponymy based on entropy which also interacts well with compositionality and provide experimental support. A compositional version of the distributional inclusion hypothesis (DIH) (Geffet and Dagan, 2005) is examined in Kartsaklis and Sadrzadeh (2016). In the current paper, we use the framework of Bankova et al. (2019) to build positive operators that represent words. The operators are built using GloVe vectors (Pennington et al., 2014) and information about hyponym-hypernym relationships, which may be sourced either from human-curated resources like WordNet (Miller, 1995), or unsupervised sources, via the use of pattern-based methods. We give two new measures for graded hyponymy that provide a wider range of comparisons than the entropy-derived measure developed in Balkır et al. (2016) or the eigenvalue-related measure of Bankova et al. (2019).

We test our word representations and measures on a range of datasets. Three of the datasets are single-word entailment and have been designed to test directionality (BLESS) (Baroni and Lenci, 2011), detection (WBLESS) (Weeds et al., 2014), and both directionality and direction together (BIBLESS) (Kiela et al., 2015). We also test our models on the compositional dataset of Kartsaklis and Sadrzadeh (2016). This dataset provides a test for entailment at the phrase and sentence level. We find that our model performs fairly well on BLESS and its variants, and very well on the compositional dataset.

2 Background and Related Work

Vector space models of meaning often rely on some form of the distributional hypothesis: that words that occur in similar contexts have similar meanings. However, as well as deriving word meanings, we also need to give meanings to sentences and phrases. This means that we need some method for composing vector representations of words. Commonly-used methods include neural network methods, as seen in Socher et al. (2013); Bowman et al. (2014), simpler element-wise combination methods (Mitchell and Lapata, 2010), and tensor-based methods (Baroni and Zamparelli, 2010; Coecke et al., 2010; Paperno et al., 2014). Tensor-based methods operate by modelling words of different grammatical types in different vector spaces, and viewing relational words such as verbs and adjectives as linear maps that operate on their arguments. This allows methods from formal semantics to be more easily mapped onto vector space representations, and thereby gives us mechanisms for composing words, in line with their grammatical types, to form phrases and sentences.

We use an extension of the tensor-based approach, based on the methods given in Piedeleu et al. (2015); Bankova et al. (2019); Balkır et al. (2016). We represent nouns as *positive operators*, which can be considered as representing *collections of vectors*. Functional words like adjectives and verbs are now represented as *completely positive maps*, i.e. linear maps which preserve the positivity of their arguments. These can be thought of as linear maps which take valid collections of vectors to valid collections of vectors.

2.1 Related Work

Entailment is an important and thriving area of research within distributional semantics. The PASCAL Recognising Textual Entailment Challenge (Dagan et al., 2006) has attracted a large number of researchers in the area and generated a number of approaches. Previous lines of research on entailment for distributional semantics investigate the development of directed similarity measures which can characterize entailment (Weeds et al., 2004; Kotlerman et al., 2010; Lenci and Benotto, 2012). Geffet and Dagan (2005) introduce a pair of *distributional inclusion hypotheses*, where if a word v entails another word w, then all the typical features of the word v will also occur with the word w. Conversely, if all the typical features of v also occur with w, v is expected to entail w. Clarke (2009) defines a vector lattice for word vectors, and a notion of graded entailment with the properties of a conditional probability. Rimell (2014) explores the limitations of the distributional inclusion hypothesis by examining the properties of those features that are not shared between words. An interesting approach in Kiela et al. (2015) is to incorporate other modes of input into the representation of a word. Measures of entailment are based on the dispersion of a word representation, together with a similarity measure. All of these look at entailment at the word level. Related to the current work are the ideas in Balkır (2014); Balkır et al. (2016). In this work, the authors develop a graded form of entailment based on von Neumann entropy and with links to the distributional inclusion hypotheses developed by Geffet and Dagan (2005). The authors show how entailment at the word level carries through to entailment at the sentence level.

More recent approaches involve specialising word vectors for entailment Vulić and Mrkšić (2018), using non-Euclidean geometries Nickel and Kiela (2017); Nguyen et al. (2017); Le et al. (2019), and using pattern-based hyponymy extraction Roller et al. (2018); Le et al. (2019).

Most approaches, however, provide only word-word hyponymy. To test hyponymy in a compositional setting, we refer to the dataset of Kartsaklis and Sadrzadeh (2016) where a number of sentence and phrase-level hyponymy relationships are built from WordNet (Miller, 1995)

Another approach to detecting lexical entailment is via the identification of certain text patterns which indicate a hyponym-hypernym relationship. Examples are: *y such as x*, *x is a type of y*, which allow us to pick out pairs (x, y) which stand in the relation x is-a y. This approach was first outlined in Hearst (1992) and has been recently used in Roller et al. (2018) to build vectors able to encode the required hierarchical relationships.

In the current paper we provide methods for building word representations as positive operators, using hierarchical information either from human-curated sources such as WordNet, or unsupervised methods such as using Hearst patterns. We will show how these word representations can be composed, and how the hierarchical information percolates to the phrase level. Our contribution is to provide a means of building hierarchically ordered word representations, that can be composed into phrases and sentences. Previous work in this area has either concentrated on word-level hyponymy or phrase-level hyponymy. In this paper we combine the two in one framework.

3 Methods

We model words as collections of vectors, as follows. For a given vector $\vec{v} \in V$,[1] we can 'lift' this vector into the larger space $V \otimes V$, by taking the outer product of the vector with itself. We use the following notation:

$$\bar{v} := \vec{v}\vec{v}^\top \tag{1}$$

When $\vec{v}$ is a unit vector, the resulting matrix $\bar{v}$ is a projection operator. Multiplying another vector $\vec{x}$ by $\bar{v}$ projects $\vec{x}$ onto the one-dimensional subspace spanned by $\vec{v}$. A matrix of the form $\bar{v}$ can be thought of as a collection of just one vector, giving sharp, unambiguous information.

To represent collections of more than one vector, we sum together their matrix representations, resulting in another matrix:

$$\begin{aligned}\{\vec{v}, \vec{w}, \vec{x}\} &\mapsto \bar{v} + \bar{w} + \bar{x} \in V \otimes V \\ &= \vec{v}\vec{v}^\top + \vec{w}\vec{w}^\top + \vec{x}\vec{x}^\top \in V \otimes V\end{aligned}$$

Matrices M built in this way are called *positive operators* and have the following two properties:

- $\forall v \in V. \langle \vec{v}, M\vec{v} \rangle \geq 0$
- M is self-adjoint.

If we additionally impose that M has trace 1, then we can understand M as encoding a probability distribution over $\vec{v} \in V$ (Nielsen and Chuang, 2010). In the present work, we do not impose this condition, instead viewing M as representing a collection of vectors.

The eigenvectors and eigenvalues of M can be thought of as providing a summary of the information contained in M. A matrix of the form $\bar{v} = \vec{v}\vec{v}^\top$ will have one non-zero eigenvalue, corresponding to the normalized eigenvector $\vec{v}/||\vec{v}||$. When multiple vectors have been included in the collection, the matrix M will have more than one non-zero eigenvalue, and these will represent the weights for their corresponding eigenvectors.

We take a kind of extensional stance. We consider words to be modelled as collections of their instances. To model a noun, we can consider the collection of nouns that are hyponyms of that noun, and form the matrix representation corresponding to that collection.

Example 1 (Nouns)**.** Consider the noun *pet*, and suppose we have three types of pet: a pug, a goldfish, and a tabby cat. We give these values in a distributional space spanned by the basis vectors $\{\overrightarrow{\mathit{furry}}, \overrightarrow{\mathit{domestic}}, \overrightarrow{\mathit{working}}, \overrightarrow{\mathit{aquatic}}\}$ as follows:

	pug	*goldfish*	*tabby*
furry	3	0	5
domestic	4	5	5
working	0	0	0
aquatic	0	6	0

We form the representation of the noun *pet* by summing over the matrix representations of each vector:

$$\begin{aligned}[\![\mathit{pet}]\!] &= \overline{\mathit{pug}} + \overline{\mathit{goldfish}} + \overline{\mathit{tabby}} \\ &= \overrightarrow{\mathit{pug}}\,\overrightarrow{\mathit{pug}}^\top + \overrightarrow{\mathit{gfish}}\,\overrightarrow{\mathit{gfish}}^\top + \overrightarrow{\mathit{tabby}}\,\overrightarrow{\mathit{tabby}}^\top \\ &= \begin{pmatrix} 34 & 37 & 0 & 0 \\ 37 & 66 & 0 & 30 \\ 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 36 \end{pmatrix}\end{aligned}$$

Each of the matrices $\overline{\mathit{pug}}$, $\overline{\mathit{goldfish}}$, and $\overline{\mathit{tabby}}$ has just one non-zero eigenvalue, which is $||\vec{v}||$, and corresponds to the normalised eigenvector $\vec{v}/||\vec{v}||$, for $\vec{v} = \overrightarrow{\mathit{pug}}$, $\overrightarrow{\mathit{goldfish}}$, and $\overrightarrow{\mathit{tabby}}$ respectively.

The matrix $[\![\mathit{pet}]\!]$, however, has three non-zero eigenvalues of 100.52, 35.21, and 0.25, each corresponding to a combination of the basis vectors $\overrightarrow{\mathit{furry}}, \overrightarrow{\mathit{domestic}}, \overrightarrow{\mathit{aquatic}}$. The basis vector $\overrightarrow{\mathit{working}}$ has an eigenvalue of 0, indicating that $[\![\mathit{pet}]\!]$ is orthogonal to the vector $\overrightarrow{\mathit{working}}$.

3.1 Ordering Positive Operators

The set of positive operators on a vector space has an ordering introduced by Löwner (1934). For positive operators A and B, we define:

$$A \sqsubseteq B \iff B - A \text{ is positive}$$

In Bankova et al. (2019) the authors introduce a notion of graded hyponymy. The hyponymy relation may be true up to some error term, as follows. If $A \sqsubseteq B$, then $B - A = D$, where D is some positive operator. If this does not hold, it is possible to add in some error term E so that $A \sqsubseteq B + E$. This is viewed as saying that A entails B to the extent E. We wish to find the smallest such error term.

In Bankova et al. (2019), the error term was of the form $(1 - k)A$ and the scalar $k \in [0, 1]$ gave a graded notion of hyponymy. The effect of this

[1] Throughout the paper, we assume that vector spaces are $\mathbb{R}^n$

scalar is to reduce the size of A until it 'fits inside' B, giving a notion of graded hyponymy that says that A is a k-hyponym of B, $A \sqsubseteq_k B$ if $B - kA$ is positive.

One of the drawbacks of this measure is that if the space spanned by eigenvectors of A, called $Span(A)$, is not a subspace of $Span(B)$, then the value of k must be 0. We therefore consider two new measures, which we now describe. If $B - A$ is not positive, it is possible to make it positive by adding in a positive operator constructed in the following manner.

1. Firstly diagonalize $B - A$, resulting in a real-valued matrix, since $B - A$ is real symmetric.
2. Construct a matrix E by setting all positive entries of $B - A$ to 0 and changing the sign of all negative eigenvalues.

Then $B - A + E$ will give us a positive matrix. This E is our error term. The size of E is bounded above by the size of A, since certainly $B - A + A$ is positive. We propose two different measures related to this error term that give us a grading for hyponymy.

The first measure is

$$k_{BA} = \frac{\sum_i \lambda_i}{\sum_i |\lambda_i|} \tag{2}$$

where λ_i is the ith eigenvalue of $B - A$ and $|\cdot|$ indicates absolute value. This measures the proportions of positive and negative eigenvalues in the expression $B - A$. If all eigenvalues are negative, $k_{BA} = -1$, and if all are positive, $k_{BA} = 1$. This measure is symmetric in the sense that $k_{BA} = -k_{AB}$.

Secondly, we propose

$$k_E = 1 - \frac{||E||}{||A||} \tag{3}$$

where $||\cdot||$ denotes the Frobenius norm. This measures the size of the error term as a proportion of the size of A. Since $A = E$ in the worst case, this measure ranges from 0 when $E = A$ to 1 when $E = 0$.

Example 2 (Full Hyponymy)**.** Recall the example

$$[\![pet]\!] = \overline{pug} + \overline{goldfish} + \overline{tabby}$$

To determine whether a goldfish is a pet, we calculate:

$$[\![pet]\!] - \overline{goldfish} = \overline{pug} + \overline{tabby}$$

Now, since $\overline{pug}$ and $\overline{tabby}$ are both positive, and positivity is preserved under addition, we know that $[\![pet]\!] - \overline{goldfish}$ is also positive. Therefore, under either of our graded measures, the extent to which a goldfish is a pet is 1.

Example 3 (Graded Hyponymy)**.** Now suppose that we define $[\![dog]\!] = \overline{pug} + \overline{collie}$, with $\overrightarrow{pug}$ and $\overrightarrow{collie}$ defined as below:

	pug	*collie*
furry	3	3
domestic	4	2
working	0	2
aquatic	0	0

Then to determine whether a dog is a pet, we calculate:

$$\begin{aligned}
&[\![pet]\!] - [\![dog]\!] \\
&= \begin{pmatrix} 34 & 37 & 0 & 0 \\ 37 & 66 & 0 & 30 \\ 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 36 \end{pmatrix} - \begin{pmatrix} 18 & 18 & 6 & 0 \\ 18 & 20 & 4 & 0 \\ 6 & 4 & 4 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 16 & 19 & -6 & 0 \\ 19 & 46 & -4 & 30 \\ -6 & -4 & -4 & 0 \\ 0 & 30 & 0 & 36 \end{pmatrix}
\end{aligned}$$

The eigenvalues of $[\![pet]\!] - [\![dog]\!]$ are 75.38, 24.39, -5.77, 0, i.e., they are *not* all positive. This is because the subspace corresponding to $[\![dog]\!]$ is not a subspace of $[\![pet]\!]$, in particular because $[\![dog]\!]$ is not orthogonal to the basis vector $\overrightarrow{working}$.

However, much of $[\![dog]\!]$ is included in $[\![pet]\!]$. Using our graded measures given in equations (2) and (3), we can calculate that under k_{BA}, dog is a hyponym of pet to the extent 0.89 and under k_E, dog is a hyponym of pet to the extent 0.86.

3.2 Composing Positive Matrices

To compose positive matrices, we combine the methods outlined in Bankova et al. (2019) with the type-lifting methods outlined in Kartsaklis et al. (2012) to lift word representations into a higher-dimensional space. The impact of these methods are that we can define nouns and verbs within the same distributional space, and then lift the verb representations to a space that corresponds to a completely positive map.

We have choices about how to implement this type-lifting. One choice is composition, i.e. matrix multiplication, of the two operators. This latter operation results in a matrix that is no longer self-adjoint, and so Piedeleu (2014) suggests using the non-commutative and non-associative operator $M_2^{\frac{1}{2}} M_1 M_2^{\frac{1}{2}}$ in its place. This operator can be thought of as a kind of subspace projection, where M_1 is projected onto M_2. Piedeleu (2014) also notes that the pointwise multiplication of two positive operators is a completely positive map, giving us another choice for composition.

Following Kartsaklis et al. (2012), this gives us a method for building higher-level operators for verbs from lower-level operators. Firstly, we assume the noun space $N \otimes N$ to be equal to the sentence space $S \otimes S$, and refer to these both as $W \otimes W$. Given a representation of an intransitive verb $\llbracket \textit{verb} \rrbracket \in W \otimes W$, the effect of lifting the verb to a higher order space and then composing with a noun $\llbracket \textit{noun} \rrbracket \in W \otimes W$ is to apply the Frobenius multiplication to $\llbracket \textit{noun} \rrbracket \otimes \llbracket \textit{verb} \rrbracket$.

For intransitive verbs we can therefore combine the noun and the verb via three operations which we call **Mult**, **MMult1** (for matrix multiplication), and **MMult2**:

$$\text{Mult: } \llbracket n \textit{ verb} \rrbracket = \llbracket n \rrbracket \odot \llbracket \textit{verb} \rrbracket \tag{4}$$

$$\text{MMult1: } \llbracket n \textit{ verb} \rrbracket = \llbracket n \rrbracket^{\frac{1}{2}} \llbracket \textit{verb} \rrbracket \llbracket n \rrbracket^{\frac{1}{2}} \tag{5}$$

$$\text{MMult2: } \llbracket n \textit{ verb} \rrbracket = \llbracket \textit{verb} \rrbracket^{\frac{1}{2}} \llbracket n \rrbracket \llbracket \textit{verb} \rrbracket^{\frac{1}{2}} \tag{6}$$

For transitive verbs there is one possibility for pointwise multiplication of the operators, since this is both commutative and associative. For the second operation there are a number of composition orders. We will concentrate on two which reflect the difference between composing the verb with the object first and composing with the subject first. We therefore have:

$$\text{Mult: } \llbracket s\ v\ o \rrbracket = \llbracket s \rrbracket \odot \llbracket v \rrbracket \odot \llbracket o \rrbracket \tag{7}$$

$$\text{MMultO: } \llbracket s\ v\ o \rrbracket = \llbracket s \rrbracket^{\frac{1}{2}} \llbracket o \rrbracket^{\frac{1}{2}} \llbracket v \rrbracket \llbracket o \rrbracket^{\frac{1}{2}} \llbracket s \rrbracket^{\frac{1}{2}} \tag{8}$$

$$\text{MMultS: } \llbracket s\ v\ o \rrbracket = \llbracket o \rrbracket^{\frac{1}{2}} \llbracket s \rrbracket^{\frac{1}{2}} \llbracket v \rrbracket \llbracket s \rrbracket^{\frac{1}{2}} \llbracket o \rrbracket^{\frac{1}{2}} \tag{9}$$

4 Experimental Setting

We build representations of words as positive matrices, and selected from a number of alternative embeddings including GloVe vectors (Pennington et al., 2014), FastText (Bojanowski et al., 2017), and distributional vectors built from the concatenation of the UKWaC and Wacky corpora using PPMI and dimensionality reduction, all with 300 dimensions. To select the vector embeddings, we built word matrices as described above and tested them using the word vector evaluation package provided by Faruqui and Dyer (2014). We compared on the Rubinstein and Goodenough (Rubenstein and Goodenough, 1965), WordSim353, (Finkelstein et al., 2002), Miller and Charles (Miller and Charles, 1991), SimLex999 (Hill et al., 2015), SimVerb (Gerz et al., 2016), the Yang and Powers dataset (Yang and Powers, 2006) and the MEN dataset (Bruni et al., 2012). We did not inclue the rare word dataset or the similarity/relatedness splits of WS-353. Word matrices derived from GloVe vectors score best overall when using WordNet-derived hyponyms (table 1). This is also seen in the validation settings of the non-compositional datasets. When using Hearst-derived hyponyms, the generated word matrices perform poorly, although GloVe vectors still score most highly.

Table 1: Performance of word matrices derived from different word embeddings, using WordNet derived hyponyms. Bold highlights the highest value of each row.

	GloVe	Count	FastText
MC	**0.8441**	0.7124	0.8223
MEN	0.4836	0.2861	**0.5090**
RG	**0.8460**	0.6325	0.8387
SIMLEX-999	**0.4426**	0.2638	0.4272
SimVerb	**0.3458**	0.2158	0.3290
WS-353	0.3001	0.1677	**0.3033**
YP-130	**0.5619**	0.3465	0.5166

We compare our WordNet models to a symbolic model called Symb. For this model we mark two words w_1 and w_2 as being in the hyponym-hypernym relationship if w_1 is found in the transitive closure of the hyponyms of w_2.

4.1 Nouns

In order to build positive matrices for nouns, we use information about hypernymy relations. This information can be elicited using human built resources such as WordNet Miller (1995), or using Hearst patterns Hearst (1992). These are patterns like 'y such as x', which give markers for hyponym-hypernym pairs (x, y). To collect the hyponyms of a given word w from WordNet, we traverse the WordNet hierarchy and collect every word w_i in the transitive closure of the hyponymy relation.

For hyponyms generated by Hearst patterns, we use the publicly available dataset described in Roller et al. (2018), and refer the reader to that paper for details of its creation. The dataset consists of a set of word pairs $\mathcal{P} = \{(x_i, y_i)\}_i$ which are in a hyponym-hypernym relationship, together with the count $w(x, y)$ of the number of times that relationship has been seen in the text. As described in Le et al. (2019), the relationships thus extracted are both noisy and sparse, containing cycles and inconsistencies. As one example of this, the dataset contains the pair (*rome*, *european country*). To mitigate these phenomena, we apply a ppmi weighting to the counts. The weighting is as described in Roller et al. (2018), specifically

$$\text{ppmi}(x, y) = \max\left(0, \log\frac{p(x, y)}{p^-(x)p^+(y)}\right),$$

where, letting $W = \sum_{(x,y)\in\mathcal{P}}$, we have:

$$\begin{aligned} p(x, y) &= w(x, y)/W \\ p^-(x) &= \sum_{(x,y)\in\mathcal{P}} w(x, y)/W \\ p^+(x) &= \sum_{(y,x)\in\mathcal{P}} w(y, x)/W \end{aligned}$$

These equations give, respectively, the probability that the pair (x, y) is chosen from $\mathcal{P}$, the probability that x appears as a hyponym, and the probability that x appears as a hypernym. This sets the weighting of various unwanted pairs, such as the aforementioned (*rome*, *european country*), to 0.

To collect the set of hyponyms of a noun, we use only those hyponyms with a non-zero ppmi weighting, and take one transitive step. So the set of hyponyms of a given word x is the union of the sets $\{y_i|\text{ppmi}(x, y_i) > 0\}_i$ and $\{z_{ij}|\text{ppmi}(y_i, z_{ij}) > 0\}_{ij}$. We limit to one transitive step to again mitigate the noisiness of the dataset.

4.2 Verbs

WordNet contains verb hyponymy relationships, and therefore we can use similar methods to extract lists of hyponyms. However, we cannot use Hearst patterns to collate verb hyponymy relationships. As a proxy, we represent verbs as collections of their arguments. The intuition behind this is that of the extensional approach in formal semantics, mapped to distributional semantics in Grefenstette and Sadrzadeh (2011). We can think of both nouns and verbs as predicates, and consider the instances that the predicate applies to.

To collect the arguments of the verbs, we use the concatenation of the dependency parsed ukWaC and WaCky corpora (Ferraresi et al., 2008), and collect those arguments that appear at least 300 times in the corpus.

4.3 Building Matrices

Finally, having collected instances of nouns and verbs, we take the vectors $\overrightarrow{w}_i$ corresponding to each of these instances, take the outer product of each with itself, and add these together, i.e.:

$$[\![w]\!] = \sum_i \overrightarrow{w}_i \overrightarrow{w}_i^\top \in W \otimes W \qquad (10)$$

We have discarded weighting information. Words which have more instances are both more widely dispersed in terms of their eigenvalues, and also larger in terms of their matrix norm.

4.4 Datasets

We evaluate single word representations on the non-compositional BLESS hyponymy subset (Baroni and Lenci, 2011), WBLESS (Weeds et al., 2014), and BIBLESS (Kiela et al., 2015) datasets. We test our models using the hypernymy suite provided by Roller et al. (2018). The BLESS dataset requires the model to infer the direction of a hypernym pair. All pairs in the model are indeed in the hyponym-hypernym relationship, and the model must identify that this is the case. WBLESS consists of a set of pairs which may be in the hyponym-hypernym relationship, or unrelated. Each pair is assigned a value of 1 or 0 based on whether or not there is a hyponymy relationship. The software provided performs 1000 random iterations in which 2% of the data is used as a validation set to learn a classification threshold, and tests on the remainder of the data. Average accuracy across all iterations is reported. The BIBLESS dataset assigns values of 1, 0, and -1 based on whether the first word is a hyponym of the second, whether there is no relationship, or whether the second is a hyponym of the first. The software firstly tunes a threshold using 2% of the data, identifying whether a pair exhibits hypernymy in either direction. Secondly, the relative comparison of scores determines which direction is predicted. Again, the average accuracy over 1000 iterations is reported.

We further test on the compositional datasets from Kartsaklis and Sadrzadeh (2016). This is a series of three datasets, covering simple intransitive sentences, transitive sentences, and verb phrases. The intransitive verb dataset consists of paired sentences consisting of a subject and a verb. In half the cases the first sentence entails the second, and in the other half of cases, the order of the sentences is reversed. For example, we have:

summer finish, season end, T

season end, summer finish, F

The first sentence is marked as entailing, whereas the second is marked as not entailing. The dataset is created by selecting nouns and verbs from WordNet that stand in the correct relationship.

To test our models, we build the basic word representations as in equation (10). We then use the compositional methods outlined in section 3.2 to create the sentence representations. We calculate the graded entailment value between the composed sentence representations, and in results report area under ROC curve for comparison with previous literature. In particular, we compare with the best model from Kartsaklis and Sadrzadeh (2016), which uses a metric based on the distributional inclusion hypothesis, together with a tensor-based compositional model.

4.5 Significance Testing

To test significance of our results, we use bootstrapping Efron (1979) to calculate 100 values of the test statistic (either accuracy or AUC) drawn from the distribution implied by the data. We compare with figures from the literature using a one-sample t-test, and compare between models using a paired t-test. We apply the Bonferroni correction to compensate for multiple model comparisons.

5 Results

5.1 BLESS Variants

We present results on variants of the BLESS dataset in terms of accuracy, for comparison with other models, presented in table 2. Our best perfoming model is the WordNet based model with metric k_E. Althoough this model does not outperform the best supervised model (the differences in score are significant), the differences are fairly minimal (0.01 accuracy). Our methods (and those of others) outperform the symbolic baseline for the BLESS dataset. Our WordNet-based model does outperform the earlier model HyperVec with significance. Hearst-pattern based representations do not perform so strongly.

Table 2: Accuracy on the variants of the BLESS dataset. HyperVec figures are from Nguyen et al. (2017), Hearst from Roller et al. (2018), HypeCones from Le et al. (2019), LEAR from Vulić and Mrkšić (2018). Entries tagged with WN use WordNet.

Model	BLESS	WBLESS	BIBLESS
HyperVec - WN	0.92	0.87	0.81
Hearst	0.96	0.87	0.85
HypeCones	0.94	0.90	0.87
LEAR - WN	0.96	0.92	0.88
Symb - WN	0.91	0.93	0.91
k_{BA} - WN	0.95	0.88	0.84
k_E - WN	0.95	0.91	0.87
k_{BA} - Hearst	0.91	0.84	0.76
k_E - Hearst	0.91	0.86	0.80

Table 3: Area under ROC curve on the KS2016 datasets using k_E and WordNet derived hyponyms. For the SV and VO datasets, MMult1 refers to the model described in equation (5) and MMult2 refers to the model described in equation (6). For SVO, MMult1 refers to the model described in equation (8) and MMult2 refers to the model described in equation (9). $*$ indicates statistically significantly higher than the previous best performance Kartsaklis and Sadrzadeh (2016). $+$ indicates significantly higher than the additive baseline.

Model	SV	VO	SVO
KS2016 best	0.84	0.82	0.86
Verb only	0.870^{*}	0.944^{*}	0.908^{*}
Addition	0.941^{*}	0.948^{*}	0.972^{*}
Mult	0.975^{*+}	0.981^{*+}	0.978^{*}
MMult1	0.970^{*+}	0.971^{*+}	0.965^{*}
MMult2	0.967^{*+}	0.969^{*+}	0.971^{*}

Table 4: Area under ROC curve on the KS2016 datasets, using k_{BA} and WordNet derived hyponyms. Refer to Table 3 for explanations.

Model	SV	VO	SVO
KS2016 best	0.84	0.82	0.86
Verb only	0.902^{*}	0.967^{*}	0.931^{*}
Addition	0.970^{*}	0.964^{*}	0.978^{*}
Mult	0.974^{*}	0.984^{*+}	0.982^{*}
MMult1	0.987^{*+}	0.985^{*+}	0.995^{*+}
MMult2	0.987^{*+}	0.985^{*+}	0.995^{*+}

Table 5: Area under ROC curve on the KS2016 datasets, using k_E and Hearst-pattern derived hyponyms. Refer to Table 3 for explanations.

Model	SV	VO	SVO
KS2016 best	0.84	0.82	0.86
Verb only	0.714	0.808	0.716
Addition	0.877^{*}	0.807	0.912^{*}
Mult	0.887^{*}	0.808	0.864
MMult1	0.902^{*+}	0.824^{+}	0.883^{*}
MMult2	0.903^{*+}	0.800	0.877^{*}

Table 6: Area under ROC curve on the KS2016 datasets, using k_{BA} Hearst-pattern derived hyponyms. Refer to Table 3 for explanations.

Model	SV	VO	SVO
KS2016 best	0.84	0.82	0.86
Verb only	0.719	0.819	0.716
Addition	0.880^{*}	0.811	0.909^{*}
Mult	0.867^{*}	0.792	0.843
MMult1	0.909^{*+}	0.842^{*+}	0.930^{*+}
MMult2	0.904^{*+}	0.830^{*+}	0.924^{*+}

5.2 Compositional Datasets

On the KS2016 compositional datasets results are reported in terms of area under ROC curve. Our measures perform particularly well with WordNet derived hypernyms (Tables 3 and 4). This is likely to be due to the fact that both the dataset and our word representations were constructed from WordNet, and hence the high performance is to be expected. More interestingly, the word representations built using unsupervised methods also outperform previous best scores on this dataset, (tables 5 and 6), based on a form of the distributional inclusion hypothesis for tensor-based composition (Kartsaklis and Sadrzadeh, 2016), despite not performing so strongly on the single-word datasets.

6 Discussion and Further Work

We have suggested a mechanism for building the positive operators needed for the theory presented in Bankova et al. (2019), together with two novel measures of graded hyponymy. We tested these representations and measures on a number of well-known datasets, looking at similarity at the word level, hyponymy at the word level and one of which gives hyponymy at the phrase and sentence level. The representations and the measures we have developed perform competitively on these datasets. We have used both human-curated information and unsupervised methods to build the word representations. Unsurprisingly, human-curated information gives better performance.

A nice comparison is with the symbolic model. The fact that our WordNet-based models outperform this baseline shows that the models we propose can provide a ‘smoothed’ representation that allows hyponymy relationships to be inferred. For example, one of the hyponymy relationships not picked up in WordNet is *(oven, device)*. However, there are a number of other instances such as *(electric appliance, device)* that are similar enough to *oven* that *device* can be understood as including *oven*. What cannot be remedied, however, is where a term has no hyponyms in WordNet. For example, *herbivore* has no hyponyms in WordNet. This means that the WordNet-based representations have no way of forming a wide representation of *herbivore* thatincludes any of its instances.

As well as performing well on single-word hyponymy datasets, the representations we build sit within a compositional framework that allows us to form phrases and sentences and to reason about their entailment relationships. The WordNet-based representations behaved particularly well on this dataset, due to the fact that the dataset is built from WordNet. However, it is still an interesting set of results in that our graded measures interact well with the compositional methods we have proposed. Note that the measures we propose result in high baseline values to beat - i.e. for the verb-only and addition models. Again, this is likely due to the construction of the dataset. The dataset is formed from upwardly-monotone contexts, so computing entailment based only on the verb will still perform extremely well. Again, although this is due to the construction of the dataset, is is interesting to note that the measures and word representations we developed can model this structure so well. Furthermore the Hearst-pattern derived representations also outperformed previous work, indicating that these representations interact nicely with compositionality.

Similarities to our approach can be found in the notion of words being represented as Gaussians (Jameel and Schockaert, 2017; Vilnis and McCallum, 2014). The positive operators we build have the same structure as covariance matrices and, if appropriately normalized, are interpreted as representing a probability distribution over vectors. Representing words as Gaussians does not come with a given mechanism for composing words as we do. Exploring these connections is an area of further work.

Finally, a crucial extension to this whole approach is to be able to model hyponymy, composition, and their interaction in more contexts, for example using the natural logic introduced in Barwise and Cooper (1981).

Acknowledgements

Martha Lewis gratefully acknowledges funding from NWO Veni grant ‘Metaphorical Meanings for Artificial Agents’

References

Esma Balkır. 2014. Using density matrices in a compositional distributional model of meaning. Master's thesis, University of Oxford.

Esma Balkır, Mehrnoosh Sadrzadeh, and Bob Coecke. 2016. Distributional sentence entailment using density matrices. In *Topics in Theoretical Computer Science*, pages 1–22. Springer.

Dea Bankova, Bob Coecke, Martha Lewis, and Dan Marsden. 2019. Graded hyponymy for compositional distributional semantics. *Journal of Language Modelling*, 6(2):225–260.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. In *Philosophy, language, and artificial intelligence*, pages 241–301. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks can learn logical semantics. *arXiv preprint arXiv:1406.1827*.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 112–119. ACL.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *arXiv:1003.4394*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

B. Efron. 1979. Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7(1):1–26.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proc. of ACL: System Demonstrations*.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on ACL*, pages 107–114. ACL.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *EMNLP*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1394–1404, Stroudsburg, PA, USA. ACL.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Shoaib Jameel and Steven Schockaert. 2017. Modeling context words as regions: An ordinal regression approach to word embedding. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 123–133.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2016. Distributional inclusion hypothesis for tensor-based composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2849–2860.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *In Proceedings of COLING: Posters*, pages 549–558.

Douwe Kiela, Laura Rimell, Ivan Vulic, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the ACL*. ACL.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389.

Matt Le, Stephen Roller, Laetitia Papaxanthos, Douwe Kiela, and Maximilian Nickel. 2019. Inferring concept hierarchies from text corpora via hyperbolic embeddings. *arXiv preprint arXiv:1902.00913*.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 75–79. ACL.

Karl Löwner. 1934. Über monotone Matrixfunktionen. *Mathematische Zeitschrift*, 38(1):177–216.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 233–243.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

Michael A Nielsen and Isaac L Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.

Denis Paperno, Marco Baroni, et al. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 90–99.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Robin Piedeleu. 2014. Ambiguity in categorical models of meaning. Master's thesis, University of Oxford.

Robin Piedeleu, Dimitri Kartsaklis, Bob Coecke, and Mehrnoosh Sadrzadeh. 2015. Open system categorical quantum semantics in natural language processing. *arXiv:1502.00831*.

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 511–519.

Stephen Roller, Douwe Kiela, and Maximilian Nickel. 2018. Hearst patterns revisited: Automatic hypernym detection from large text corpora. *arXiv preprint arXiv:1806.03191*.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*.

Ivan Vulić and Nikola Mrkšić. 2018. Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1134–1145.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259. Dublin City University and Association for Computational Linguistics.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1015. ACL.

Dongqiang Yang and David M. W. Powers. 2006. Verb similarity on the taxonomy of wordnet. In *In the 3rd International WordNet Conference (GWC-06), Jeju Island, Korea*.

The Impact of Semantic Linguistic Features in Relation Extraction: A Logical Relational Learning Approach

Rinaldo Lima[1], Bernard Espinasse[2], Fred Freitas[3]
[1] Departamento de Computação, Universidade Federal Rural de Pernambuco, Brazil
[2] LIS UMR CNRS 7020, Aix-Marseille Université/Université de Toulon, France
[3] Centro de Informática, Universidade Federal de Pernambuco, Brazil
rinaldo.jose@ufrpe.br, bernard.espinasse@lis-lab.fr, fred@cin.ufpe.br

Abstract

Relation Extraction (RE) consists in detecting and classifying semantic relations between entities in a sentence. The vast majority of the state-of-the-art RE systems relies on morphosyntactic features and supervised machine learning algorithms. This paper tries to answer important questions concerning both the impact of semantic-based features, and the integration of external linguistic knowledge resources on RE performance. For that, a RE system based on a logical and relational learning algorithm was used and evaluated on three reference datasets from two distinct domains. The yielded results confirm that the classifiers induced using the proposed richer feature set outperformed the classifiers built with morphosyntactic features in average 4% (F1-measure).

1 Introduction

Relation Extraction (RE) consists in detecting and classifying binary semantic relations between entities in a sentence.

Many RE systems use statistical machine learning techniques, such as feature-based and tree kernels-based methods (Choi et al., 2013) are based on a propositional hypothesis space for representing examples, i.e., they employ an attribute-value representation achieving robust results. However, this representation is not able to effectively capture structural information from parse trees without loss of information (Choi et al., 2013). More recently, other RE systems using deep learning techniques, such as Convolution Neural Networks (Nguyen and Grishman, 2015) and Recurrent Neural Networks (Miwa and Bansal, 2016) have also been proposed. They are based on a dense vector representation of the input words retrieved from word embeddings (Mikolov et al., 2013).

Some other RE systems rely on natural language processing (NLP) techniques for extracting relevant features from the text. They typically integrate shallow NLP tools for coping with lexical and syntactic aspects of the texts such as POS tagging, lemmatization, chunking, and syntactic parsing (Choi et al., 2013). However, according to Zouaq (2011), there are two main reasons to seriously consider deeper linguistic processing for RE: (i) it may provide deeper semantic meaning; (ii) NLP tools are becoming sufficiently robust to be considered as reliable tools for knowledge and model extraction.

Contrarily to related work above, this work subscribes to the idea of performing RE employing the Logical and Relational Learning (LRL) (de Raedt, 2008) approach that can generate classification models from complex data structures such as graphs or multiple tables. More precisely, we rely on Inductive Logic Programming (ILP) (Muggleton, 1991) as one of the most successful relational learning techniques because it employs a symbolic and declarative representation for the examples and the extraction models are both understandable and interpretable by humans. Moreover, ILP allows for many forms of prior knowledge, including semantic resources, to be integrated into the induction of the extraction rules (de Raedt, 2018).

In this paper, we try to answer experimental questions concerning not only the impact of semantic linguistic features but also the integration of external knowledge resources on RE. For that, an ILP-based RE system was employed and evaluated on three datasets from two distinct domains.

The main contribution of this work consists in the experimental validation of our working hypothesis that a feature engineering step comprising a sub-

Proceedings of Recent Advances in Natural Language Processing, pages 648–654,
Varna, Bulgaria, Sep 2–4 2019.

stantial body of deep linguistic knowledge, in combination with an expressive inductive learning technique, can generate effective RE models.

2 Relational Learning and Inductive Logic Programming

Relational Learning (RL) concerns the learning task from complex, heterogeneous examples represented by multirelational datasets. RL enables the development of applications in many fields including bio-informatics, networks analysis, and drug design (de Raedt, 2008). LRL (DeRead, 2008) combines machine learning and logic-based formalisms to automatically induce first-order rules from multi-relational examples.

ILP (Muggleton, 1991) is one of the most successful LRL-based technique that can not only induce symbolic rules from examples represented as multi-relational data, but also integrate background knowledge (BK) represented as logical clauses in first-order logic (FOL). Unlike traditional machine learning methods, classification models (rules) in ILP are both understandable and interpretable by humans. Most of the current ILP systems induce a set of Horn-clauses and employ Prolog as their core inference engine (Muggleton, 1991).

3 Logical Relational Learning System for Relation Extraction

The main contribution of this paper consists in the experimental validation of our working hypothesis that a feature engineering step composed by a substantial body of deep linguistic knowledge in combination with an expressive inductive learning technique can generate effective RE models. For testing this hypothesis, a LRL RE system (Lima et *al.*, 2017; Lima et *al.*, 2019) was used. The remainder of this section describes the RE system, the rich feature engineering component, and the underlying model for representing semantic features.

3.1 System Architecture

Fig. 1 shows the functional architecture our LRL system for RE. Its major components are highlighted as darker boxes and briefly presented next.

Deep NLP Component. This component performs the automatic annotation of the input documents in English. Its output is formed by XML files containing several layers of linguistics annotations. Its distinguishing characteristic consists in various NLP analysis it performs, starting from tokenization, passing for shallow analysis, and finishing with more advanced semantic analysis.

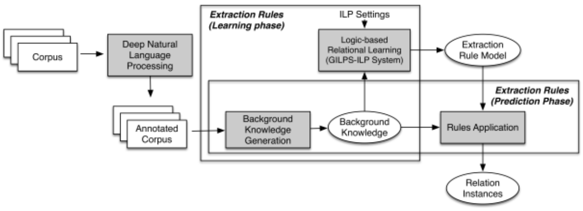

Corpus
Deep Natural Language Processing
Annotated Corpus
Extraction Rules (Learning phase)
ILP Settings
Logic-based Relational Learning (GILPS-ILP System)
Extraction Rule Model
Extraction Rules (Prediction Phase)
Background Knowledge Generation
Background Knowledge
Rules Application
Relation Instances

Figure 1. RE System architecture.

Background Knowledge Generation component. This component automatically generates and represents relevant features from an annotated set of documents. The generated features are converted into a knowledge base implemented as a Prolog factual base.

ILP rule learning component relies on an LRL system rooted on ILP that induces Horn-like extraction rules from training data. The extraction rules follow the same syntax of a Prolog predicate. This learning component is based on GILPS, a general ILP system proposed by Santos (2010).

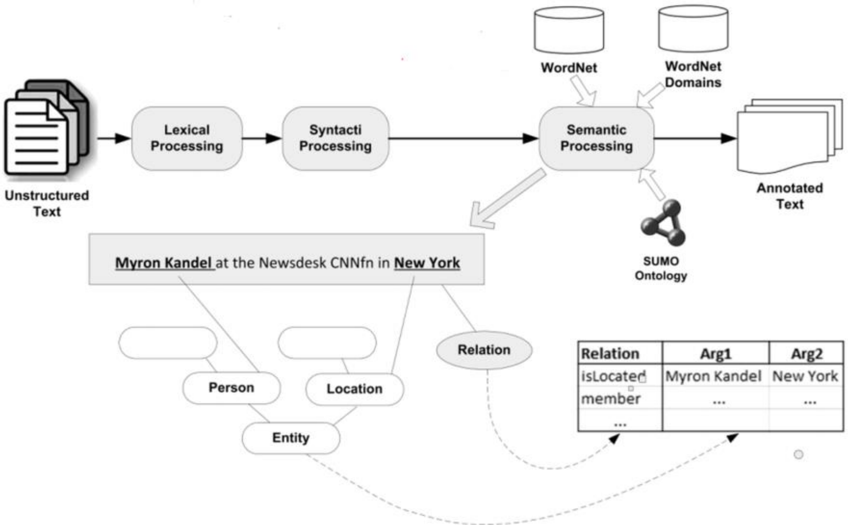
Rule application component. It applies the induced rules on the Prolog factual base generated from new documents not seen in the rule model learning phase. As a result, new instances of relations are identified and extracted.

Due to their importance, the above first two components will be detailed next.

3.2 Feature Engineering via Deep NLP

NLP technologies are of paramount importance in RE, since they can analyze unstructured texts and extracting their meaning. Indeed, the result of NLP analyses (or *annotation process*) is a richer version of the input text with further lexical, syntactical, and semantic metadata seem as a normalized representation of texts.

Due to its inherent complexity, NLP is not carried out in a single large stage. Instead, the annotation process is carried out as a chain of processes in which the output of a previous process becomes the input to the next one. Accordingly, the NLP component in our RE architecture is composed of the three major components as depicted in Fig. 2. The first analysis provides the basic morphological elements and lemmas, i.e., canonical base form of the words. In addition, categorical information is attached to each lexical item as Part-of-Speech (POS) tags (e.g., noun, verb, etc.). This facilitates the posterior task of determining groups of words (chunking analysis) that grammatically belong to the same category. Next, the syntactical analysis (syntactic parsing) identifies the structural relationships holding between words at the sentence level. The final semantic analysis links words to lexical semantic resources, including WordNet (Fellbaum, 1998), SUMO ontology (Niles and Pease, 2003), and WordNet Domain hierarchy (Bentivoli et al., 2004). Such semantic resources offer a variety of semantic relations including synonyms and hyponyms from WordNet, and additional semantic relations between verbs and their arguments considered here as predicates.



Relation	Arg1	Arg2
isLocated	Myron Kandel	New York
member	...	...
...		

Figure 2. Overview of the Deep NLP pipeline.

The Deep NLP component relies on the Stanford CoreNLP[1] for carrying out the following pipeline: tokenization, sentence splitting, POS tagging, lemmatization, NER, and dependency parsing. Chunking analysis is performed by Apache OpenNLP[2], while morphological analysis, gazetteers look-up, and pronoun normalization were implemented as ad hoc programs. That is followed by the Sense Learner (Mihalcea and Faruque, 2004) that disambiguates noun and verbs. Then, using the Java WordNet library[3], the *sense_id* of nouns and verbs are found in WordNet, along with of all synonyms and hyponyms of a given word. In addition, Lin´s similar words dataset (Lin et al., 2003) is used for retrieving a list of N (N = 5) most similar words to a given word. Semantic Role Labeling (SRL) is performed on all verbs by ClearNLP[4]. Still, for verbs, the SuperSense Tagger[5] (Ciaramita and Altun, 2006) finds their selectional preferences. It annotates text with 41 broad semantic categories (WordNet supersenses) for both nouns and verbs. Next, the mapping between all WordNet synsets and the SUMO ontology[6] is exploited for retrieving the related class from the SUMO ontology. Finally, an ad-hoc program maps words to labels from the WordNet Domains **(**Bentivogli *et al.*, 2004). Table 1 summarizes the entire pipeline.

3.3 Relational Representation of Sentences

The goal of the BK Generation Component is to extract and represent features according to a relational model consisting of documents, sentences, phrases, and tokens. They are converted into a Prolog factual base and used as input by the ILP-based learner. There are the four main groups of features:

Lexical features which concern word, lemma, length, and general morphological type information at the token level.

Syntactic features denote word POS tags; head word of nominal, prepositional or verbal chunk; bi-grams and tri-grams of consecutive POS tags of words, chunking features that segment sentences into a noun, prepositional, and verb phrases, chunk head word, and its relative position to the main verb of the sentence.

Semantic features include named entities, entity mentions provided by the corpus as well as all lexical-based semantic features (sense/hypersense,

[1] https://stanfordnlp.github.io/CoreNLP

[2] http://opennlp.apache.org

[3] https://sourceforge.net/projects/jwordnet

[4] https://github.com/clearnlp/clearnlp

[5] https://sourceforge.net/projects/supersensetag

[6] http://www.adampease.org/OP

synonyms, domain sense), and semantic roles of the verbs with their arguments.

Structural features consist of the structural elements connecting all the above features according to our relational model. They denote (i) the sequencing of tokens preserving the token order in the input sentence; (ii) the part-whole relation between tokens and the chunk containing them; (iii) the sequencing of chunks is represented by edges between their head tokens; and (iv) the grammatical dependency between two tokens in a sentence according to the typed dependencies between words.

The relational representation of all the above types of features is straightforward: a unary predicate in Prolog denotes identifiers, e.g., *token(id),* while binary predicates correspond to attribute–value pairs and relations, e.g., *rel(arg1, arg2).*

NLP Subtask	Tool or Resource
Tokenization	*Stanford CoreNLP*
Sentence Splitter	
POS	
Lemmatization	
Chunking	*OpenNLP Chunker*
NER	*Stanford CoreNLP*
Morphological Analysis	*ad hoc programs*
Gazetteer Look-up	
Pronoun Normalization	
Syntactic Parsing - Dependency	*Stanford CoreNLP*
Worde Sense Disambiguation	*Sense Learner*
WordNet Synsets (synonyms and hypernyms)	*WordNet 3.0*
Similar words	*Lin´s database*
SRL with Propbank/VerbNet	*ClearNLP*
Selectional Preferences	*SuperSense Tagger*
Semantic mapping to Domains	*WordNet domains*
Semantic mapping to SUMO	*Ad hoc program*

Table 1. Complete pipeline of the Deep NLP tools component.

4 Experiments

This section reports the results of experiments performed on three benchmark datasets from two distinct domains (newswire and biomedical).

4.1 Experimental Questions

We investigate the effectiveness of the proposed semantic linguistic features used in the induction of the relation extraction rules by the our ILP system. More precisely, we want to answer the following experimental questions (EQ):

EQ1. *Do the features present a complementary contribution to the performance results?*

EQ2. *What is the impact of the semantic linguistic features on the final induced set of extraction rules?*

EQ3. *How well do the rules generalize among different datasets: either in the same domain or in distinct domains?*

4.2 Dataset and Evaluation Measures

Three publicly available RE datasets from the newswire and the biomedical domain containing binary relations were selected for analysis:

reACE 2004/2005. The reACE 2004/2005 datasets introduced by Hachey et al. (2011) are the result of several transformation steps (refactoring, preprocessing, and reannotation) for normalizing the two original ACE datasets so that they adhere to a common notion of relation that is more intuitive and simpler: relation instance denotes a predicate over two arguments, where the arguments represent concepts in the real world.

Table 2 shows the distributions of the relation types in reACE 2005 dataset, whereas Table 3 shows some examples of them.

reACE 2005 - Relation Types	Freq
Employment	228
Membership	36
Located	280
Citizen-Resident-Religion-Ethnic	39
Business	16
Family	42
Geographical	119
Subsidiary	47

Table 2: reACE 2005 relation types.

Relation type	Example phrases
business (John, superiors)	John´s superiors…
Employ (Investors, “Wall Street”)	Investors on Wall Street…
citizen (voters, Missouri)	Some Missouri voters…

Table 3. Examples of reACE 2005 relations.

IEPA. The Interaction Extraction Performance Assessment (IEPA) corpus (Ding et al.,2002) is a biomedical dataset comprising 303 abstracts retrieved by ten queries suggested by domain experts to the PUBMED repository. An interaction between two terms, i.e., a specific pair of co-occurring chemicals in the IEPA corpus, was defined as a direct or indirect influence of one on the quantity or activity of the other (Ding et al., 2002). Examples of interactions between terms A and B are *"A increased B"*, and "*A activated C, and C activated B*".

Evaluation Measures. The classical IR measures of Precision *P*, Recall *R*, and F1-measure (Baeza-Yates and Ribeiro-Neto, 1999) were used for measuring the effectiveness (impact) of the proposed enhanced features on the RE task.

4.3 Experimental Protocol

We employed five-fold cross-validation which allows both the maximal use of the available training data on all the datasets used in the experiments. Moreover, preliminary experiments were performed for determining the optimal learning parameters according to the criteria of achieving high accuracy and preventing model overfitting. The best parameter setting for the ILP-based learning component found were: *evalfn = coverage*, *i (depth)* = 3, *minpos = 5*, and *noise = 0.2*.

4.4 Results

Table 4 summarizes the results of using several combinations of features on reACE 2004/2005 and IEPA datasets, while Tab. 5 conveniently displays the difference in performance between each pair of corresponding lines indexed by the column id.

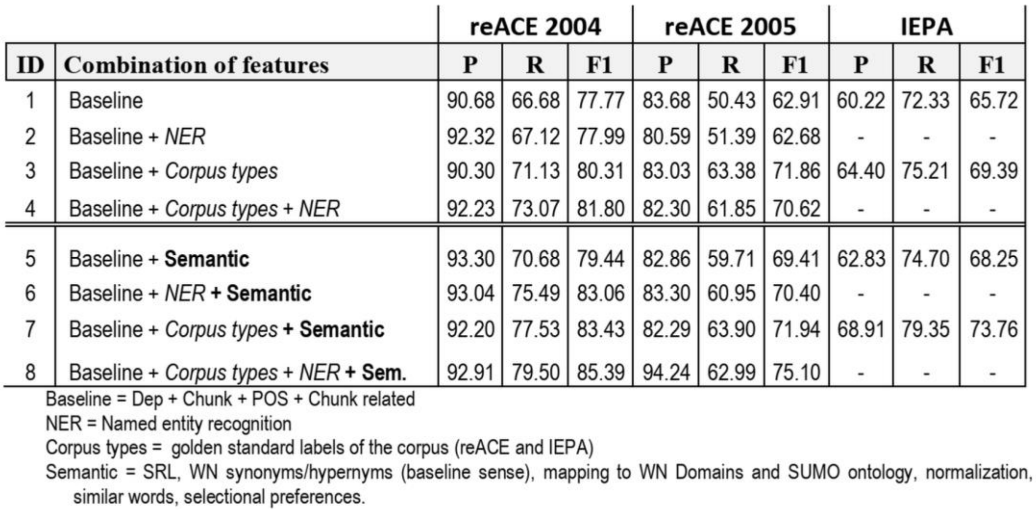
Starting from Line 2 in Tab. 4, a given group of features are incrementally added to the baseline (Line 1) which, in turn, includes the following group of features: lexical, syntactic and structural features, i.e., syntactical dependencies (Dep), chunk information (Chunk), POS tagging, and other chunk related features. The baseline setting corresponds to all the features that do not take into account the semantic features (i.e., lexical semantics and mapping to semantic resources). The other lines (Line 2-4) in Tab. 4 integrate other groups of features (NER, Corpus types) to the baseline: NER denotes recognized named entities whereas Corpus types features denote the golden standard annotations already available in the given corpus. For instance, the reACE datasets provide named entities such as *Organizations* and *Person*, while the *IEPA* corpus only assigns the label *protein* to each term denoting a given protein. The last group of features (*semantic*), denotes the semantic features comprising SRL, synonyms/hypernyms, and mapping of words to WordNet, WordNet Domains, SUMO ontology, similar words, and selectional preferences. The missing entries in IEPA column are due to the fact that typical named entities are useless in the IEPA biomedical corpus, and therefore, they were not considered.

5 Discussion on Experimental Questions

This section discusses both the impact of semantic linguistic features on RE, and related aspects on domain adaptability.

On the Impact of Semantic Linguistic Features. EQ1 can be positively answered because by incrementally incorporating new groups of features to the baseline, that contributed to the improvement of the scores for all datasets. Indeed, the performance improves as more features are used, starting with the F-measure of 77.77 and reaching 81.80 for the reACE 2004 dataset. Analogously for the reACE 2005, the best overall F1 performance (71.86%) may indicate that this dataset is more difficult than the reACE 2004. One possible explanation is that, in the reACE 2005 dataset, some relations (particularly *Business*) are very poorly represented with only 16 positive examples, which hampers the overall score. More importantly, the overall F1 scores suggest that the proposed four groups of features have both a positive and complementary impact on the overall F1 scores for all the datasets evaluated.

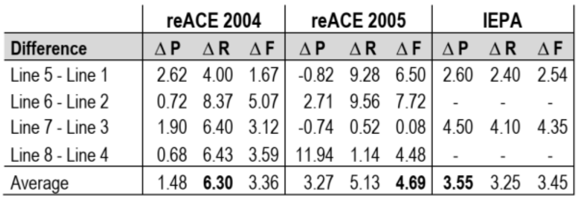
Concerning EQ2, one can notice that including semantic features into the RE process improves average performance in terms of F1 measure for all datasets. In fact, the boost in F1 measure was 4% in average for the reACE datasets, while for the IEPA dataset, the improvement was more than 3%. However, the impact on both P and R scores were unbalanced for the reACE corpora, since the semantic features contributed relatively more in recall than in precision. This contrast with the results on the IEPA corpus that were very balanced. On the one hand, the highest difference in performance was achieved on the reACE 2005 corpus, as the semantic features improved P in almost 12%. On the other hand, for two other combinations in this dataset (Line 5 and Line 7), adding semantic features to the learner in fact slightly hampered precision. Such impact on both P and R were expected since the effect of adding semantic features to the learner could not only improve R over P, but also provide to it an extended layer of categorization of all terms. Contrastingly, for the IEPA corpus, the use of semantic features slightly increased precision more than recall. After inspecting the final induced extraction rules, we found that this was mainly due to the semantic role labeling features. Actually, many verbs denoting the interaction between two proteins terms in IEPA corpus were correctly annotated along with the roles of its arguments. As a result, the ILP-based learner is more precise when a given verb has semantic role features attached to it.



		reACE 2004			reACE 2005			IEPA		
ID	**Combination of features**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
1	Baseline	90.68	66.68	77.77	83.68	50.43	62.91	60.22	72.33	65.72
2	Baseline + *NER*	92.32	67.12	77.99	80.59	51.39	62.68	-	-	-
3	Baseline + *Corpus types*	90.30	71.13	80.31	83.03	63.38	71.86	64.40	75.21	69.39
4	Baseline + *Corpus types* + *NER*	92.23	73.07	81.80	82.30	61.85	70.62	-	-	-
5	Baseline + **Semantic**	93.30	70.68	79.44	82.86	59.71	69.41	62.83	74.70	68.25
6	Baseline + *NER* **+ Semantic**	93.04	75.49	83.06	83.30	60.95	70.40	-	-	-
7	Baseline + *Corpus types* **+ Semantic**	92.20	77.53	83.43	82.29	63.90	71.94	68.91	79.35	73.76
8	Baseline + *Corpus types* + *NER* **+ Sem.**	92.91	79.50	85.39	94.24	62.99	75.10	-	-	-

Baseline = Dep + Chunk + POS + Chunk related
NER = Named entity recognition
Corpus types = golden standard labels of the corpus (reACE and IEPA)
Semantic = SRL, WN synonyms/hypernyms (baseline sense), mapping to WN Domains and SUMO ontology, normalization, similar words, selectional preferences.

Table 4. Results on reACE 2004/2005 and IEPA datasets.



	reACE 2004			reACE 2005			IEPA		
Difference	**Δ P**	**Δ R**	**Δ F**	**Δ P**	**Δ R**	**Δ F**	**Δ P**	**Δ R**	**Δ F**
Line 5 - Line 1	2.62	4.00	1.67	-0.82	9.28	6.50	2.60	2.40	2.54
Line 6 - Line 2	0.72	8.37	5.07	2.71	9.56	7.72	-	-	-
Line 7 - Line 3	1.90	6.40	3.12	-0.74	0.52	0.08	4.50	4.10	4.35
Line 8 - Line 4	0.68	6.43	3.59	11.94	1.14	4.48	-	-	-
Average	1.48	**6.30**	3.36	3.27	5.13	**4.69**	**3.55**	3.25	3.45

Table 5. Performance difference between RE models

On Domain Adaptability. Some authors investigated the domain adaptation problem on NER.

Ciaramita et al. (2005) studied the effects of domain adaptation on NER using two distinct datasets for training and testing. They trained several NER classifiers on the CONLL 2003 dataset and evaluated them on a manually annotated section from the Wall Street Journal portion of the Penn Treebank. They found that, even for such similar types of texts, the results of their supervised NER models dropped significantly. However, they had improved scores (almost 5% in F1-measure) just by coupling their original NER system with both a domain-independent dictionary and a simple string similarity function.

In (Pyysalo, 2008), very similar results were reported when a general POS tagger was employed for tagging a dataset from the biomedical domain.

In our work, this same trend was observed since our RE models trained with all the linguistic features yielded, in average, a relative gain up to 4% in F1-measure. In fact, our RE models trained with deep semantic features outperformed the RE models that did not used them according to the statistical significant tests (Wilcoxon signed-rank) for the difference between F1 scores at α = 0.05 (95% confidence interval). These results are very encouraging and seem to validate the proposed features even in a more difficult application scenario handling the changing of domains. To conclude, the overall achieved results suggest that more accurate semantic information about entity instances can contribute a great deal to RE. This is not surprising, given that semantic information, e.g., hypernyms, and classes from an ontology, typically impose strong constraints on the types of the entities participating in a relation, indicating that such kind of feature can have significant discriminative power in RE. Thus, we can also positively answer to EQ3.

6 Conclusion and Future Work

This paper presented a LRL system for RE employed to test of our hypothesis that a set of features based on a deep linguistic analysis can improve RE. This was demonstrated by experimental evaluation showing that automatic acquisition of a substantial body of linguistic knowledge in combination with an expressive inductive learning technique, it is possible to generate effective RE models. Moreover, such features significantly contributed to generalize the proposed RE to other domains of interest.

This research work can be improved in several ways. We intend to test our solution on larger datasets aiming to promote future scalability. In addition, mechanisms for allowing the parallel execution of the IE process will enable the decomposition of the learning problem into smaller more manageable ones. Finally, the system will be adapted Event Extraction, a fundamental IE task in the biomedical domain (Björne and Salakoski, 2015).

References

Amal Zouap. 2011. An Overview of Shallow and Deep Natural Language Processing for Ontology Learning. *In W. Wong, W. Liu, & M. Bennamoun (Eds.), Ontology Learning and Knowledge Discovery Using the Web*: Challenges and Recent Advances. Hershey, PA: IGI Global, pages 16-37.

Baeza-Yates, R., Ribeiro-Neto, B., 1999. *Modern Information Retrieval*. Addison-Wesley, Boston.

Bentivogli L., Forner P., Magnini B. and Pianta E. 2004. Revising WordNet Domains Hierarchy: Semantics, Coverage, and Balancing. *In COLING Workshop on Multilingual Linguistic Resources*, Geneva, Switzerland, pp. 101-108.

Björne, J., Salakoski, T., 2015. TEES 22: Biomedical event extraction for diverse corpora. *BMC Bioinformatics* 16 (Suppl 16), S4, PMC Web.

Choi S.P, S. Lee., H. Jung, and S. Song. 2013. An Intensive Case Study on Kernel-based Relation Extraction. *In Proceedings of Multimedia Tools and Applications, Springer, US, 2013, pp. 1 -27.*

Ciaramita M., Y. Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. *Adv. in Structured Learning for Text and Speech Processing Workshop* (NIPS).

Ciaramita M., Yasemin Altun. 2006. Broad-Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. *EMNLP*, pages 594- 602.

Ding J., Berleant, D., Nettleton, D., and Wurtele, E. 2002. Mining MEDLINE: abstracts, sentences, or phrases? *In Proc. of the Pacific Symposium on Biocomputing*, 326-337.

Fellbaum, C.D., 1998. *WordNet – An Electronic Lexical Database*. Language, Speech and Communication. MIT Press.

Hachey H., C. Grover, and R. Tobin. 2011. Datasets for Generic Relation Extraction. *Journal of Natural Language Engineering*, Cambridge University Press.

Lin D., Zhao S., Qin L., Zhou M. 2003. Identifying Synonyms among Distributionally Similar Words. *IJCAI* 2003: pages 1492-1493.

Lima R., B. Espinasse, F. Freitas. 2017. OntoILPER: an Ontology- and Inductive Logic Programming-based Relations from Text, in: *Knowledge And Information System (KAIS) Journal*, Springer London.

Lima, R., Espinasse, B. Freitas, F. 2019. A logic-based relational learning approach to relation extraction: The OntoILPER system. Journal of Engineering Applications of Artificial Intelligence. Volume 78, pages 142-157.

Mihalcea R. and E. Faruque. 2004. SenseLearner: Minimally supervised word sense disambiguation for all words in open text". In *Proceedings of ACL/SIGLEX Senseval-3,* Barcelona, Spain.

Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *In Proceedings of Workshop at ICLR.*

Miwa, M., & Bansal, M. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* Berlin, Germany, pages 1105–1116.

Muggleton S. 1991. "Inductive Logic Programming" *New Generation Computing 8* (4): 29.

Nguyen, T.H., Grishman, R. 2015. Relation extraction: Perspective from convolutional neural networks. In: In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. pages 39–48.

Niles, I. and Pease, A. Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. 2003. *In Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE 03)*, Las Vegas, Nevada, June 23-26.

Pyysalo S. 2008. A dependency parsing approach to biomedical text mining, Ph.D. thesis. *Department of Information Technology, University of Turku*, 2008.

Luc De Raedt. 2008. Logical and Relational Learning: From ILP to MRDM *(Cognitive Technologies*). Springer-Verlag, Berlin, Heidelberg.

Santos J. 2010. Efficient Learning and Evaluation of Complex Concepts in Inductive Logic Programming. Ph.D. Thesis, *Imperial College*.

Detecting Anorexia in Spanish Tweets

Pilar López-Úbeda, Flor Miriam Plaza-del-Arco, Manuel Carlos Díaz-Galiano,
L. Alfonso Ureña-López, Maria-Teresa Martín-Valdivia

Department of Computer Science, Advanced Studies Center in ICT (CEATIC)
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain
{plubeda, fmplaza, mcdiaz, laurena, maite}@ujaen.es

Abstract

Mental health is one of the main concerns of today's society. Early detection of symptoms can greatly help people with mental disorders. People are using social networks more and more to express emotions, sentiments and mental states. Thus, the treatment of this information using NLP technologies can be applied to the automatic detection of mental problems such as eating disorders. However, the first step for solving the problem should be to provide a corpus in order to evaluate our systems. In this paper, we specifically focus on detecting anorexia messages on Twitter. Firstly, we have generated a new corpus of tweets extracted from different accounts including anorexia and non-anorexia messages in Spanish. The corpus is called SAD: Spanish Anorexia Detection corpus. In order to validate the effectiveness of the SAD corpus, we also propose several machine learning approaches for automatically detecting anorexia symptoms in the corpus. The good results obtained show that the application of textual classification methods is a promising option for developing this kind of system demonstrating that these tools could be used by professionals to help in the early detection of mental problems.

1 Introduction

Mental health is one of the main concerns of today's society. The World Health Organisation estimates that 1 in 4 individuals experience mental disorders at some stage of their lives. Globally, it is estimated that about 450 million people worldwide are mentally ill, with this kind of illness making up 13% of diseases around the world (Vos et al., 2015).

Traditionally, mental health evaluation is based on face-to-face interviews, self-reported issues or the distribution of questionnaires, which is usually labor-intensive and time consuming. However, in recent years several studies have used different technologies to improve the detection of mental health issues. Specifically, some interesting studies explore the relationship between data from online social networks and users' mental conditions (Rahman et al., 2018). Some of them focus on stress (Thelwall, 2017; Lin et al., 2017), depression (Tsugawa et al., 2015), suicide (O'Dea et al., 2015; Astoveza et al., 2018) or anxiety (Shen and Rudzicz, 2017), and most of them use and extract data from Twitter, probably because the information is open and more accessible than on other platforms, and also because it is one of the most popular social networks among young people. In this paper we focus on mental health problems related to eating disorders because they exhibit the highest mortality rate of any mental illness and 20% of all deaths from anorexia are the result of suicide (Arcelus et al., 2011).

Eating disorders are complex mental disorders considered serious and often fatal illnesses associated with severe disturbances in people's eating behaviors and related thoughts and emotions (Prieto et al., 2014). Common eating disorders include anorexia nervosa, bulimia nervosa, and binge-eating disorder and affect both females and males although they are most usual among young women.

The early detection of eating disorders can increase the chances of recovering, and technology can be applied to developing systems to help professionals. Different approaches to text and data mining methods can be applicable to social media data and may prove invaluable for health monitoring and surveillance. Specifically, Natural Language Processing (NLP), also known as Language Technologies (LT) can be used to generate systems for early anorexia detection. One of the main problems is the lack of resources to train systems and more if we focus on a language other than English.

The main goal of this paper is to develop a system for the automatic detection of anorexia in textual information. For this, we first generated a corpus with tweets written in Spanish including both people talking about anorexia and people talking about healthy food habits. The corpus is called SAD (Spanish Anorexia Detection). Using the SAD corpus, we have developed different models based on Machine Learning approaches that integrate several linguistic features. We have analyzed the results and compared the different approaches.

The rest of the paper is structured as follows: In Section 2 we comment on some related studies. The SAD corpus is described in Section 3, and present some interesting statistics. The different machine learning approaches and the results obtained are shown in Section 4. Finally, the analysis of errors is conducted in Section 5 and conclusions are presented in Section 6.

2 Related Work

The detection of mental health issues using textual information is a recent task mainly inspired by the massive of use and access to social networks. People have become accustomed to using social networks to express all kinds of opinions, feelings and emotions. This valuable information can be captured and treated by an automatic system to learn how people with some health problems use language to express the frustration, depression or bad feelings. In this way, NLP can help to build systems to detect early on health problems such as eating disorders, depression or suicidal tendencies.

Although this task is relatively new, some challenging workshops and shared tasks related to the detection of health conditions have been proposed in recent years. For example, Social Media Mining for Health Applications (SMM4H) is a workshop and share task that has been held since 2016 (Sarker et al., 2016) and continues every year. The main goal is to attract researchers interested in automatic methods for the collection, extraction, representation, analysis, and validation of social media data for health informatics. Furthermore, eRisk (Losada et al., 2017) is a challenging workshop focused on mental health disorders and it has been held from 2017 in the framework of the well-known international conferences CLEF[1]. eRisk explores the evaluation methodology, effectiveness metrics and practical applications (particularly those related to health and safety) of early risk detection on the Internet. The different tasks proposed include depression and anorexia detection.

Concerning to mental health, we can find some interesting papers studying NLP techniques for treating textual information. (Rahman et al., 2018) review several studies focused on detecting mental health using and analyzing the information extracted from social networks. After analyzing several methods, machine learning algorithms, languages and sources of information, the authors conclude that machine learning is the most frequently used method used for mental health detection, with Support Vector Machine (SVM) presenting the best results. In addition, the study shows that Twitter is the major data source from social networks and English is the main language studied in the different papers. In (Prieto et al., 2014) four different health conditions are classified using machine learning methods over a corpus of tweets extracted by applying a set of crafted regular expressions. They integrate some relevant features in order to improve the final system. In addition, this is one of the few papers which center on languages other than English. Specifically, the authors work on Spanish and Portuguese tweets and the results indicate that the approach is a feasible option for tracking health information on social networks.

Regarding eating disorders, we can also find some recent studies. For example, (De Choudhury, 2015) focuses on detecting anorexia on the social network Tumblr using different affective, social, cognitive, and linguistic features. They also analyze the clinical implications of detecting anorexia related content on social media. (Chancellor et al., 2016a) use Instagram in order to study the eating disorders community and propose a statistical model combining topic modeling and clinical annotations. Finally, (Wang et al., 2017) center on Twitter generating a corpus by collecting eating disorders and non-eating disorders data. Then

[1] http://www.clef-initiative.eu/

they train a SVM classifier, obtaining promising results. The high performance achieved suggests that it is feasible to design automatic text analysis tools that give early warnings of signs of eating disorders. However, this study only focuses on English and it is important to prove that the systems can also be applied to other languages. Thus, in this paper we create a Spanish corpus from Twitter with information concerning of anorexia and non-anorexia data. Then we apply several machine learning algorithms in order to demonstrate the feasibility of implementing systems to automatically detect sings of anorexia in Spanish messages written on social networks.

3 SAD Corpus

Anorexia and bulimia are two of the most worrisome eating disorders, affecting adolescents and young people the most. ”Ana y mia” are the names used on the web pages that promote anorexia and bulimia to identify themselves. ”Ana” is anorexia and ”mia” is bulimia. But it is not a recent phenomenon, it began to become popular on the Internet in 2004 (Campos Rodríguez, 2007). Today, they have millions more pages and loyal followers, and the Internet has connected thousands of people with eating disorders. For this reason there are currently several studies of this disease (Moessner et al., 2018; Bermejo et al., 2011; Chancellor et al., 2016b). Specifically, for Spanish there is no set of Twitter messages concerning this problem, and for this reason we have compiled our own corpus, SAD (Spanish Anorexia Dataset) in order to accomplish the experiments.

3.1 Data Collection

We decided to use the social network Twitter because it is currently one of the most common sites for sharing information. This social network allows people to freely post short messages (called tweets) of up to 280 characters. Twitter has rapidly gained popularity worldwide, with more than 326 million active users generating more than 500 million tweets daily.

The task of downloading tweets has been performed through the Application Programming Interface (API) offered by Twitter. The API allows us to download messages using a query in a specific language. Our retrieving system always sets the option to Spanish, thus our classification system only works on tweets in Spanish. However, our method can easily be adapted to other languages since the Twitter API allows specification of the language of the posts retrieved.

In order to obtain enough tweets, we had to download messages from past years, more concretely, in a date range of February 2014 to March 2019.

To make the corpus more interesting, we used as a query different hashtags related to food, nutrition, diet and healthy living in a converse way to anorexia. We collected data referring to anorexia using as query the hashtag *#anaymia* on Twitter. In addition, we collected three sets of reference data as negative samples using the hashtag *#realfood #comidareal* and *#fitness*.

Label 1 (anorexia) has been assigned to tweets that satisfy the query *#anaymia*, label 0 (control) for the other cases. Different messages are shown in Table 1 and in Table 2 we can see the English translation.

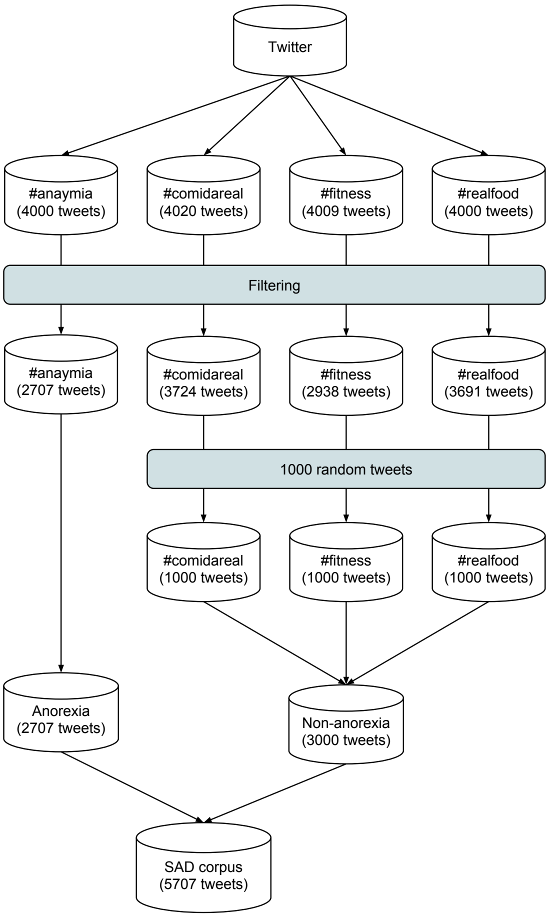


Figure 1: Process of generating the corpus from Twitter.

Tweet	*Label*
Solo quiero llegar a mi casa a comer csm, no puedo más con esta hambre. Pero el hambre es belleza entrando a tu cuerpo.	1
La comida de hoy es ligera, muy ligera. Alcachofas al horno, simplemente llevan ajo, aceite, perejil y sal. Mmmm. #masendocrino #alcachofas #dietasana #dietamediterranea #aove #aceitedeoliva #hungry #cocinaespañola #comidacasera #foodpic #banquetesv	0

Table 1: Examples of Spanish tweets tagged in SAD corpus.

Tweet	*Tag*
I just want to get home to eat, I can no longer cope with this hunger. But hunger is beauty coming into your body.	1
Today's food is light, very light. Baked artichokes, simply with garlic, oil, parsley and salt. Mmmm. #masendocrino #alcachofas #dietasana #dietamediterranea #aove #oaceitedeoliva #hungry #cocinaespañola #comidacasera #foodpic #banquetesv	0

Table 2: Example of translated tweets.

3.2 Data Filtering

Secondly, the extracted data is very noisy, so the set requires thorough cleaning before any analysis can be carried out. The language used by Twitter users contains some attributes that we had to remove to provide useful information for the classification process. This filtering that was performed:

- Repeat - the first filter to be performed was the removal of repeated tweets. Repeated tweets do not bring new information to the collection.
- Hashtag queries - we removed from the tweets the hashtag that we used as a query for downloading messages.
- All hashtag - we also removed tweets that only contained hashtags in the message. This step was followed since the experiments described in Section 4 were carried out without using hashtags.
- Short tweet - finally, tweets containing fewer than four words were removed since we consider that they do not provide enough representative information.

The objective was to obtain as balanced a corpus as possible. For cases of anorexia all tweets were incorporated. For the negative cases, we followed a different strategy, with 1000 random tweets being taken from each hashtag (*#comidareal*, *#fitness* and *#realfood*), in this way, the corpus contains 2707 tweets annotated as positive (anorexia) and 3000 tweets annotated as negative (control). Figure 1 shows the number of tweets downloaded and how the collection decrements at each step.

3.3 Corpus Statistics

In this Section we will focus on obtaining statistics referring to the corpus containing relevant information. These statistics refer to the number of words, stopwords, hashtags, and part-of-speech tagging, among others.

The first study carried out consisted of obtaining the number of tweets, the number of words, the number of users and the number of stopwords in Spanish that exist in the corpus. This is shown in Table 3, where we can find the difference between the messages annotated with anorexia and those annotated as control.

It is interesting to see how the percentage increase in controlled tweets is 44% greater than the anorexia vocabulary, taking into account the number of total words as it can be seen in Table 3. But this information is reasonable because the average of tweet words is higher in controlled cases.

The grammatical labelling can be found in the Table 4. For this study we have used the spaCy[2]

[2] https://spacy.io/

	Total	*Anorexia*	*Control*
Number of tweet	5707	2707	3000
Number of different users	2585	1120	1466
Number of total words	122798	43179	79619
Number of different words	24635	8761	18515
Average of tweet words	21.52%	15.95%	26.54%
Number of total stop words	30619	13118	17501
Number of different stop words	207	183	185
Average of tweet stop words	5.37%	4.85%	5.83%

Table 3: Linguistic statistics on SAD corpus.

library with the module *es_core_news_sm*[3]. spaCy is a free open-source library for NLP in Python.

Table 4 shows the statistics obtained, and in it we can see relevant information on verbs, nouns, adjectives and adverbs. We found special interest in the high number of verbs and nouns used in annotated tweets without anorexia.

We wanted to obtain some statistics about the mood of users and how they express themselves through social networks. To obtain this information we used the resource iSOL (Molina-González et al., 2013). This resource has a list of opinion indicator words in Spanish independent of the domain. The list consists of 2,509 positive words and 5,626 negative. The results are described in Table 5. This table shows that users with anorexia problems use more negative language than users without anorexia. The same happens in the opposite case, whereby the tweets annotated as controlled are written with more positive words.

Finally, Table 6 shows some data about the use of hashtags in the messages collected. We can observe that the number of hashtags used in controlled tweets is much higher than on the contrary, and consequently there is also more variety of hashtags in messages annotated without anorexia.

4 Experiments and Results

In this section, we describe different experiments we carried out to test the validity of the SAD corpus. In particular, we trained several classifiers based on machine learning.

4.1 Pre-Processing

Pre-processing the data is the process of cleaning and preparing the text for classification. It is one of the most important steps because it should help improve the performance of the classifier and speed up the classification process. Online texts usually contain lots a great deal of noise and uninformative parts which increases the dimensionality of the problem and hence makes the classification more difficult. For this reason, we applied pre-processing techniques in order to prepare the data for the text classification. In particular, we preprocessed the tweets of the SAD Dataset following these steps: The tweets were tokenized using NLTK TweetTokenizer[4] and all hashtags were removed.

Features in the context of text classification are the words, terms or phrases that express the opinion of the author. They have a higher impact on the orientation of the text. There are several ways to assess the importance of each feature by attaching a certain weight to it in the text. We use the most popular: The Term Frequency Inverse Document Frequency scheme (TF-IDF). Specifically, using this scheme each tweet is represented as a vector of unigrams.

4.2 Machine Learning Algorithms

Machine learning techniques are popular in the binary classification. For this reason, we decide to employ different machine learning algorithms in order to classify the corpus in anorexic and non anorexic tweets. The algorithms are Support Vector Machine (SVM), Naive Bayes (NB), Random Forest (RF), Multilayer Perceptron (MLP), Logistic Regression (LR) and Decision Tree (DT).

4.3 Results

In this subsection, we report and discuss the performances of our systems on the Spanish anorexic

[3] `https://github.com/explosion/spacy-models/releases/tag/es_core_news_sm-2.1.0`

[4] `https://www.nltk.org/api/nltk.tokenize.html`

	Total	*Anorexia*	*Control*
Adjectives in corpus	15332	3996	11336
Nouns in corpus	28594	8536	20058
Verbs in corpus	13647	5592	8055
Adverbs in corpus	5326	2518	2808
Number of different adjectives in corpus	4786	1493	3638
Number of different nouns in corpus	7326	2769	5449
Number of different verbs in corpus	4990	2342	3256
Number of different adverbs in corpus	622	296	455
Average adjectives in tweet	2.69%	1.48%	3.78%
Average nouns in tweet	5.01%	3.15%	6.69%
Average verb in tweet	2.39%	2.07%	2.69%
Average adverbs in tweet	0.93%	0.93%	0.94%

Table 4: Part-of-speech tagging statistics on SAD corpus.

	Total	*Anorexia*	*Control*
Negative words in corpus	1549	1070	479
Positive words in corpus	2530	807	1723
Different negative words in the corpus	456	319	236
Different positive words in the corpus	460	227	358
Average of negative words in tweet	0.44%	0.30%	0.57%
Average of positive words in tweet	0.27%	0.40%	0.16%

Table 5: Statistics about positive and negative words in the corpus.

classification task on the SAD corpus. In order to evaluate and compare the results obtained by our experiments, we use the usual metrics in text classification, called precision *(P)*, recall *(R)*, F-score *(F_1)* and Accuracy *(Acc)*.

To determine the optimal classification algorithm, we conducted experiments with the six classification models. We used 10-fold cross validation to evaluate the machine learning classification approaches including: The *SVM* classifier, the *Naive Bayes* classifier, the *Random Forest* classifier, the *Decision Tree* classifier, *Logistic Regression* and the *Multilayer Perceptron* classifier. The test results achieved by these algorithms on the SAD corpus are shown in Table 7.

The classifiers with the best performance were SVM and MLP with the default settings in the Scikit-learn 0.19.1 package (Pedregosa et al., 2011). The other classifiers also showed good results, all achieving an accuracy score superior to 80%. It should be noted that they performed well in both classes (anorexia and control) because the corpus is well balanced.

5 Error Analysis

The main purpose of this section is to carry out an error analysis to identify the weaknesses of our system. For this, we analyze some of the tweets not correctly classified by our system.

Of the total number of tweets (5707), 478 were not correctly classified, only 8.38% of the total tweets. In Figure 2, the confusion matrix of our system can be seen. It shows that there were more false positives (300) than false negatives (178). Therefore, we analyzed some of these tweets manually in order to find the main reasons why our system can be confused.

Table 2 presents some examples of tweets incorrectly classified by our system and Table 1 shows the corresponding translation. Specifically, two of the tweets are false positives and the other two false negatives. On the one hand, if we look at the false positives, we can see that two of the reasons why our system can be wrong is because it detects that there are words related to food and also that the vocabulary of the other tweets labeled as control is very similar to the vocabulary used in anorexia. Therefore, the system is sometimes con-

	Total	*Anorexia*	*Control*
Hashtags in corpus	25133	5037	20096
Different hashtags in corpus	7479	1282	6341
Average hashtags in tweet	4.40%	1.86%	6.70%

Table 6: Statistics about hashtag in the corpus.

Classifier	Anorexia			Control			Macro-avg			
	P	R	F_1	P	R	F_1	P	R	F_1	Acc
SVM	0.894	0.934	0.914	0.938	0.9	0.919	0.916	0.917	0.916	0.916
MLP	0.894	0.934	0.913	0.938	0.9	0.92	0.916	0.917	0.916	0.916
RF	0.837	0.895	0.865	0.899	0.842	0.87	0.868	0.868	0.867	0.867
NB	0.823	0.849	0.835	0.859	0.835	0.847	0.841	0.842	0.841	0.841
LR	0.846	0.898	0.871	0.902	0.853	0.878	0.874	0.875	0.874	0.874
DT	0.795	0.823	0.809	0.835	0.809	0.822	0.815	0.816	0.815	0.815

Table 7: Results obtained by different classifiers on the SAD corpus (10-fold cross validation).

fused when, for example, the user talks about sport in general. On the other hand, if we focus on false negatives, we see that one of the problems is the irony in the tweet and another of the problems is when the user is transmitting a negative emotion but does not say the cause.

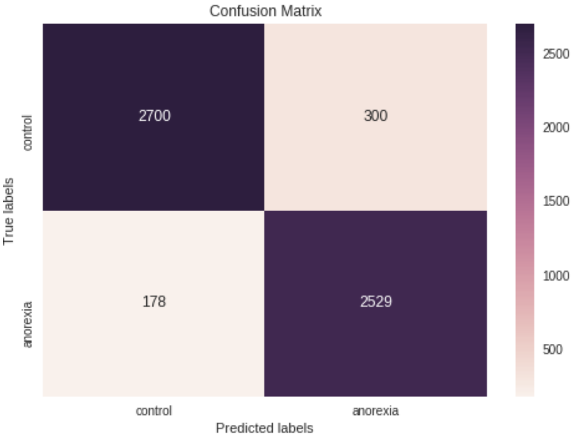


Figure 2: Confusion matrix.

6 Conclusion

This article presents a new corpus in Spanish for detecting anorexia in social network messages. Several systems are also developed to test the performance of this task with different classifiers. The results obtained show that the performance is very similar in all systems, although SVM and MLP are the only ones that obtain accuracy above 0.9.

Error analysis reveals that there are cases where classification systems do not work properly. Our next goal will be to apply other techniques (such as irony detection or sentiment analysis) in cases where textual information is poor or where rhetorical figures such as irony and sarcasm are used.

Acknowledgments

This work has been partially supported by Fondo Europeo de Desarrollo Regional (FEDER), LIVING-LANG project (RTI2018-094653-B-C21) and REDES project (TIN2015-65136-C2-1-R) from the Spanish Government.

References

Jon Arcelus, Alex J Mitchell, Jackie Wales, and Søren Nielsen. 2011. Mortality rates in patients with anorexia nervosa and other eating disorders: a meta-analysis of 36 studies. *Archives of general psychiatry* 68(7):724–731.

Ghelmar Astoveza, Randolph Jay P Obias, Roi Jed L Palcon, Ramon L Rodriguez, Bernie S Fabito, and Manolito V Octaviano. 2018. Suicidal behavior detection on twitter using neural network. In *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE, pages 0657–0662.

Belén G Bermejo, Luis Ángel Saúl, and Cristina Jenaro. 2011. La anorexia y la bulimia en la red. ana y mia dos malas compañías para las jóvenes de hoy [the anorexia and bulimia on the web: Ana and mia two "bad company" for youth today]. *Acción psicológica* 8(1):71–84.

José Miguel Campos Rodríguez. 2007. Anorexia, bulimia e internet. aproximación al fenómeno pro-ana

Tweet	True label	Predicted
”El físico no importa”	1	0
Momento de escuchar música para relajarme y olvidarme de la mierda de mundo en el que vivo	1	0
Hola @IKEASpain mi bebé de 9 meses come sólido, no ha comido un potito nunca y me parece injusto que a él le cobréis la comida	0	1
Rutina de ejercicios para glúteos	0	1

Table 8: Examples of tweets badly classified by our system.

Tweet	True label	Predicted
”The physical aspect doesn’t matter”	1	0
Time to listen to music to relax and forget about the shitty world I live in	1	0
Hello @IKEASpain my 9 month old baby eats solid food, and has never eaten a baby food and I think it’s unfair that you charge him for the food	0	1
Buttock Exercise Routine	0	1

Table 9: Examples of translated tweets badly classified by our system.

y mía desde la teoría subcultural. *Frenia. Revista de Historia de la Psiquiatría* 7(1):127–144.

Stevie Chancellor, Zhiyuan Lin, Erica L Goodman, Stephanie Zerwas, and Munmun De Choudhury. 2016a. Quantifying and predicting mental illness severity in online pro-eating disorder communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, pages 1171–1184.

Stevie Chancellor, Jessica Annette Pater, Trustin Clear, Eric Gilbert, and Munmun De Choudhury. 2016b. # thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, pages 1201–1213.

Munmun De Choudhury. 2015. Anorexia on tumblr: A characterization study. In *Proceedings of the 5th international conference on digital health 2015*. ACM, pages 43–50.

Huijie Lin, Jia Jia, Jiezhong Qiu, Yongfeng Zhang, Guangyao Shen, Lexing Xie, Jie Tang, Ling Feng, and Tat-Seng Chua. 2017. Detecting stress based on social interactions in social networks. *IEEE Transactions on Knowledge and Data Engineering* 29(9):1820–1833.

David E Losada, Fabio Crestani, and Javier Parapar. 2017. erisk 2017: Clef lab on early risk prediction on the internet: experimental foundations. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, pages 346–360.

Markus Moessner, Johannes Feldhege, Markus Wolf, and Stephanie Bauer. 2018. Analyzing big data in social media: Text and network analyses of an eating disorder forum. *International Journal of Eating Disorders* 51(7):656–667.

M Dolores Molina-González, Eugenio Martínez-Cámara, María-Teresa Martín-Valdivia, and José M Perea-Ortega. 2013. Semantic orientation for polarity classification in spanish reviews. *Expert Systems with Applications* 40(18):7250–7257.

Bridianne O’Dea, Stephen Wan, Philip J Batterham, Alison L Calear, Cecile Paris, and Helen Christensen. 2015. Detecting suicidality on twitter. *Internet Interventions* 2:183–188.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Víctor M Prieto, Sérgio Matos, Manuel Álvarez, Fidel Cacheda, and José Luís Oliveira. 2014. Twitter: A good place to detect health conditions. *PLoS ONE* 9:e86191.

Rohizah Abd Rahman, Khairuddin Omar, Shahrul Azman Mohd Noah, and Mohd Shahrul Nizam Mohd

Danuri. 2018. A survey on mental health detection in online social network. *International Journal on Advanced Science, Engineering and Information Technology* 8(4-2):1431–1436.

Abeed Sarker, Azadeh Nikfarjam, and Graciela Gonzalez. 2016. Social media mining shared task workshop. In *Biocomputing 2016: Proceedings of the Pacific Symposium*. World Scientific, pages 581–592.

Judy Hanwen Shen and Frank Rudzicz. 2017. Detecting anxiety through reddit. In *Proceedings of the Fourth Workshop on Computational Linguistics and Clinical Psychology—From Linguistic Signal to Clinical Reality*. pages 58–65.

Mike Thelwall. 2017. Tensistrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management* 53(1):106–121.

Sho Tsugawa, Yusuke Kikuchi, Fumio Kishino, Kosuke Nakajima, Yuichi Itoh, and Hiroyuki Ohsaki. 2015. Recognizing depression from twitter activity. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, pages 3187–3196.

Theo Vos, Ryan M Barber, Brad Bell, Amelia Bertozzi-Villa, Stan Biryukov, Ian Bolliger, Fiona Charlson, Adrian Davis, Louisa Degenhardt, Daniel Dicker, et al. 2015. Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990–2013: a systematic analysis for the global burden of disease study 2013. *The Lancet* 386(9995):743–800.

Tao Wang, Markus Brede, Antonella Ianni, and Emmanouil Mentzakis. 2017. Detecting and characterizing eating-disorder communities on social media. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, pages 91–100.

A Type-Theoretical Reduction of Morphological, Syntactic and Semantic Compositionality to a Single Level of Description

Erkki Luuk
Institute of Computer Science, University of Tartu, J. Liivi 2, Tartu 50409, Estonia
erkkil@gmail.com

Abstract

The paper presents NLC, a new formalism for modeling natural language (NL) compositionality. NLC is a functional type system (i.e. one based on mathematical functions and their types). Its main features include a close correspondence with NL and an integrated modeling of morphological, syntactic and semantic compositionality. The paper also presents an implementation of NLC in Coq. The implementation formalizes a diverse fragment of NL, with NLC expressions type checking and failing to type check in exactly the same ways that NL expressions pass and fail their acceptability tests. Among other things, this demonstrates the possibility of reducing morphological, syntactic and semantic compositionality to a single level of description. The level is tentatively identified with semantic compositionality — an interpretation which, besides being supported by results from language processing, has interesting implications on NL structure and modeling.

1 Introduction

As shown by Asher (2014), Luo (2010, 2014) and Ranta (1994), in a logical approach (i.e. in one with simpler alternatives such as zeroth, first, second and higher order logic), complex type theories outshine simpler ones in accounting for natural language (NL) phenomena like anaphora, selectional restrictions, etc. The basic judgement in type theory, $a : A$, is read "term a has type A", where

(i) type := a category of semantic value.

Since the notion of type is inherently semantic, it is by definition suited for analyzing universal phenomena in NL, as NL semantics is largely universal (as witnessed by the possibility of translation from any human language to another).

2 Interpreting Natural Language

NL functions and function applications have a distinctive form. Specifically, any *morphosyntactically admissible concatenation* $\mathrm{c}(a, e_1, ..., e_m)$, which is parsed as $a(e_1, ..., e_m)$, is a function or function application term (FFAT) in NL ($e_1, ..., e_m$ may be null). For example, *red*, *red book*, *livre rouge*, etc., are FFATs in idiosyncratic notations (viz. English, French, etc.). As linguistic expressions are FFATs, they are naturally parsed as functions or function applications.

How to decide whether a particular application $a(e_1, ..., e_m)$ holds? Usually, one has a basic intuition about what modifies what (modification is a subcase of function). The main sources for the intuition are morpheme or word classes and semantic contribution tests. For example, *-s* modifies (i.e. is a relation over) *work* in *works* rather than vice versa, as (1) affixes modify stems not vice versa, (2) person/tense and plural markers modify flexibles[1] rather than vice versa, and (3) *-s* contributes to the meaning of *work* in *works* rather than vice versa. By a similar argument, *heavy* modifies *rain* in *heavy rain* rather than vice versa, *sleeps* modifies *john* in *john sleeps* rather than vice versa, etc. In each case, there's a clear asymmetry between functions of the components, as conveyed by (1)–(2) the functions of word and morpheme classes and (3) semantic contribution tests.

While the above methods may seem sufficient by normal standards, for type-driven theories an even more rigorous way of testing NL relations

[1] See Luuk (2010).

is available. We can model a (reasonably large) fragment of NL in a suitable type system to make it *(fail to) type check in exactly the same ways as (hypothetical) NL expressions pass (and fail) acceptability tests*. Call this principle "the correspondence criterion". Arguably, this possibility, the ultimate test for any type-driven linguistic theory, has not been fully explored, while several significant steps in this direction have been taken. Chatzikyriakidis and Luo (2014b, 2016) describe (among other things) a use of proof assistants for testing the logical soundness of linguistic theories, while Grammatical Framework (GF, Ranta (2004)), a statically typed programming language for writing NL grammars, gets closest to a *type-theoretical implementation of NL*. However, GF is a high level tool, mathematically opaque to the end user, and quite specialized. Because it is geared towards writing NL grammars, it does not offer a selection of different mathematical formalisms to work with, being thus unsuitable for a general (low level) modeling of NL and theories thereof. In addition, it does not concern itself with modeling compositional semantics, although a FrameNet API for GF has been proposed and partly implemented (Gruzitis et al., 2012; Gruzitis and Dannélls, 2017).

A powerful feature of typed theories (especially of the richly typed[2] ones — Chatzikyriakidis and Luo (2014b)) is that they allow to capture not only grammatical but also semantic acceptability. The paper shows that a combination of functions and rich typing makes it possible to use *a single type system for modeling the core of NL morphology, syntax and compositional semantics*, thus questioning the soundness of the theoretical distinction between these different NL "layers" (and partly eradicating their even more theoretical "connections" such as the syntax-semantics interface). Because of a wide selection of mathematical formalisms in a richly typed setting, combined with a relatively "instant" compile time type checking, proof assistants (e.g. Coq, Agda, Lean) are suitable tools for such work (cf. Chatzikyriakidis and Luo (2016)). As shown below, implementing polymorphic functions with a subclass of compound types (called lump types) allows to partly collapse different "levels" of NL (morphology, syntax, and compositional semantics).

3 A Note on Selectional Restrictions

More or less overlooked in Montagovian (Montague, 2002) and categorial (Lambek, 1958) traditions, selectional restrictions have recently become a focus of intense research in modern type theories (Asher, 2014; Luo, 2010; Bekki and Asher, 2013). The essence of the semantic (and perhaps even more logical than linguistic) phenomenon of selectional restrictions is prescribing types for a relation's arguments.

There is an important difference between (1) arguments belonging to types and (2) relations imposing types on their arguments. While an argument can clearly belong to different types[3], a relation should not impose different types on its *k*th argument, for a fixed *k*. Modeling selectional restrictions along with grammar is an important motivation for lump types, described below (in section 5.1.1).

4 Introducing NLC

Call the formalism we are considering NLC. It is a type system for modeling NL syntax, morphology and compositional semantics — briefly, compositionality in NL. The basic unit of NLC is a function of a small (usually ≤ 3) arity. The expressions of NLC are functions, function applications, function types (Π-types), lump types, terms of lump types, and universes (types of types). Elementary terms of NLC are functions. This is possible if we interpret "proper arguments" as nullary functions (functions that take no arguments). So functions are divided into proper arguments and proper functions (the latter being functions that take arguments). For a fixed NL k, let T^k be a proper type variable of NLC, where "proper type" refers to a type that is not a universe, and $\mathcal{M}^k$ the set of morphemes[4]. Then we have the rule:

$$\frac{a \in \mathcal{M}^k}{a : T^k} \text{ ATV-Intro},$$

for generating atomic terms of NLC and introducing type variables for them. The rule ATV-Intro says that all morphemes have types in NLC (technically: "if a is a morpheme of language k then a has type T^k").

Some morphemes (e.g. stems) occur only in an argument position (i.e. are proper arguments),

[2] Rich typing coincides mainly (although perhaps not exclusively) with dependent and/or polymorphic types.

[3] E.g. a book is a physical and informational object.

[4] Morphemes are smallest signs (form-meaning correspondences) in language.

while others (e.g. plural markers) are proper functions. More generally, a function or function application is a parsimonious interpretation of morphemes, words, phrases and sentences in NL. *Sentences, multimorphemic words and phrases are function applications*. This amounts to a rigorous interpretation of the more general "principle of compositionality", as it is known at least since Frege[5]. Broadly speaking, there are only two ways to explain the emergence of compositional meaning: by specifying a relation together with its (a) arguments or (b) type. The first corresponds to e.g. function application and the second to function declaration.

In a functional type system, generating complex terms and introducing type variables for them is straightforward (rule CTV-Intro, with T_i^k ranging over proper types):

$$\frac{e_1 : T_1^k, ..., e_m : T_m^k \quad a : T_1^k \to ... \to T_m^k \to T_{m+1}^k}{a(e_1, ..., e_m) : T_{m+1}^k}$$

where $a(e_1, ..., e_m)$ is an application and $T_1^k \to ... \to T_{m+1}^k$ the usual (right-associative) function type. Since CTV-Intro is the standard function type elimination rule, the function type introduction rule is derivable from CTV-Intro.

5 NLC: The Types

Since grammatical (and semantic) categories have a limited, finite number of members, we need some atomic types with limited membership. Let $\mathcal{U}$ denote the top-level universe of NLC. We use axioms of the form $S : \mathcal{U}$ and $T : S$, where S may be a universe, for introducing atomic type constants corresponding to linguistic categories like stem, case, nominative, noun, verb, etc. For proper functions, we need function types (Π-types). Besides this, we need only polymorphism and lump types, both of which can be (in various ways) implemented with function types.

Since a term of type A may contain another term of type A (or in case of a function type, take another term of the type as an argument), we have sufficient complexity without recursion (self-reference, which we do not need). For example, a sentence A containing another sentence B does not imply recursion unless $A = B$ or B references A. Thus, we have same-type-reference without self-reference.

5.1 Polymorphism and Lump Types

The complexity of NLC goes well beyond regular function types. In considering a NL expression type-theoretically, one is frequently inclined to assign it to more than one type. Confining our analysis to only the linguistically relevant features, we may want to type e.g. *stone* as a flexible, physical object, word in nominative case, etc. A way — corresponding to polymorphism — to go about this is to define coercions to (i.e. coercive subtyping for) all the types we need. In fact, we have three possibilities: either we (1) lose some type information, type *stone* (2) polymorphically or (3) with a lump type.

As the nominal and verbal readings of *stone* preclude each other, a polymorphism is required if we want to encode them both (cf. footnote 7). In many other cases, however, a lump type may be preferred. Thus, NLC features types for morphemes, function types, lump types and polymorphism.

5.1.1 Lump Types

While possibility (3) is new, the superclass of lump types, compound types have been used for NL modeling in the form of multi-field record types (Cooper, 2005; Ranta, 2004; Luo, 2011; Chatzikyriakidis and Luo, 2014a). Also, some kind of polymorphism (e.g. by subtyping — Luo (2010)) is frequently thought to be necessary. However, the use of compound types has been so far confined to record types only, i.e. not properly generalized[6]. A compound type is a type which is a syntactic compound of multiple types or their terms. Normally, the compounded types are different; in the degenerate case, they are the same. Examples (or implementations) of compound types include Σ-, Π-, Cartesian product and multi-field record types.

We defined types as "categories of semantic value" but, as the example of *stone* shows, for NL expressions the value covers not only linguistic semantics but also the meanings of syntactic and

[5] The principle is more general because it holds also for interpretations of formal languages.

[6] In many (most?) programming languages that support them, the notion of "compound type" (or "compound data type") is synonymous with a multi-field record type. This is *not* the way it is used here. While a record type can be defined as a (mathematically more fundamental) Σ-, Π- or Cartesian product type (Constable, 2003), I have never seen it defined as a function application.

morphological categories (we will return to this point in section 9). As compared to (1) and (2), packing an expression's meaning into a lump type allows to do away with both the loss of information and typing complexity. Of course, the lump type itself will be complex but this will, hopefully, present less problems than alternatives (1) and (2). As a bonus, the underlying linguistic model will simplify on account of reducing compositional semantics and parts of morphology and syntax to a single level of description. Below is the rule for lump type introduction (LT-Intro):

$$\frac{B : T^k \quad c_0 : C_0^k,\ ...,\ c_{n+1} : C_{n+1}^k \quad B \mapsto c_0, ..., B \mapsto c_{n+1}}{B : C_0^k..C_{n+1}^k}$$

where T^k is a proper type variable, B a term constant and $C_0^k, ..., C_{n+1}^k$ type constants in NLC, $x \mapsto y$ a function interpreting x as y[7], and $C_0^k..C_{n+1}^k$ the notation for a lump type (comprising types C_0^k through C_{n+1}^k, i.e. there must be at least two). LT-Intro is formalism-agnostic — the exact mathematical structure used for lumping is irrelevant. In particular, as shown in Supplement A[8], we can implement lump types for NL as 1) record types, 2) function applications, 3) Cartesian product types, or 4) Π-types. In languages that have them (e.g. TypeScript, Flow...), it is natural to encode lump types as intersection types. Lump types are defined as compound types that satisfy LT-Intro (i.e. we are not interested in empty lump types).

Supplement B[9] proceeds to formalize a diverse fragment of NL with function applications. The fragment comprises stems, nouns, verbs, flexibles, proper names, pronouns, XPs[10], adjectives, sentential, adjectival and generic adverbs, determiners, demonstratives, quantifiers, tense-aspect-mood, gender, number and nonfinite markers, cases, adpositions, sentences (both simple and complex), connectives, connective phrases (for substantives, adjectives, adverbs and sentences), complementizers, copulas, and selectional restrictions (for physical, informational, limbed, biological, animate and sentient entities).

The linguistic categories not formalized in Supplement B are gerunds, participles, auxiliary verbs, interrogatives, numerals, negation, mass/-count distinction and unspecified selectional restrictions (and possibly others). These are omitted not because of a special difficulty formalizing them would pose but because the formalized fragment is sufficiently expressive (and representative of NL) to make the points of utility and feasibility of NL formalization with the combination of compound and polymorphic types. The formalization has been done in the proof assistant Coq (ver. 8.9), making use of its features like Ltac programming, custom notations, etc. Besides showing the use of lump (viz. application) types in NL modeling, Supplement B should fulfill the abovementioned "correspondence criterion".

5.1.2 Polymorphism

Besides lump types, there is some use for polymorphism as well — if not for any other reason, then because NL expressions may be underspecified. E.g. *sleep* and *stone* are flexible stems that can function both as nouns and verbs. As a verb, *sleep* selects for a specific argument, say, a sentient entity (only higher animals can sleep — for trees, stones and bacteria it is not an option). As a noun, it is quite similar to many others: a stem, a flexible in singular, an event, etc. Since verbs are functions, *sleep*'s type must be a function type, but since it also functions as a noun, a polymorphism is desirable. The alternative, defining two distinct *sleeps*, one noun and one verb, would be redundant and inelegant — esp. in a programming language, where (differently from NL) they would have to be formally distinct (e.g. `sleep` and `sleep0`) even in the absence of any discriminating context (markers and/or arguments).

There are several ways to implement polymorphism, but dependent and/or polymorphic types and subtyping are the most common. For example, Coq supports polymorphic types (e.g. `∀x:Type, x`), but an additional formalization layer is sometimes desirable to improve type inference. One of the main obstacles for formalizing NL in Coq (and likely also in other proof assistants) is that NL type inference is much more powerful than that of the (terms of) relatively simple types like sorts, sensu stricto variables (e.g. those defined with `Parameter` and `Variable` in Coq) and function types over them. The reason is that the simple types have an unbounded num-

[7] The interpretations must not preclude each other; if they do (as e.g. the interpretations of *run* as a noun and verb), they are dealt with polymorphism instead.

[8] https://gitlab.com/jaam00/nlc/blob/master/compound.v

[9] https://gitlab.com/jaam00/nlc

[10] Frequently (and theory-dependently) alternatively referred to as NPs or DPs.

ber of terms, while in NL the number is fixed, very limited, and usually known in advance[11]. The only counterexamples to this rule are phrases, clauses and complex words. So the additional layer of formalization is used for downgrading the over-powerful simple types to something on which type inference would work. In Coq, the main devices for such downgrading are inductive types, “canonical structures”, and type classes. Our formalization uses them all, relying most heavily on canonical structures (essentially, canonical records of a record type).

6 Truth-Functionality

So far, the semantics developed here is not truth-functional, i.e. it is type- but not model-theoretic. As type theory is ‘semantic’ by definition, it is clearly sufficient for a semantical cast of semantics without a recourse to model theory. This is evident in programming language semantics, where the role of model theory is marginal as compared to that of type theory. Traditionally, in natural language semantics the opposite is true, as sentential semantics is usually construed as model-theoretic even in a type-theoretical setting (e.g. Chatzikyriakidis and Luo (2015, 2016)). The obvious reason for this is that the (prevailing, i.e. Montagovian) tradition is model-theoretic. For this reason, an optional truth-functionality module has been added to the implementation. Degenerate models (where all NLC sentences (S) are uniformly true, false or undecidable) can be specified trivially by subtyping, e.g.

```
Parameter s_prop:> S -> Prop.
(* all S-s undecidable *)
```

and with only a little effort non-trivial models can be specified, too (with subtyping and a special notation matching NLC constructions with appropriate values). Below is an example from Supplement B:

```
Check $(PRES walk john): Prop. (*False*)
Fail Check $(PAST walk stone). (*type mismatch*)
Check $(PRES sleep john). (*True*)
Check $(PRES sleep (PL boy)). (*False*)
Check $(PAST sleep (-s boy)). (*True*)
Fail Check $sleep. (*type inference fail*)

(*a trivial proof that "boys
don’t sleep" and "john sleeps"*)
Theorem pres: (~ ($ (PRES sleep (-s boy)))) /\
($ (PRES sleep john)). Proof. firstorder. Qed.
```

7 Comparisons

In section 2 we compared NLC with related typed approaches. This section takes a broader (albeit still related) perspective, comparing NLC with Combinatory Categorial Grammar and Head-Driven Phrase Structure Grammar.

7.1 Combinatory Categorial Grammar

A feature of NLC, Combinatory Categorial Grammar (CCG) (Steedman, 2000) and some other categorial formalisms is that they tend to do away with syntax: “...syntactic structure is merely the characterization of the process of constructing a logical form, rather than a representational level...” — Steedman (2000), p. xi[12]. The main differences are as follows. CCG is a categorial formalism, NLC not. CCG has complex (combinatorial) syntactic types of the form $X\backslash Y$ and X/Y instead of lump types with (what is conventionally viewed as) morphological, syntactic and semantic information. Another difference is CCG’s (by default limited) support for word order. (In particular, CCG does not handle concatenations (of terms of types) c$(X/Z, Y, Z)$ and c$(Z, Y, X\backslash Z)$, where Y is nonempty[13].) Also, in CCG, NL constructions of sentence level and below can have multiple structures *independently* of (what is traditionally called) interpretational ambiguity.

7.2 Head-Driven Phrase Structure Grammar

It is also useful to compare NLC with a more dedicated syntactic formalism such as Head-Driven Phrase Structure Grammar (HPSG). As a mature formalism that has been implemented for several languages (Pollard and Sag, 1994; DELPH-IN, 2019), HPSG is currently implementation-wise much superior to NLC (which has been implemented only for a fragment of English[14]), so it is appropriate to compare only formalisms. We start with similarities. Both formalisms model parts of semantics, syntax and morphology, but HPSG’s scope is much wider, as it covers also

[11] The latter will depend on your linguistic theory, as different theories posit different categories and members for them.

[12] However, as described below, CCG still features syntactic types.

[13] I am not sure whether such concatenations exist in NLs with fixed word order, but a decision to rule them out by default is arbitrary. However, perhaps it would be feasible to introduce a special rule for accommodating Y in this case.

[14] Structurally, the fragment is quite universal. In fact, with a slightly more general notation one can approximate a Universal Grammar (a statement that will make more sense after reading section 9 and recalling that NL semantics is universal).

lexicon and word and morpheme orders. Both formalisms are compositional in that the meaning of a sentence is given by its constituent structure (Carnie, 2012), but the ways of achieving this are very different: HPSG uses attributes (features) and attribute value matrices while NLC uses types and functions. With this the similarities seem to end. HPSG and NLC are fundamentally different kinds of formalisms — the former features an extensive set of attributes, rules and attribute complexes, while the latter has only three rules (introducing atomic, function application and lump types), leaving the specification of types to the implementor. In a sense, there is little to compare, as HPSG is a full-blown *NL grammar formalism* (that has been extended to cover also selectional restrictions) while NLC is a *generic type system for modeling NL compositionality*. Thus, NLC is a much simpler and more general system, and its implementor has significantly more freedom in NL modeling than an implementor of HPSG.

8 Implementing NLC

My experience of implementing NLC is quite limited, as I have so far tried to implement it only in one programming language and have implemented at best a half of NL in terms of its general (or typological) category structure. Below is a test of an implementation of NLC. The test is by type-checking possible NL(C) expressions. The code (from Supplement B) is generously commented and should be self-explanatory.

```
Check PAST throw john. (* "John threw"
type checks -- but not as a sentence: *)
Fail Check PAST throw john: S. (* "At the hut"
can be the 3rd argument of "throw": *)
Check PAST throw john (-s stone)
(at (the hut)). (*..but not the 2nd one:*)
Fail Check PAST throw john (at (the hut)).
(* "In a hut" cannot be
an argument of "throw": *)
Fail Check PAST throw john
(-s stone) (in (a hut)).
(* ..but can be a sentence modifier: *)
Check in (a hut) (PAST throw john (-s stone)).
(* ..and so can "at every hut" and
sentential adverbs like "however": *)
Check at (every hut) (PAST throw john
(-s stone)): S.
Check however (PAST throw john (-s stone)): S.
(* Connectives cannot range over a
sentential and nominal argument: *)
Fail Check and (PAST throw john (a stone))
john. (* ..but can range over nominal: *)
Check and (every john) (all (the (-s boy))).
(* ..or sentential arguments: *)
Check and (PAST walk (-s boy)
(to (all (-s hut)))) (PAST sleep john).
(* ..(also w/ optional arguments omitted): *)
Check and (PAST walk john) (PAST sleep john).

(* Examples of lump types *)
(* "John" is an XP, proper name, male, in
nominative, singular, a physical entity: *)
Check john: XP0 Phy NOM SG (Pn Ml).
(* ..and a limbed entity: *)
Check john: XP0 Lim NOM SG (Pn Ml).
(* "The entire hut and all Johns" is an XP
and physical entity in nominative: *)
Check and (the (entire hut))
(all (-s john)): XP2 Phy NOM _ _. (* ..or
pseudo-accusative (by zero-derivation): *)
Check and (the (entire hut))
(all (-s john)): XP2 Phy ACC′ _ _.
(* ..and can be made into a sentient
entity in Lax mode only: *)
Check [and (the (entire hut))
(all (-s john))]: XP2 Sen ACC′ _ _.

(* "John threw madly blue stones at the hut
and red limbed boys." has 2 parses: *)
Check madly (PAST throw) john
(blue (-s stone)) (at (and (the hut)
(red [limbed (-s boy)]))): S.
Check PAST throw john
(madly blue (-s stone)) (at (and (the hut)
(red [limbed (-s boy)]))): S.
(* Here we used "[...]" to make a
limbed entity into a physical one. *)

(* We can stack adverbs and adjectives, and
use adjectival and adverbial connectives: *)
Check all ((and madly madly) red (red
[and blue limbed [-s john]])). (* All madly
and madly red, red, blue and limbed Johns *)
```

In Coq, _ is a placeholder for any admissible term or type. A switch in the file the code is taken from allows to choose between Strict and Lax modes, respecting and ignoring selectional restrictions, respectively. The notation [...] interfaces with the current mode. The example omits all technical details like type definitions, etc. These are not instrumental to NLC, as the type system — i.e. one capturing the morphological, syntantic and semantic compositionality of NL with lump types as faithfully as possible — can be implemented in several ways (cf. Supplement A) and different programming languages. The implementation uses only a tiny subset of Coq's features, and its main functionality, theorem proving, is entirely optional here. As I am not at all convinced that Coq is the best language for implementing NLC, I encourage the interested reader to experiment with a programming language of their choice. That being said, statically typed programming languages with a sufficiently complex type system and advanced type inference have some advantages for this kind of work (e.g. in terms of rigor and the similarity of implemented formulas to NL expressions).

9 Implications

The driving force behind NLC has been to correspond to NL as closely as possible. Since ontology (or world knowledge) interfaces with the compositional semantics of NL, it is desirable to formalize some of it in the form of selectional restrictions. We have collapsed syntactic, morphological and semantic compositionality to a single level — to that of the type system. In effect, some types have become syntactic, but the syntax has only two rules: functionality (CTV-Intro) and lumping (LT-Intro). In sum, the paper (and the underlying formalization) have shown that:

(†) A feature of natural language — viz. morphological, syntactic and semantic compositionality — can be reduced to a single level of description.

It is not clear what (†) means, so let us try to explore it further, by (temporarily) assuming that (†) posits a new level of description — call it compositionality — which, moreover, would have to interface with lexical semantics and (what is left of) morphology and syntax. This would be not only theoretically unheard-of (which would be only a mild objection) but would have the undesirable consequence of complicating the general framework of linguistic theory. However, it would have some positive outcomes as well, namely "eliminating" compositional semantics and simplifying morphology and syntax proper. The general theory of natural language would become more complex while three subtheories (morphology, syntax and semantics) would simplify.

Depending on one's outlook on the general theory of natural language, this might seem like a path worth pursuing. However, below I will argue that it is not the only one. The alternative would be to assume that:

(‡) There's nothing "morphological" or "syntactic" about morphological and syntactic compositionality — it is all just semantic compositionality.

Clearly, (†) and (‡) are not mutually exclusive — in fact, (‡) is just a more radical version of (†) (and incidentally, also subsumes (†)). (‡) just conflates the hypothetical new level of description of (†) with compositional semantics. Word and (subword) morpheme order pertain to syntax and morphology, respectively; the compositionality of words, phrases, morphemes and clauses pertains to semantics. As a desirable consequence, we could continue using the existing general theory of natural language with only a few terminological changes. But (how) would (‡) be viable?

A possible justification would make at least two arguments. First, from the theoretical side, morphological, syntactic and semantic compositionality all refer to certain (parts of) knowledge — namely, about morphology, syntax and world, respectively. The only way to have knowledge is by way of meaning, which, given the above, is clearly linguistic. This consideration roots our enterprise in linguistic semantics. Second, from the formalization side, we are using type theory, which is a theory of semantics (broadly defined[15] — cf. (i)). This argument formally corroborates the claim that NLC models only natural language semantics.

Of course, the fact that NL can be modeled this way does not entail that this is the way it works in the brain[16]. So far, our argument has been solely about modeling: It is more parsimonious to model compositionality in a functional type system than e.g. with syntax trees or phrase structure rewrite rules, since the latter cannot, neither separately nor when combined, account for all compositionality. The only advantage of the rewrite rules and syntax trees over the type-theoretical modeling is that they allow, in principle, to capture word order. However, not all syntactic theories support linear order preserving trees (the Chomskian transformational grammar being a case in point — Chomsky (1965, 1981)). Secondly, a word order rule is, differently from compositionality, not a linguistic universal (there are many languages with flexible word order — Dryer (2013)). Incidentally, this also means that *not all natural languages have syntax*.

One thing that seems to emerge from the literature on language processing is the role of syntax as guiding semantic interpretation, or (more figuratively) serving semantics (Kempson et al., 2001; Morrill, 2010; Christiansen and Chater, 2016). Some authors have explicitly argued against syn-

[15] Historically, the semantics of mathematics, more recently also the semantics of programming languages.

[16] Incidentally, there is little sense in trying to make a case of "how language works in the brain", as there is no consensus on this among psycho- and neurolinguists (Chater and Christiansen, 2016).

tax as a separate representational level of linguistic structure (Pulman, 1985; Steedman, 2000). Topographical patterns of brain activation to nouns and verbs are driven not by lexical (grammatical) class but by semantics and word meaning (Moseley and Pulvermüller, 2014). A cognitive architecture with a multi-level (syntactic, semantic, morphological, etc.) NL processing usually requires positing at least as many memory buffers for it (Levinson, 2016), while our short-term memory (obviously recruited in e.g. dialogues) is very limited (Cowan, 2001). These pieces of evidence from language studies also corroborate (‡).

In sum, we hypothesize that combinatorial (im)possibilities in syntax and morphology are better analyzed as belonging to the domain of compositional semantics. Moreover, the conjecture that the traditional boundaries between the levels of description reflect more of a sociological (a division of labor among linguists) than a linguistic fact is not too bold.

If the hypothesis is correct, nothing remains in syntax except word order[17]. Since word order can label (now already semantic) constituents, as in $John_{\text{SUB}}$ $loves$ $Mary_{\text{OBJ}}$, a limited form of syntax-semantics interface is also expected. This accords with the view of the function of syntax as serving semantics, viz. in interpretation disambiguation, which is the primary function of word order constraints (as witnessed in the example above). Syntax-semantics interface is also an appropriate level for phenomena like anaphora and ellipsis. Likewise, morphology proper retains only (subword) morpheme order and fusion, while morphophonological (i.e. morphology-phonology interface) phenomena like e.g. sandhi must also be accounted for. As a result, syntax and morphology emerge as by-products of a contingent conformation to the serial channel over which language is processed.

10 Conclusion

If we interpret proper arguments as nullary functions, all NL expressions up to the sentence level (i.e. morphemes, words, phrases, clauses and sentences) can be interpreted as functions and function applications. The paper presents NLC, a generic functional type system (i.e. one consisting primarily of functions and their applications). NLC is generic in at least two respects: 1. It is applicable to all NLs, and 2. It allows for generic modeling of morphological, syntactic and semantic compositionality. Besides functions, applications and their types, NLC features polymorphic and lump types. The latter are compound types satisfying LT-Intro. Compound types are types which are syntactic compounds of multiple types or their terms (Σ-, Π- and Cartesian product types are examples of compound types). At its core, NLC is a simple system for an integrated modeling of the morphological, syntactic and semantic compositionality of NL with lump types.

The paper also presents an implementation of NLC in Coq (which, unfortunately, is not quite as simple, which may be Coq's fault). The main goal of the implementation was to formalize a reasonably diverse fragment of NL in NLC, with formalized NLC expressions type checking and failing to type check in exactly the same ways that NL expressions pass and fail their acceptability tests. Aside from this goal's feasibility, the implementation shows several things: (1) the viability and simplicity of NLC for modeling NL compositionality, (2) the utility of lump and polymorphic types in NL modeling, and most importantly, (3) the possibility of reducing morphological, syntactic and semantic compositionality to a single level of description. In discussion we have tried to identify this level as semantic compositionality — an interpretation which, besides being supported by results from language processing (Pulman, 1985; Steedman, 2000; Moseley and Pulvermüller, 2014), has interesting implications on NL structure and modeling. In particular, it may reduce syntax and morphology to word and morpheme orders, respectively (with the syntax-semantics and phonology-morphology interfaces reducing correspondingly), with NL architecture taking on a rather different look. This has also implications on linguistic typology, as syntax would, much like morphology before it (Muansuwan, 2002; Grandi and Montermini, 2005; Klamer, 2005), cease to be a logically necessary component of NL.

Acknowledgments

I thank Jason Gross, Hendrik Luuk, Erik Palmgren and Enrico Tassi for their advice. This work has been supported by IUT20-56 and European Regional Development Fund through CEES.

[17] If we do not posit lexicon as a separate level, lexical categories also pertain to morphology or syntax.

References

Nicholas Asher. 2014. Selectional restrictions, types and categories. *Journal of Applied Logic* 12(1):75–87. https://doi.org/10.1016/j.jal.2013.08.002.

Daisuke Bekki and Nicholas Asher. 2013. Logical polysemy and subtyping. In Yoichi Motomura, Alastair Butler, and Daisuke Bekki, editors, *New Frontiers in Artificial Intelligence*, Springer, Berlin, Heidelberg, pages 17–24.

Andrew Carnie. 2012. *Syntax: A Generative Introduction*. Wiley-Blackwell, Malden, MA, 3rd edition. https://www.wiley.com/en-ee/Syntax:+A+Generative+Introduction,+3rd+Edition-p-9780470655313.

Nick Chater and Morten H. Christiansen. 2016. Squeezing through the Now-or-Never bottleneck: reconnecting language processing, acquisition, change, and structure. *Behavioral and Brain Sciences* 39:e91. https://doi.org/10.1017/S0140525X15001235.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014a. Natural language inference in Coq. *Journal of Logic, Language and Information* 23(4):441–480. https://doi.org/10.1007/s10849-014-9208-x.

Stergios Chatzikyriakidis and Zhaohui Luo. 2014b. Natural language reasoning using proof-assistant technology: Rich typing and beyond. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 37–45. http://www.aclweb.org/anthology/W14-1405.

Stergios Chatzikyriakidis and Zhaohui Luo. 2015. Individuation criteria, dot-types and copredication: A view from modern type theories. In *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*. Association for Computational Linguistics, pages 39–50. https://doi.org/10.3115/v1/W15-2304.

Stergios Chatzikyriakidis and Zhaohui Luo. 2016. Proof assistants for natural language semantics. In Maxime Amblard, Philippe de Groote, Sylvain Pogodalla, and Christian Retoré, editors, *Logical Aspects of Computational Linguistics. Celebrating 20 Years of LACL (1996–2016)*. Springer, Berlin, Heidelberg, pages 85–98. http://www.cs.rhul.ac.uk/ zhaohui/LACL16PA.pdf.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge. http://www.amazon.com/Aspects-Theory-Syntax-Noam-Chomsky/dp/0262530074.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

Morten H. Christiansen and Nick Chater. 2016. The Now-or-Never bottleneck: a fundamental constraint on language. *Behavioral and Brain Sciences* 39:e62. https://doi.org/10.1017/S0140525X1500031X.

Robert L. Constable. 2003. Recent results in type theory and their relationship to Automath. In Fairouz D. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, Springer Netherlands, Dordrecht, pages 37–48.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation* 15(2):99–112. https://doi.org/10.1093/logcom/exi004.

Nelson Cowan. 2001. The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences* 24(1):87–114; discussion 114–85. https://doi.org/10.1017/S0140525X01003922.

DELPH-IN. 2019. The DELPH-IN collaboration. Accessed 18.03.2019. http://www.delph-in.net.

Matthew S. Dryer. 2013. Order of Subject, Object and Verb. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*, Max Planck Institute for Evolutionary Anthropology, Leipzig. http://wals.info/chapter/81.

Nicola Grandi and Fabio Montermini. 2005. Prefix-suffix neutrality in evaluative morphology. In Geert Booij, Emiliano Guevara, Angela Ralli, Salvatore Sgroi, and Sergio Scalise, editors, *On-line Proceedings of the Fourth Mediterranean Morphology Meeting (MMM4), Catania, 21-23 September 2003*. Università degli Studi di Bologna. https://geertbooij.files.wordpress.com/2014/02/mmm4-proceedings.pdf.

N. Gruzitis and D. Dannélls. 2017. A multilingual FrameNet-based grammar and lexicon for controlled natural language. *Lang Resources & Evaluation* 51(1):37–66. https://doi.org/10.1007/s10579-015-9321-8.

Normunds Gruzitis, Peteris Paikens, and Guntis Barzdins. 2012. FrameNet resource grammar library for GF. In Tobias Kuhn and Norbert E. Fuchs, editors, *Controlled Natural Language*, Springer, Berlin, Heidelberg, pages 121–137.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell, Oxford.

Marian Klamer. 2005. Explaining structural and semantic asymmetries in morphological typology. In Geert Booij, Emiliano Guevara, Angela Ralli, Salvatore Sgroi, and Sergio Scalise, editors, *On-line Proceedings of the Fourth Mediterranean Morphology Meeting (MMM4), Catania, 21-23 September 2003*. Università degli Studi di Bologna. https://geertbooij.files.wordpress.com/2014/02/mmm4-proceedings.pdf.

Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly* 65(3):154–170.

Stephen C. Levinson. 2016. “Process and perish” or multiple buffers with push-down stacks? *Behavioral and Brain Sciences* 39:e81. https://doi.org/10.1017/S0140525X15000862.

Zhaohui Luo. 2010. Type-theoretical semantics with coercive subtyping. In *Semantics and Linguistic Theory*. Vancouver, volume 20, pages 38–56.

Zhaohui Luo. 2011. Contextual analysis of word meanings in type-theoretical semantics. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *Logical Aspects of Computational Linguistics*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 159–174.

Zhaohui Luo. 2014. Formal semantics in modern type theories: is it model-theoretic, proof-theoretic, or both? In Nicholas Asher and Sergei Soloviev, editors, *Logical Aspects of Computational Linguistics 2014 (LACL 2014)*, Springer, Berlin, Heidelberg, number 8535 in LNCS, pages 177–188.

Erkki Luuk. 2010. Nouns, verbs and flexibles: implications for typologies of word classes. *Language Sciences* 32(3):349–365. https://doi.org/10.1016/j.langsci.2009.02.001.

Richard Montague. 2002. The proper treatment of quantification in ordinary English. In Paul Portner and Barbara H. Partee, editors, *Formal Semantics: The Essential Readings*, Blackwell, Oxford, pages 17–34.

Glyn Morrill. 2010. *Categorial grammar: Logical syntax, semantics, and processing*. Oxford University Press, Oxford.

Rachel L. Moseley and Friedemann Pulvermüller. 2014. Nouns, verbs, objects, actions, and abstractions: Local fMRI activity indexes semantics, not lexical categories. *Brain and Language* 132:28 – 42. https://doi.org/10.1016/j.bandl.2014.03.001.

Nuttanart Muansuwan. 2002. *Verb Complexes in Thai*. Ph.D. thesis, University at Buffalo, The State University of New York. https://arts-sciences.buffalo.edu/content/dam/arts-sciences/linguistics/AlumniDissertations/Muansuwan dissertation.pdf.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Stephen G. Pulman. 1985. A parser that doesn’t. In *Proceedings of the 2nd European Meeting of the Association for Computational Linguistics, Geneva: ACL*. pages 128–135. https://www.aclweb.org/anthology/E85-1019.

Aarne Ranta. 1994. *Type-theoretical grammar*. Clarendon Press, Oxford; New York.

Aarne Ranta. 2004. Grammatical Framework: a type-theoretical grammar formalism. *The Journal of Functional Programming* 14(2):145–189. https://doi.org/10.1017/S0956796803004738.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

v-trel: Vocabulary Trainer for Tracing Word Relations - An Implicit Crowdsourcing Approach

Verena Lyding
Institute for Applied Linguistics,
Eurac Research Bolzano/Bozen
verena.lyding@eurac.edu

Christos T. Rodosthenous
Open University of Cyprus
christos.rodosthenous@ouc.ac.cy

Federico Sangati
Orientale University of Naples
federico.sangati@gmail.com

Umair ul Hassan
Insight Centre for Data Analytics,
National University of Ireland Galway
umair.ulhassan@insight-centre.org

Lionel Nicolas
Institute for Applied Linguistics,
Eurac Research Bolzano/Bozen
lionel.nicolas@eurac.edu

Alexander König
Institute for Applied Linguistics,
Eurac Research Bolzano/Bozen
alexander.koenig@eurac.edu

Jolita Horbacauskiene
Kaunas University of Technology
jolita.horbacauskiene@ktu.lt

Anisia Katinskaia
Department of Computer Science,
University of Helsinki
anisia.katinskaia@helsinki.fi

Abstract

In this paper, we present our work on developing a vocabulary trainer that uses exercises generated from language resources such as ConceptNet and crowdsources the responses of the learners to enrich the language resource. We performed an empirical evaluation of our approach with 60 non-native speakers over two days, which shows that new entries to expand ConceptNet can efficiently be gathered through vocabulary exercises on word relations.

We also report on the feedback gathered from the users and an expert from language teaching, and discuss the potential of the vocabulary trainer application from the user and language learner perspective. The feedback suggests that v-trel has educational potential, while in its current state some shortcomings could be identified.

1 Introduction

One of the major challenges for the NLP community is the continuing lack of comprehensive and high-quality language resources (LRs) for most languages. While LR creation can partly be approached in an automatic fashion (e.g., by harvesting a vocabulary from existing corpora), it often requires human intervention to reach a high level of quality and coverage. Crowdsourcing (Howe, 2006) is one promising approach that can be leveraged for the task of LR creation. However, any crowdsourcing ambition is faced with the challenge of attracting and retaining crowdworkers and with safeguarding the quality of results produced by the crowd (Daniel et al., 2018). EnetCollect[1] (Lyding et al., 2018; Nicolas et al., 2018) aims at exploring a solution to this challenge by combining the activities performed in language learning with approaches for crowdsourcing language-related datasets. Thus exploring a new path to address the NLP bottleneck of high-quality language resource creation.

In this paper, we present an application for vocabulary training that is specifically designed to crowdsource learners' answers to improve LRs like concept networks. The application builds on top of an architecture for crowdsourcing of language resources (Rodosthenous et al., 2019), which instantiates one of the core ideas of enetCollect: the *implicit crowdsourcing paradigm* (Section 2). Accordingly, the vocabulary trainer aims at a two-fold purpose: serving automatically

[1] COST Action enetCollect: European Network for Combining Language Learning and Crowdsourcing Techniques, http://enetcollect.eurac.edu/

Proceedings of Recent Advances in Natural Language Processing, pages 674–683,
Varna, Bulgaria, Sep 2–4 2019.

generated vocabulary exercises and gaining continuous input from the learners to improve LRs.

In the remainder of the paper, we first introduce the vocabulary trainer *v-trel* and describe its different modules and their technical implementation (Section 2). We describe how v-trel instantiates a generic prototype architecture for crowdsourcing language resources (Rodosthenous et al., 2019) and discuss technical decisions taken during the implementation process. Second, we describe the first experiment that has been carried out with a small crowd of advanced language learners[2] (Section 3) and discuss the results and their implications on the potential of the proposed approach. Third, we point out and discuss related work relevant to the presented approach (Section 4). Finally, we sum up preliminary conclusions and indicate directions for future work (Section 5).

2 Vocabulary Trainer

The vocabulary trainer builds on top of a prototype architecture for crowdsourcing language resources (Rodosthenous et al., 2019). It implements an *implicit crowdsourcing paradigm* which follows the idea that if a language resource (e.g., a lexicon) can be used to generate language learning exercises, then the answers of learners to these exercises can be used to improve the resource, which in turn will improve the quality and versatility of the exercises generated (Rodosthenous et al., 2019).[3]

2.1 Motivation and Design

The vocabulary trainer delivers interactive vocabulary exercises for learning word senses. The learner is asked to input words which are related to a given word by the semantic relation `RelatedTo`, and will in the future be extended to other relations such as `PartOf`, `AtLocation` etc. The learner input is collected and evaluated to enhance the LRs that it is generated from.

From the language learning perspective, vocabulary exercises play an important role in language learning (Nation and Hunston, 2013). Hulstijn (2013) notes that every word in a mental lexicon has formal as well as semantic associative features. Depending on the learner's level of language, vocabulary building may encompass single lexical words with a specific meaning or formulaic sequences / lexical chunks where pedagogical relations are structured by a particular object representation or a part of a particular object (Aldabe et al., 2015). As noted by Schmitt (2013), vocabulary learning is a complex phenomenon that may be explored not only from the aspects of form, meaning and usage but also from a representation of different meanings in different contexts.

Vocabulary exercises based on words' semantic relations are considered to be effective activities. Rosenbaum (2001) shows that background knowledge, context and morphology are essential in vocabulary instruction to enable the learner to understand and disambiguate word meanings effectively. The richness of acquired vocabulary depends not only on the number of learned lexical items but also on the ability to connect and share semantic networks of similar concepts. Hadley et al. (2018) argue that *"word learning is not simply the process by which isolated object-label associations are added to the mental lexicon one by one but also involves the learning of interrelated clusters of concepts, in which the knowledge of one concept supports the learning of another"* (p. 42).

From the crowdsourcing perspective, learners are used as crowdworkers to enhance the LR underlying the vocabulary trainer, namely the common sense ontology *ConceptNet*[4] (see Section 2.2). While using the vocabulary trainer for learning word senses the learners are providing their knowledge of related words which is collected and evaluated in order to validate and enhance the LR.

The vocabulary trainer is composed of four modules which are presented in the following subsections: 1) The *exercise generation module* that retrieves words from ConceptNet and generates exercise content (Section 2.2), 2) the *exercise and result storage dispatcher* that ingests the previously created exercise content (Section 2.3), dispatches it to the various learner interfaces and handles the responses from the learners, 3) the *evaluation module* that is responsible for evaluating if learners' contributions are fit for expanding the language resource and assign points to each learner according to the response given (Section 2.4), and 4) the *user interaction module*, where users are presented with the exercises and submit their responses (Section 2.5). In Figure 1, a high-

[2] For the initial experiment we involved proficient non-native speakers of English (see Section 3.2 for details).

[3] The related article discusses the paradigm in more detail and points out strategies to counter the risk of collecting wrong or low-quality data from non-proficient learners.

[4] http://conceptnet.io/

level diagram of the vocabulary trainer's architecture is depicted along with the exchange of data between the core modules of the system.

Interested readers are invited to also browse the project repository [5].

2.2 The Exercise Generation Module

The exercise generation module is responsible for content retrieval from language resources like ConceptNet (Speer et al., 2017) which is a large semantic network that describes general human knowledge and how it is expressed in natural language. ConceptNet provides a large set of background knowledge for different terms that not only describes them but also connects them with other terms using relations such as `RelatedTo`, `AtLocation`, `PartOf`, `IsA`, etc.

In its current version, the exercise generation module is able to search ConceptNet for terms connected with the relation `RelatedTo`, `AtLocation` and `PartOf` and generate exercises using a template such as "Name one thing that is <RELATION> <TERM>". For instance, if a search for knowledge that is `RelatedTo` the term "dog" is initiated, ConceptNet will yield results such as "bark", "pet", "animal", etc. The generation module processes these by removing stopwords, duplicates and terms that have a language other than English. The relevant information is stored in a database to be processed later by other modules along with the exercise data.

Searching ConceptNet is a straightforward process since the knowledge base offers a number of APIs to query it. In our case we use the typical search query[6] to get `relatedTo` terms. The search term is provided in canonical form in ConceptNet, e.g., `/c/en/cat` and offset is the number of records to skip and show the next, as ConceptNet API has a limit of 1000 results per call.

An example of a generated exercise is "Name one thing that is related to dog", where the learner is expected to enter a word that exists in the results retrieved from the knowledge base. In cases where new words are entered by the learner, the evaluation module handles whether they should be added to the knowledge base or not while a specific user feedback strategy is used to account for the unknown correctness of the new answers (see Section 2.4).

2.3 The Exercise and Results Storage Dispatcher Module

Transactions between modules are handled using web services through API calls. Data are presented in a JSON[7] format that can be consumed by any programmatically created interface. Specification of the API is available at the project repository using the Swagger[8] Opensource API management tool. This abstraction layer allows the exploitation of the system from various interfaces without developers having to know its underlying functionality. Currently, the system offers web services for registering users, retrieving exercises from the exercise generation module, checking learners' contributions, assigning points/awards, storing presented hints and showing a leaderboard.

For the latter part, the evaluation module is employed. The outcome of the evaluation module is used to update both the learner's points and awards and the knowledge base list of answers for that specific exercise.

The dispatcher is also directly connected with the database for storing/retrieving data in/from tables and provide another abstraction layer for information handling workflows.

2.4 The Evaluation Module

The evaluation module processes the learner's answers in order to both update the knowledge base with new words and to assign points and award badges to the learners, which are transformed into feedback messages and leaderboards in the user interface.

The evaluation module operates on pairs of exercise and result (see Figure 1) produced each time a user completes one exercise. It checks whether the user's answers are known or new to the knowledge base and evaluates if new answers are valid candidates to enhance it. According to the evaluation points received, badges are assigned to the learners.

More specifically, the evaluation module checks for each user's answer, whether it is part of the results set for that exercise or whether the answer is new. If the answer is part of the knowledge base, the user receives one point. If the answer is new, the user receives *potential* points, and the answer is stored as a candidate answer for that exercise together with the user id. The feedback message

[5] `https://gitlab.com/crowdfest_task3`

[6] `http://api.conceptnet.io<TERM>?other=/c/en&limit=1000&offset=<OFFSET>`

[7] `https://www.json.org/`

[8] `https://swagger.io/`

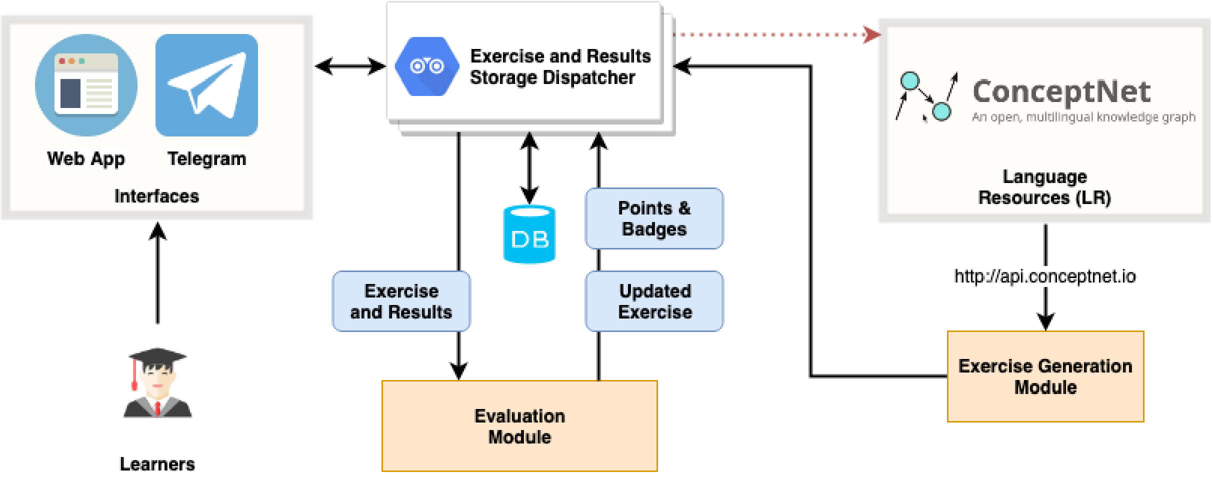


Figure 1: An overview of the vocabulary trainer architecture, depicting the main modules of the architecture, the data interchange between them and user interaction at the interface level.

to the learner informs him that the answer is new and that he is gaining *potential* points, which may transform into actual points if the answer is validated by several users over time.

Every time a predetermined number K of candidate answers has been accumulated, the evaluation process is triggered:

- All new answers are ranked by their occurrence frequency (i.e., how many mentions).
- The top-ranked answer is selected (given that it was mentioned at least N times).
- All learners that gave the selected answer get informed that it was a correct answer and receive two points (transforming *potential* points into actual points).
- In addition, the learner who was the first to give the selected answer receives a badge.
- The knowledge base is updated with the new word for that exercise.
- All occurrences of the selected word are removed from the candidate list.

2.5 User Interaction Module and Prototypical User Interfaces (UI)

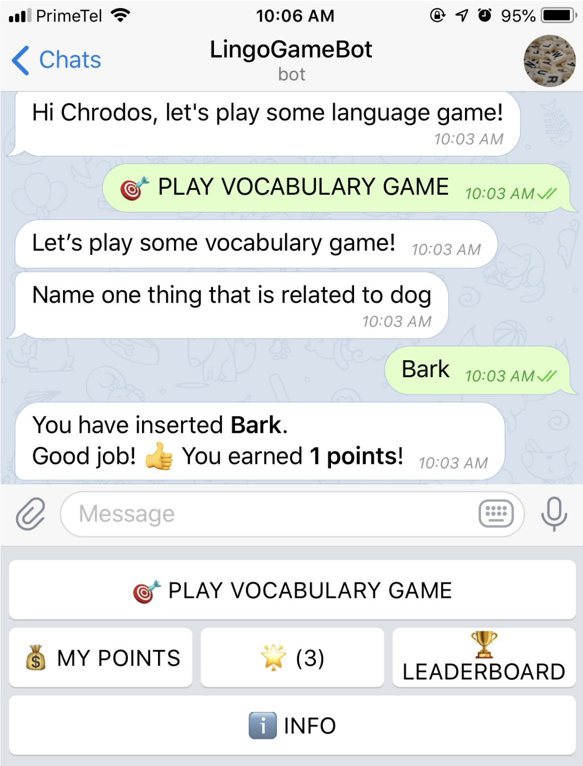
In the current version of the vocabulary trainer, two interfaces are implemented: a chatbot on Telegram and a web application. Both interfaces allow the learner to receive and complete exercises on the `RelatedTo` relation. Learners get an immediate response back on the correctness of their answer and when this is not possible (i.e., in cases of unknown answers), they first receive potential points and get notified asynchronously, once the unknown answer was confirmed by other learners.

Currently, the vocabulary trainer forces the user to input a word in order to move to the next exercise. In order to support the learner in case he runs out of ideas for related words on a given exercise, the “I Don’t Know” feature provides functionality for requesting a “hint”. The hint provides the user with a correct related word, which is taken from the knowledge base. After reading the hint, the learner is free to input the hint word or any other word that he/she deems fitting. Whenever a learner uses that feature, there is an underlying mechanism that stores the presented hint in the database.

Within both interfaces, learners can see their points and badges gained and for the Telegram chatbot, they can also access a leaderboard.

Chatbot interface The chatbot interface was implemented on Telegram, a very versatile messaging system. Telegram is available both as a native application for mobile phones and desktop computers for all operating systems and as a web application. It enables the implementation of chatbots via the Telegram Bot APIs[9]. In the implemented chatbot[10], users interact with the system via a standard dialogue chat interface (see Figure 2). Apart from textual input, the interface provides a button area that changes during the dialogue flow to simplify the interaction.

[9] `https://core.telegram.org/bots/api`
[10] `https://t.me/LingoGame_bot`



Hi Chrodos, let's play some language game! 10:03 AM

🎯 PLAY VOCABULARY GAME 10:03 AM

Let's play some vocabulary game! 10:03 AM

Name one thing that is related to dog 10:03 AM

Bark 10:03 AM

You have inserted **Bark**.
Good job! 👍 You earned **1 points**! 10:03 AM

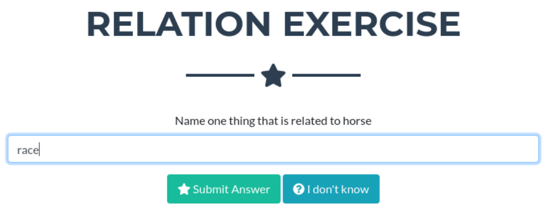
Figure 2: A screenshot from the chatbot interface, where a user is interacting with the system.

Web interface The web interface[11] (see Figure 3) currently offers three exercise types for learners to practice with. It was implemented using a free bootstrap template[12] to ensure a stable interface that works well both on computers and mobile devices.

3 Empirical Evaluation

To evaluate the potential of the v-trel architecture for the purpose of (1) gathering commonsense knowledge and populating the language resource used to generate the exercises, and (2) delivering a meaningful application for vocabulary learning, we designed an experiment as follows.

For the initial experiment, the focus was put on objective (1), evaluating the quality of collected data. However, a feedback questionnaire and the manual analysis of the gathered data by an expert in language teaching served to gain first insights also related to (2), the educational value of such a vocabulary trainer, which need to be expanded on in a follow-up study (see Section 5).



Name one thing that is related to horse

race

Figure 3: A screenshot from the web interface.

Term	`RelatedTo` term	#terms
cat	animal, dog, house. . .	94
dog	friend, puppy, bone. . .	135
bird	chicken, chick, canary. . .	219
cow	sacred, animal, steak. . .	96
fish	water, creature, lure. . .	221

Table 1: An example of the terms retrieved to generate the experiment exercises. The subject is `RelatedTo` the object. In the last column the number of filtered terms is depicted.

3.1 Setup of the Experiment

First we generated 26 exercises using the `RelatedTo` relation from ConceptNet and 26 terms (see Table 1) that fall under the A1 and A2 language learning level. For each of these terms, we acquired more than 20 terms that are related to them. We used the conceptnet.io API to retrieve all terms connected with the `RelatedTo` relation where the language is set to English. We filtered out terms that had less than 20 `RelatedTo` terms in order for the learners to have a plethora of possible answers for the "I Don't Know" feature.

We also implemented a recording mechanism that captures the answers presented to the learner when clicking the "I Don't Know" button for a specific exercise. This served the purpose for us to analyze how the learner used this feature and reacted to the words presented to them while contributing and completing exercises.

For the experiment the evaluation parameters were set to a threshold of K=*5* candidate answers to trigger the evaluation and a minimum limit of N=*2* words for including the candidate into the knowledge base was applied.

3.2 Implementation of the Experiment

The initial experiment was conducted with people from our peer group, about 60 non-native speakers

[11] `http://cognition-srv1.ouc.ac.cy/vtrel/`

[12] `https://startbootstrap.com/themes/freelancer/`

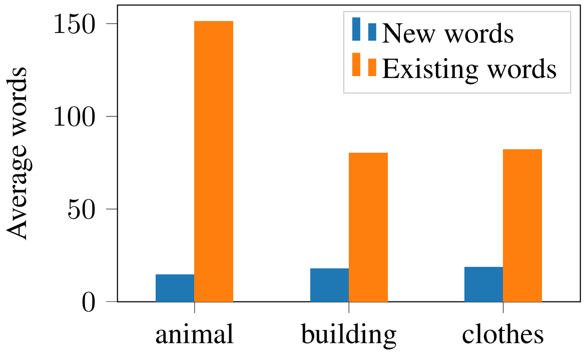


Figure 4: The average number of new and existing words per exercise for each category.

of English, with a high proficiency level.[13] Each of these users received an email with a link to both the Web interface and the Telegram chatbot and was asked to try any of the two for more than ten minutes within a period of two days. At the end of this period, each user received a link to an online questionnaire[14] to provide feedback on both the interfaces and the presented questions.

To summarize, during the 2 days period we managed to gather 4533 contributions to 26 exercises presented to the user in random order. The contributions were collected in 44 distinct user sessions, of which 17 were on the Telegram chatbot and 27 on the Web interface. Presumably, the sessions mostly relate to unique users, although we know of at least one user, who accessed both interfaces (see Section 3.4).[15] We also captured 683 possible answers presented to learners through the "I Don't Know" feature. Out of the 4533 contributions, 449 new words were crowdsourced from the users based on the evaluation mechanism with the parameter settings described above (see Section 2.4 and 3.1). In terms of questionnaire feedback, we gathered 17 fully-completed and 17 partially-completed responses.

3.3 Results Analysis

Characteristics of new words. Overall, the experiment allowed to gather 449 new words, divided by exercise category as follows: 119 words on animals, 168 words on clothing, and 162 words on buildings. The lower number of new words for the category animal relates to the higher number of existing words in the knowledge base for that category (see Figure 4). For example for the term "cow", 15 new words were gathered. Table 2 shows the 8 new words that were named three or more times, while seven words[16] met the minimum threshold of two mentions. This shows that ConceptNet has empty spots even for basic vocabulary like "grass" or "farmer", which could be gathered through the learners. Also, it shows that learners were able to propose advanced vocabulary such as "ruminate" or "pasture".

New word	Frequency	Level
grass	15	basic
calf	6	advanced
meat	5	basic
cowboy	4	basic
farmer	4	basic
herd	4	advanced
horn	3	moderate
pasture	3	advanced

Table 2: Top new words for the term "cow", their frequency of mentions and proficiency levels.

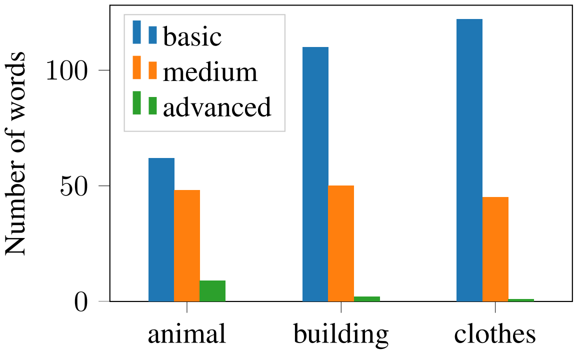


Figure 5: Level of proficiency of new words by exercise category.

A manual analysis of the proficiency level of all new words was carried out by an expert from language teaching. It showed that the vast majority of new words is part of basic vocabulary (65% of all added words), while 32% are of moderate level and only 3% belong to advanced vocabulary.

[13] We are aware that involving speakers with a high English proficiency implicates that the crowd does not represent genuine language learners. By being composed of non-native speakers, we however assume them to resemble advanced learners to a degree that allows to draw meaningful conclusions for the scope of this first evaluation.

[14] LimeSurvey, a Free Opensource online tool was locally deployed: https://www.limesurvey.org

[15] For simplicity, we refer to user sessions as *users*.

[16] New words for term "cow" with two mentions only: *bell, burger, methane, ox, ruminate, sheep* and *veal*

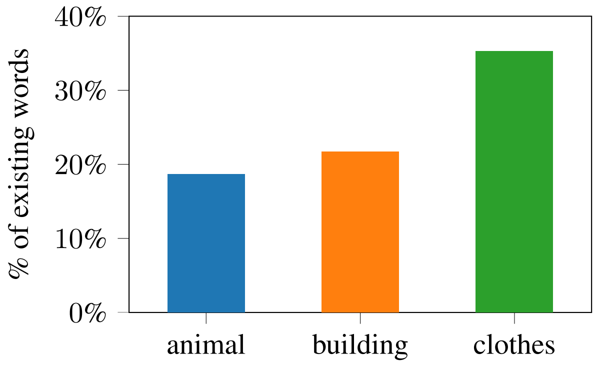


Figure 6: Average percentage of existing words, provided by users, per exercise for each category.

Interestingly, for the animal category the ratio between basic and moderate vocabulary is far more balanced than for the other two categories, also more advanced vocabulary is found (see Figure 5). This could be explained by the higher number of existing words in the knowledge base for the category animal. Most basic words are probably already part of ConceptNet, which implies that new words necessarily need to be more advanced.

Figure 6 shows the average percentage of existing words (per exercise) that were entered by the learners, divided by category. It shows that learners named less than 40% of the words present in ConceptNet[17], while they can still gain knowledge on more than 60% of the words, e.g. by requesting hints through the "I don't know" feature.

Responses to "I Don't Know" hints. Overall, 683 hints were provided to the learners by means of the "I Don't Know" button. In response, 365 words were entered by the learners of which 331 were a direct repetition of the hint word. The lower number of entered words in relation to hints is due to multiple "I Don't Know" clicks before entering a word (up to 17 clicks in a row).

In 34 cases, the word entered in response to the hint(s) was different from the hint(s), which indicates that the hint activated the learner's knowledge on related words. A look at the words shows that:

- 2 times a variation of the lemma was entered
- 11 times an analogy of the hint was entered
- 16 times a different word class was entered

Existing relations in ConceptNet. To get a better idea of the type and quality of words learners contributed regarding our initial LR, i.e., ConceptNet, we queried what other relations might exists in ConceptNet between the search word used for generating the exercise and the contributed words other than the `RelatedTo` relation. From the 26 exercises learners contributed, on average 14.15 words have no other direct relation (not bidirectional) with the original word from ConceptNet, i.e., <SUBJECT> <RELATION> <CONTRIBUTED_WORD>. When searching also for bidirectional relations, 100% of contributed words have such a relation between them in ConceptNet, including `RelatedTo`, which can be taken as indication for the appropriateness of the words added by learners. On average 4.54 new relations (other than direct `RelatedTo`) were identified between the contributed terms and the subject used to generate the exercise.

3.4 Feedback Questionnaire

After the experiment, a feedback questionnaire with six items was sent to all participants:

- Level of English: [A1/A2; B1/B2; C1/C2]
- Interface used: [Chatbot; Web; both]
- What did you notice regarding the UI? [open]
- What did you think of the questions? [open]
- What did you like about this approach to a vocabulary exercise? [open]
- Any other comments? [open]

Out of 34 users, 22 completed the closed questions and 9 to 14 also responded to the open questions; 18 of 22 respondents indicated an advanced level of English; 9 users used the Chatbot, 12 used the Web interface, and one user used both interfaces.

User interface. 9 respondents perceived the interface as clear, easy and pleasant to use, while 9 users criticized unclear navigation and pop-ups interrupting the workflow.

Questions. 9 users remarked that words repeated too often and that the phrasing of the question "name one thing" can be misleading in terms of which word class is requested (3 respondents).

Approach to vocabulary exercise. 12 users evaluated positively the interactivity and simplicity of the exercise, the opportunity to learn new

[17] The lowest coverage were found for category animal, the category with the biggest set of available words.

words by means of the “I don’t know” function, and its effect to reactivate words and to incentivize brainstorming. Still, five users suggested that it is no real vocabulary exercise, that it would be difficult for beginners and that the processing of the answers and assignment of points/feedback was unclear, too open-ended, and lacking negative feedback.

Other comments. The criticism about the way to award points and the overall educative value were stressed further.

3.5 Discussion of the Experiment Results

The analysis of new words shows that our approach is promising for extending ConceptNet with meaningful words, in particular the more advanced level of new words added for the category animal indicates that relevant new terms can be gathered in a progressive fashion (e.g., basic vocabulary is added first). Given that this first experiment was carried out with advanced learners it needs to be evaluated to which extent similar positive results can be achieved with beginner and intermediate learners. Also, the analysis indicates that ConceptNet is ample enough to propose a wide set of new vocabulary to the user (in average more than 60% of the words per exercise). Furthermore, results suggest that this approach can also identify new relations between terms in ConceptNet. Learners managed to reproduce what was already coded in a different manner in the resource and thus improved its completeness.

User feedback verifies that the vocabulary trainer as a User Interface can be improved but the idea behind it is interesting and can be applied in different language learning scenarios.

4 Related Work

Approaches for crowdsourcing language resources can be divided into two broad categories: 1) implicit crowdsourcing, i.e., users carry out any activity of their interest while their data is crowdsourced as a byproduct, and 2) explicit crowdsourcing, i.e., the crowd is actively engaging.

The Duolingo platform (von Ahn, 2013) presents a most similar approach to our work, based on language learning, as it generates language exercises, allowing the crowdsourcing of language-related data (i.e., translations). Other research work based on implicit crowdsourcing has utilized the Games With A Purpose (GWAP) approach (von Ahn and Dabbish, 2008). Among GWAP, in particular the JeuxDeMots game by Lafourcade (2007) shows several similarities to our approach. The game is designed for constructing lexical data in French in a playful way. In order to gain points two players have to agree on words related to given terms. Also, the more recent TileAttack game by Madge et al. (2017) builds on player agreements to gather annotations for text segmentation tasks. In addition, Rodosthenous and Michael (2016) developed a platform that combined automated reasoning with games for acquisition of knowledge rules. Moreover, in work of Guillaume et al. (2016) a game titled ZombiLingo was developed, for annotating dependencies in French data. In work of Chamberlain et al. (2008), the Phrase Detectives game is presented, where players contribute relationships between words and phrases, aiming to create a resource that is rich in linguistic information. Yet another indirect form of crowdsourcing has used large collaborative knowledge bases like Wikipedia to create multilingual resources such as YAGO3 (Mahdisoltani et al., 2015) and DBpedia (Lehmann et al., 2015). Recently, Meurers et al. (2019) have proposed a web-based workbook, which can be integrated into classroom curricula and offers instructive feedback to students while also gathering data on learning processes for Second Language Acquistion (SLA) research.

Research works with explicit crowdsourcing often employ Amazon Mechanical Turk to collect data. For instance, Biemann (2013) created the Turk Bootstrap Word Sense Inventory of frequently used nouns in English and Ganbold et al. (2018) localized the WordNet in the Mongolian language. Related to SLA, MacWhinney (2017) proposes a collaborative platform for collecting and sharing learner data from corpora, online tutors, and Web-based experimentation.

Our research presents an implicit crowdsourcing approach implemented as a vocabulary learning application with open-ended questions for language resource augmentation using multiple user interaction methods (i.e., chatbots and web apps).

5 Conclusion and Future Work

In this paper, we presented the v-trel vocabulary trainer for crowdsourcing language resources and delivering exercises to language learners. V-trel can be accessed through two interfaces, a Telegram chatbot and a Web application.

Moreover, we presented the results of an empirical evaluation and a user satisfaction survey for the vocabulary trainer and provide a relevant discussion of these results. Feedback from users and the analysis of the contributed words are taken into account for updating v-trel with new features such as a more attractive interface, and new exercise types[18]. In addition, we aim at including links to pictures and definitions or examples of use of the "hint" words, to support the learner not only in refreshing their existing vocabulary, but also to acquire new words. These are first steps to strengthen the learning effect of the tool, while in the midterm we foresee to intensify further the collaboration with language teaching experts in order to tailor the offered exercises more closely to specific learning goals.

Also, in future work we aim at implementing more strategies for safeguarding the quality of the acquired data, e.g., control mechanisms for active misuse, further evaluation strategies for new words, which could also involve dynamic retrieval of knowledge from ConceptNet or evaluation cycles re-proposing new words in new exercises, and strengthened gamification elements. In addition, we intend to evaluate the crowdsourced words and their difficulty levels in relation to established reference data such as the *English Vocabulary Profile*[19] or similar resources.

As proposed in the feedback questionnaire, the educational value needs to be validated and improved further. Accordingly, a follow-up user study with focus on the educational aspect is foreseen and will be designed and carried out with authentic learners of different proficiency levels.

Last but not least, the vocabulary trainer will be integrated into the setup of the Revita online system for language learning (Katinskaia et al., 2018) to reach a larger audience. In particular, it can be implemented as a part of a testing mode where crowdsourced questions do not influence the learner's final score. We will also work towards integrating v-trel into Games With A Purpose. Previous work of Rodosthenous and Michael (2016, 2014) suggests that crowdsourcing and GWAPs, in particular, can be used to gather background knowledge

[18] E.g., various new types of vocabulary exercises such as "fill the gap", or "select all verbs among the given words"

[19] `https://www.englishprofile.org/wordlists`

Acknowledgments

This article is based upon work from COST Action enetCollect (CA16105), supported by COST (European Cooperation in Science and Technology). The work presented in this paper was started during the Crowdfest meeting organized by the Action in January 2019 in Brussels.

References

Itziar Aldabe, Mikel Larrañaga, Montse Maritxalar, Ana Arruarte, and Jon A. Elorriaga. 2015. Domain module building from textbooks: Integrating automatic exercise generation. In Cristina Conati, Neil Heffernan, Antonija Mitrovic, and M. Felisa Verdejo, editors, *Artificial Intelligence in Education*. Springer International Publishing, Cham, pages 521–524.

Chris Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation* 47(1):97–122. https://doi.org/10.1007/s10579-012-9180-5.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. 2008. Phrase detectives: A web-based collaborative annotation game. In *Proceedings of the International Conference on Semantic Systems (I-Semantics' 08)*. pages 42–49.

Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. 2018. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)* 51(1):7.

Amarsanaa Ganbold, Altangerel Chagnaa, and Gábor Bella. 2018. Using crowd agreement for wordnet localization. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC-2018)*.

Bruno Guillaume, Karën Fort, and Nicolas Lefebvre. 2016. Crowdsourcing complex language resources: Playing to annotate dependency syntax. In *International Conference on Computational Linguistics (COLING)*.

Elizabeth Hadley, David Dickinson, Kathy Hirsh-Pasek, and Roberta Golinkoff. 2018. Building semantic networks: The impact of a vocabulary intervention on preschoolers' depth of word knowledge. *Reading Research Quarterly* 54:41–61. https://doi.org/10.1002/rrq.225.

Jeff Howe. 2006. Crowdsourcing: A Definition. https://www.wired.com/2006/06/crowds/.

Jan H. Hulstijn. 2013. *Incidental Learning in Second Language Acquisition*, Wiley-Blackwell, volume 5, pages 2632–2640). https://doi.org/10.1002/9781405198431.wbeal0530.

Anisia Katinskaia, Javad Nouri, and Roman Yangarber. 2018. Revita: a language-learning platform at the intersection of its and call. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC-2018)*.

Mathieu Lafourcade. 2007. Making people play for Lexical Acquisition with the JeuxDeMots prototype. In *SNLP'07: 7th International Symposium on Natural Language Processing*. Pattaya, Chonburi, Thailand, page 7. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00200883.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web* 6(2):167–195.

Verena Lyding, Lionel Nicolas, Branislav Bédi, and Karën Fort. 2018. Introducing the european network for combining language learning and crowdsourcing techniques (enetcollect). In Peppi Taalas, Juha Jalkanen, Linda Bradley, and Sylvie Thouësny, editors, *Future-proof CALL: language learning as exploration and encounters – short papers from EUROCALL 2018*, Research-publishing.net, pages 176–181.

Brian MacWhinney. 2017. A shared platform for studying second language acquisition. *Language Learning* 67(S1):254–275. https://doi.org/10.1111/lang.12220.

Chris Madge, Jon Chamberlain, Udo Kruschwitz, and Massimo Poesio. 2017. Experiment-driven development of a gwap for marking segments in text. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play*. ACM, pages 397–404.

Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. 2015. Yago3: A knowledge base from multilingual wikipedias. In *Proceedings of the 7th Conference on Innovative Data Systems Research (CIDR)*.

Detmar Meurers, Kordula De Kuthy, Florian Nuxoll, Björn Rudzewitz, and Ramon Ziai. 2019. Scaling up intervention studies to investigate real-life foreign language learning in school. *Annual Review of Applied Linguistics* 39.

I. S. P. Nation and Susan Hunston. 2013. *Learning Vocabulary in Another Language*. Cambridge Applied Linguistics. Cambridge University Press, 2 edition. https://doi.org/10.1017/CBO9781139858656.

Lionel Nicolas, Verena Lyding, Luisa Bentivogli, Federico Sangati, Johanna Monti, Irene Russo, Roberto Gretter, and Daniele Falavigna. 2018. Enetcollect in italy. In *Proceedings of the 5th Italian Conference on Computational Linguistics (CLiC-it 2018)*.

Christos Rodosthenous, Verena Lyding, Alexander König, Jolita Horbacauskiene, Anisia Katinskaia, Umair ul Hassan, Nicos Isaak, Federico Sangati, and Lionel Nicolas. 2019. Designing a prototype architecture for crowdsourcing language resources. In Thierry Declerck and John P. McCrae, editors, *Proceedings of the Poster Session of the 2nd Conference on Language, Data and Knowledge (LDK 2019)*. CEUR, pages 17–23.

Christos Rodosthenous and Loizos Michael. 2014. Gathering background knowledge for story understanding through crowdsourcing. In *Proceedings of the 5th Workshop on Computational Models of Narrative (CMN 2014)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Quebec, Canada, volume 41, pages 154–163. https://doi.org/10.4230/OASIcs.CMN.2014.154.

Christos Rodosthenous and Loizos Michael. 2016. A hybrid approach to commonsense knowledge acquisition. In *Proceedings of the 8th European Starting AI Researcher Symposium*. pages 111–122. https://doi.org/10.3233/978-1-61499-682-8-111.

Catherine Rosenbaum. 2001. A word map for middle school: A tool for effective vocabulary instruction. *Journal of Adolescent & Adult Literacy* 45(1):44–49.

Norbert Schmitt. 2013. *An Introduction to Applied Linguistics*. Hodder Arnold Publication. Taylor & Francis. https://books.google.com.cy/books?id=5gIvAgAAQBAJ.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceeding of the 31st AAAI Conference on Artificial Intelligence*.

Luis von Ahn. 2013. Duolingo: learn a language for free while helping to translate the web. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, pages 1–2.

Luis von Ahn and Laura Dabbish. 2008. Designing Games With a Purpose. *Communications of the ACM* 51(8):57. https://doi.org/10.1145/1378704.1378719.

Jointly Learning Author and Annotated Character N-gram Embeddings: A Case Study in Literary Text

Suraj Maharjan[⋆] **Deepthi Mave**[⋆] **Prasha Shrestha**[⋆] **Manuel Montes-y-Gómez**[†]
Fabio A. González[‡] **Thamar Solorio**[⋆]
[⋆]Department of Computer Science, University of Houston
[†]Instituto Nacional de Astrofisica Optica y Electronica, Puebla, Mexico
[‡]Systems and Computer Engineering Department, Universidad Nacional de Colombia
{smaharjan2, dmave, pshrestha3, tsolorio}@uh.edu
mmontesg@ccc.inoep.mx, fagonzalezo@unal.edu.co

Abstract

An author's way of presenting a story through his/her writing style has a great impact on whether the story will be liked by readers or not. In this paper, we learn representations for authors of literary texts together with representations for character n-grams annotated with their functional roles. We train a neural character n-gram based language model using an external corpus of literary texts and transfer learned representations for use in downstream tasks. We show that augmenting the knowledge from external works of authors produces results competitive with other style-based methods for book likability prediction, genre classification, and authorship attribution.

1 Introduction

Literary texts have been computationally modelled by extracting stylistic traits such as readability and writing density, flow of emotions, and even by cover images of books (Maharjan et al., 2018b,a, 2017; Ashok et al., 2013). However, modelling of authors through their work has not been explored until now. An author's style of presenting stories has a great influence on whether a book will be liked by readers or not.

We can find evidence of the effect that an author's style has on readers in book reviews and through readers' comments left on Goodreads[1] shown in Table 1. The readers talk about the impact of the author's writing style on their reading experience. In the first two examples, it left a positive impact on the readers, while in the last it had a negative impact. These examples provide further evidence for the need for modeling authors' writing style for the task of likability prediction of books.

This author's writing style is just perfect in every way, you will feel everything one should experience when you read a genre such as this.
The author's writing style is straightforward which made it easy to understand.
I think that the writing is very uneven. Overwhelmingly episodic, not terribly consistent, and largely as dimensionless as the characters.

Table 1: Readers comments showing the importance of authors' writing style

In this paper we propose a new approach to capture style in text by jointly learning author specific embeddings and character based n-gram embeddings. The idea of using author embeddings is motivated by reader comments as discussed above. The use of character n-gram embeddings comes from previous work on authorship attribution (AA) that has shown character n-grams to have strong prediction value for the task (Kešelj et al., 2003; Peng et al., 2003; Koppel et al., 2009; Stamatatos, 2009; Sapkota et al., 2015). Rather than using plain character-based n-grams, we first annotate them with their functional roles (*prefix*, *suffix*, and *whole-word*). This is necessary since, for example, *off* is semantically distinct from when it is used as a whole word and when it is used as a suffix (e.g. *trade-off*) or when it is used as a prefix (e.g. *offend*).

After obtaining representations for authors and annotated character n-grams using an external corpus of literary texts, we transfer them to tasks where author information is useful, namely book likability prediction and authorship attribution. Moreover, we provide quantitative and qualitative analyses of the author and annotated character n-gram embeddings.

[1] https://www.goodreads.com

Proceedings of Recent Advances in Natural Language Processing, pages 684–692,
Varna, Bulgaria, Sep 2–4 2019.

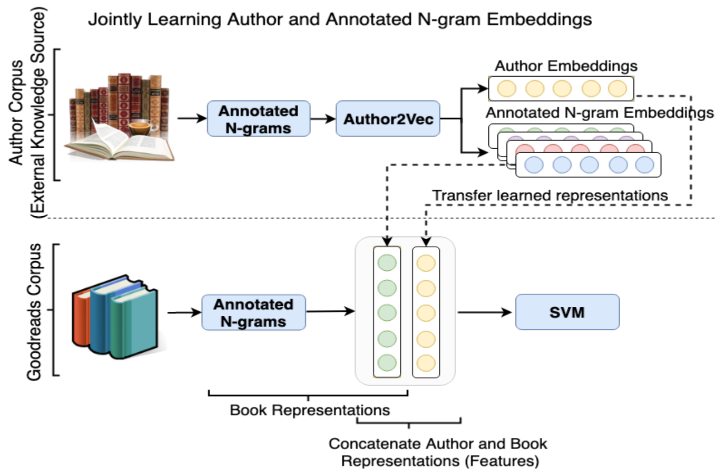

Jointly Learning Author and Annotated N-gram Embeddings
Author Corpus
(External Knowledge Source)
Annotated N-grams
Author2Vec
Author Embeddings
Annotated N-gram Embeddings
Transfer learned representations
Goodreads Corpus
Annotated N-grams
SVM
Book Representations
Concatenate Author and Book Representations (Features)

Figure 1: Learning and using author and annotated character n-gram embeddings

2 Methodology

Figure 1 shows the overall workflow of our proposed method. As shown in the figure, there are two phases. First, we jointly learn embeddings for authors and annotated character n-grams using an external corpus of books written by authors in the Goodreads corpus prepared by Maharjan et al. (2017). We refer to this corpus as Author Corpus. Next, we transfer this knowledge represented as the author and annotated character n-gram embeddings to build book representations. We describe these phases in detail below.

Phase I: Learning from an external corpus

We collected a new external corpus of books from Project Gutenberg to learn author and annotated character n-gram embeddings[2]. It consists of at most five books from each author in the Goodreads corpus (§3.1).

Annotated Character n-grams: We annotate character n-grams according to their position and function in a word as *prefix*, *suffix*, or *whole-word*. We follow the definitions by Sapkota et al. (2015) to decide which n-grams constitute each of these three types. *Prefix* and *suffix* are character n-grams that cover the first and last n-characters, respectively, of a word that is at least $n + 1$ characters. A *whole-word* n-gram is word that is exactly n characters long. This annotation helps to distinguish a single lexical entity as many different semantic entities. For instance, an n-gram like *the* could either be used as a *prefix* (***the**refore*), as a *suffix* (*wrea**the***), a standalone word (***the***), or occur within a word (*wi**the**r*). Note that although we do not explicitly annotate n-grams occurring mid-word, all of the remaining unannotated n-grams will fall under this category. These annotations will ensure that separate embeddings are learned according to the morphological and functional information carried by the n-grams.

Similar to Sapkota et al. (2015), we choose n as 3. While generating character 3-grams, we try various step sizes for sliding our window. With a step size of one, adjacent 3-grams will have two characters in common, one character in common with a step size of two and none with a step size of three. We name them *Overlap*, *Partial*, and *Non-Overlap*, respectively, based on the overlapping of characters in adjacent n-grams. We explore author and annotated character n-gram embeddings under these three settings.

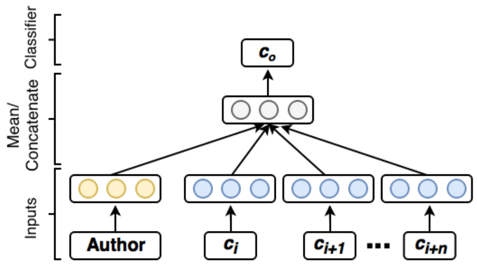


Figure 2: Author2Vec framework to learn author and character n-gram embeddings. $c_i, \ldots, c_{i+n}$ and $c_o = c_{i+n+1}$ are the input and output annotated character n-grams, respectively.

Author2Vec: Given a sequence of annotated character n-grams $c_i, \ldots, c_{i+n}$, and an author a, the objective of our Author2Vec model is to maximize

[2] The source code and data for this paper can be downloaded from https://github.com/sjmaharjan/author2vec.

the following conditional probability for the next-in-sequence output character n-gram c_o:

$$p(c_o|c_i, \ldots, c_{i+n}, a) = \frac{exp(y_{c_o})}{\sum_{c \in V} exp(y_c)} \quad (1)$$

$$y_c = Wh(c_i, \ldots, c_{i+n}; C, a; A) + b \quad (2)$$

where y_c is the unnormalized log probability for annotated character n-gram c in the vocabulary V, W and b are softmax parameters, and h is either the concatenation or mean function applied to character n-gram and author vectors from C and A, respectively. Similar to Le and Mikolov (2014), we call the concatenation method Distributed Memory Concatenation (DMC) and the mean method Distributed Memory Mean (DMM).

Phase II: Building book representations

We use the annotated character n-grams from Phase I to obtain the book's representations. We concatenate this with the book author's embedding and feed them as features to an SVM classifier. Similar to Maharjan et al. (2017), we consider the first 1k sentences from each book in the Goodreads corpus. We define the following three methods to obtain book representations:

Bag of annotated character n-grams (ACn): Similar to bag-of-word (BoW) approach, we generate the annotated character n-grams from books' content. We then represent each book by a sparse vector and weight each annotated character n-grams using their term frequency-inverse document frequency (TF-IDF) scores. The motivation behind using this representation is the success of stylistic analysis in the domain of books success prediction, author attribution, and author profiling.

Mean of Annotated character n-grams embeddings (Mean): Unlike the above method, here we use the annotated character n-gram embeddings to represent each book by a dense vector. We generate the annotated character n-grams from book content and look up their embeddings. A book is then represented by the mean of all annotated character n-gram embeddings generated from its content. Mathematically, the resulting book vector r is represented as $r = \frac{\sum_{i=1}^{N} emb(c_i)}{N}$, where N is the total number of annotated character n-grams for a book, and $emb(.)$ is the function that gets the embeddings for a given annotated character n-gram c.

Inverse Document Frequency (IDF) Weighted Average (Weighted): This method is similar to the above method of averaging the embeddings of annotated character n-grams, but weights each annotated n-grams embedding by their IDF scores before averaging. Mathematically, the resultant book vector r is represented as $r = \frac{\sum_{i=1}^{N} idf(c_i,B)*emb(c_i)}{N}$, where $idf(.)$ is the function that gets the IDF score for given annotated character n-gram c. The $idf(.)$ is learned from the training data and is defined as $idf(c, B) = \log \frac{|B|}{|\{d \in B : c \in d\}|}$, where B is the collection of books, and d is an instance of book.

3 Book Likability Prediction

Here we present results of using the author and character based n-gram embeddings for the task of predicting whether readers will like a book or not. We use likeability as a proxy to measure the success of a book. Narrowing down success as a measure of readers ratings is not ideal. But it gives us a practical starting approach to evaluate our models.

3.1 Dataset

We experiment with the publicly available book likability prediction dataset (Goodreads corpus) from Maharjan et al. (2017). They collected books from Project Gutenberg[3]. They then labeled the books into two categories: *Successful* and *Unsuccessful*, by using the average rating and the total number of reviews received by the books on Goodreads[4]. It consists of 1,003 books (654 *Successful* and 349 *Unsuccessful*) from 8 genres downloaded from Project Gutenberg: *Detective Mystery*, *Drama*, *Fiction*, *Historical Fiction*, *Love Stories*, *Poetry*, *Science Fiction*, and *Short Stories*.

3.2 Experimental Settings

We used the same stratified splits of 70:30 training to test as provided by Maharjan et al. (2017). We used the negative sampling (Mikolov et al., 2013) method to train 300-dimension embeddings for both authors and annotated character 3-grams. We filtered out 3-grams with frequency<2, set the window size to 5, configured the sample threshold to 1e-5 for randomly downsampling higher-frequency 3-grams, and trained for 100 epochs.

We predicted success separately (*Single task (ST)*) as well as simultaneously with genre (*Multi-task (MT)*). We used linear kernel SVM and tuned

[3] https://www.gutenberg.org/

[4] https://www.goodreads.com

Type	**Overlap**		**Partial**		**Non-Overlap**	
Features	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**
Character 3-grams (Maharjan et al., 2017)	66.9	70.0	-	-	-	-
All Typed n-grams (Maharjan et al., 2017)	66.3	69.1	-	-	-	-
Annotated char-3gram(AC3)	66.8	69.8	**71.1**	67.8	68.9	70.5
Mean (DMM)	60.5	67.6	63.7	66.6	65.6	**68.3**
Mean (DMC)	62.8	70.0	63.5	**70.1**	65.1	67.7
Weighted (DMM)	56.5	65.6	65.6	**69.9**	66.7	69.0
Weighted (DMC)	65.3	66.4	64.8	**67.2**	60.0	63.5
AC3 + Author (DMM)	62.8	68.5	67.5	67.5	**70.6**	68.5
AC3 + Author (DMC)	69.3	68.7	67.7	68.3	**71.3**	70.0
Mean + Author (DMM)	62.6	70.3	68.2	66.6	**71.9**	66.8
Mean + Author (DMC)	69.0	69.2	68.3	67.0	**71.5**	71.1
Weighted + Author (DMM)	62.3	70.0	66.9	66.6	70.6	**70.7**
Weighted + Author (DMC)	71.1	**73.8***	69.8	70.1	71.7	70.5

Table 2: Weighted F1-scores (%) using book representations with and without author embeddings under three settings (*Overlap*, *Partial*, and *Non-Overlap*). *statistically significant at $p < 0.02$ (McNemar significant test with and without author embeddings).

the *C* hyper-parameter through a grid search over (1e{-4,...,4}), using three-fold cross validation on the training split.

3.3 Results

Table 2 shows the results for our methods under the *Overlap*, *Partial*, and *Non-Overlap* settings. Our first set of experiments tests book representation methods (*AC3*, *Mean*, and *Weighted*) without author embeddings. The sparse feature representation method *AC3* (71.1%) performs better than embedding aggregation methods, *Mean* (69.9%) and *Weighted* (70.1%), as the mean operation likely removes important information. Here, the *Partial* setting yields the best results.

Our next set of experiments combine book representations with author embeddings and this improves the results in most cases. We obtain the overall highest F1-score of 73.8% with *Weighted* setting under concatenation (DMC) method. This result is statistically significant ($p < 0.02$) over the same setup but without authors' embeddings. This result is also better than the results from the state-of-the-art methods by Maharjan et al. (2017): *Character* 3*-grams* (70.0%) and *All typed* n*-grams* (69.1%). We also see that the DMC method yields consistently better results than the DMM method.

Author embeddings and correctness: We group authors by the genre of their books (in case of multiple genres, we pick the one with the most books). We then obtain representations for successful and unsuccessful authors in that genre by averaging the author vectors for these two classes. Our intuition was that if the distance between these representations is small, there will be fewer correct predictions for that genre. We obtained a large negative Pearson correlation coefficient of -0.753 ($p < 0.03$) between distances and the number of incorrect predictions, supporting our intuition.

Annotated vs Plain Character n-grams: To validate the importance of annotating n-grams, we trained embeddings using unannotated n-grams and performed likability prediction using the best setup from before. This produced an F1-score of 69.9% ($<$ 73.8%) illustrating the usefulness of considering the functional behavior of character n-grams. The performance further decreased to 63.4% with the removal of author embeddings.

Authors as binary vectors: To further confirm the advantage of learning author embeddings from external data, we replace author embeddings with one-hot vectors indicating the book's author. Using the best setup in Table 2, this produced a score of 70.3% ($<$ 73.8%), strengthening our intuition that author embeddings capture style related information, which is relevant for likability prediction.

Author Embeddings and Genre: We experimentally verify that author embeddings capture genre-specific information by using them to perform genre classification. We used the best model, Author (DMC), to automatically infer author vectors for all books in the dataset and fed them to an SVM classifier. We obtained F1-scores of 64.6%, 66.8%, and 64.0% with the *Overlap*, *Partial*, and *Non-Overlap* settings respectively. These scores outperform a random baseline of 15.2%, showing that author embeddings are also capturing more general style traces related to the genre.

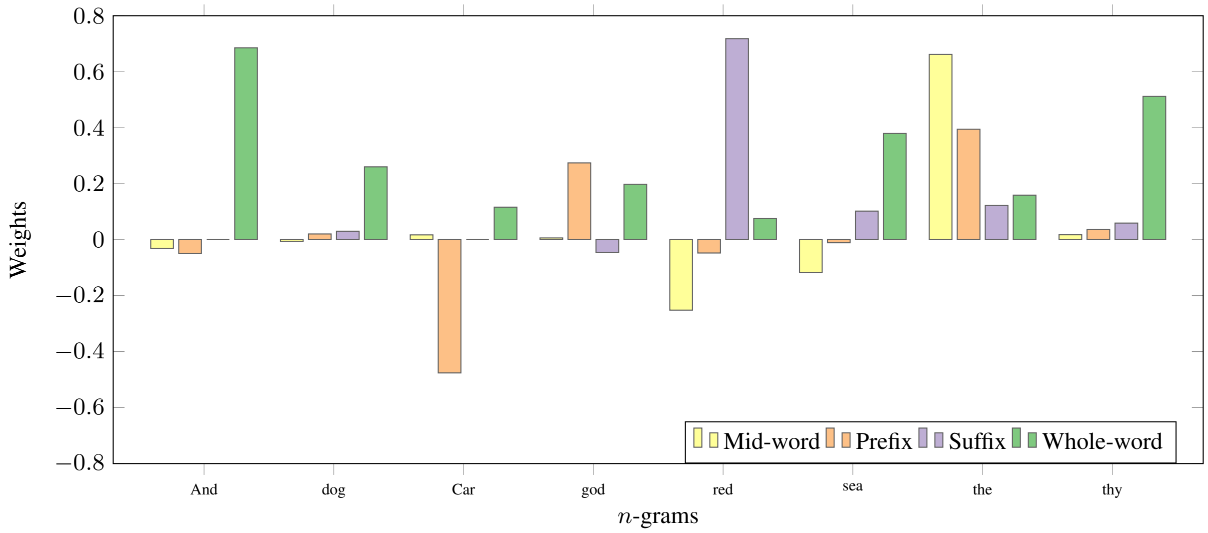


Figure 3: Feature importance assigned by SVM to different character n-grams for the likability prediction task.

4 Discriminative Annotated Character n-grams

Figure 3 shows some of the top positively and negatively weighted annotated n-grams by the classifier. We used the best performing *AC3* model from Table 2, *AC3* under *Partial* setting, to extract the weights for the annotated character n-grams. For each of the annotated n-grams, the figure also plots the weights for all four positional variants. The figure clearly shows that different forms of the same character n-gram have different contributions towards the likability prediction of books. This important piece of information would have been lost if we had treated these different forms of the n-grams as one. For instance, *sea* as a whole-word has a different meaning than when it is used as a prefix or a suffix. Accordingly, the classifier has also weighted them differently. The whole-word form of the n-gram *sea* is weighted higher than its other forms. This also holds for the case of *thy* and *dog*. During this analysis, we also found that quotation marks and male honorific titles were highly weighted by the classifier, similar to what Maharjan et al. (2017) found. This most likely points to the importance of dialogues and the preference of male characters in these books.

5 Analysis of Annotated Char n-grams

In Figure 4, we visualize the annotated character n-gram embeddings by projecting them using PCA. For some n-grams, the embeddings of different annotations are indeed distinct. For instance, the embeddings for *sub* (prefix, mid-word), *est* (whole-word, suffix), *ion* (suffix, whole-word), *mid* (prefix, suffix), and *the* (whole-word, suffix) lie far from each other. On the other hand, *ful* in suffix and mid-word form are close together, since the contexts for *ful* as a suffix (*beautiful*, *careful*) are similar to the contexts where it occurs mid-word (*beautifully*, *carefully*). In addition, we can also see a clear separation between *prefix* and *suffix* n-grams with the *mid-word* and *whole-word* n-grams occupying regions in between. The *suffix* and *prefix* n-grams mostly occupy the regions above and below the zero line respectively. This figure visually demonstrates that learning separate embeddings for n-grams with different functional roles is important to preserve their semantics.

Dataset	**RG (65)**	**WordSim (353)**	**RW (2034)**
Without Annotation	16.21	4.56	16.54
Annotated	30.75	12.27*	20.02**

Table 3: Results for word similarity task showing Spearman's rank correlation ($\rho \times 100$) with similarity scores assigned by human annotators $*p < 0.03$, $**p < 0.0001$

We also empirically show the advantage of these embeddings through the word similarity task using three standard datasets: *RG65* (Rubenstein and Goodenough, 1965), *WordSim353* (Agirre et al., 2009), and *RW* (Luong et al., 2013). Similar to FastText (Bojanowski et al., 2017), we represent a word as an average of the embeddings of its character n-grams. We create two representations

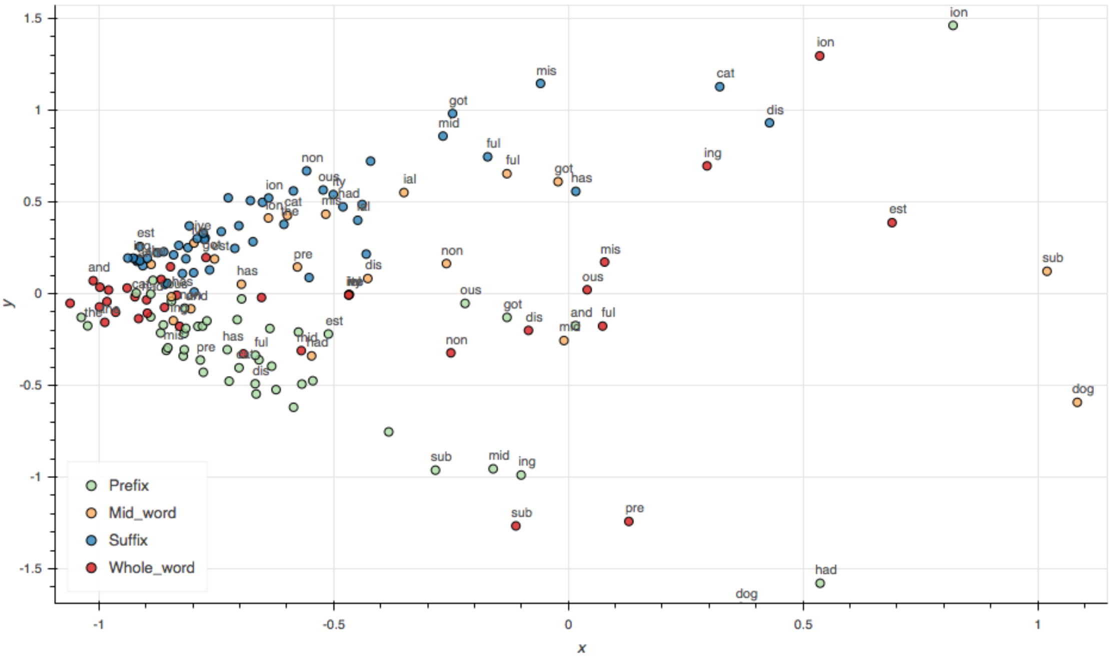


Figure 4: Projection of annotated char n-gram into 2D space using Principal Component Analysis (PCA)

for each word: one using plain character n-grams and another with annotated character n-grams. Table 3 shows the results for the word similarity task with these two different approaches. The Spearman's rank correlation coefficient between human annotations and word vectors composed of our annotated n-gram embeddings are higher than the same obtained from plain n-grams. The difference between the two proposed methods is statistically significant for WordSim353 ($p < 0.03$) and highly statistically significant for RW2034 ($p < 0.0001$). These results demonstrate that annotated character n-gram embeddings are also good at producing high-quality word representations and they might be capturing semantics at some level.

6 Authorship Attribution

Another task to test the effectiveness of our approach is authorship attribution (AA). To ensure that the model learns to discriminate authors and not the genre, we selected books from fiction genres for authors having at least five books that were not used to train our Author2Vec model. We have 12 such authors in our corpus.

We again used the first 1k sentences, used an SVM classifier in a stratified 5-fold cross-validation setup, and tuned the C hyper-parameter using grid search for each fold. Table 4 shows our results along with two baselines: word unigram and character 3-grams with tf-idf. Using only the book representations (Mean or Weighted), we obtained the highest mean accuracy of 86.67%. When we added in the inferred author embeddings (directly getting author embeddings would reveal the author, so we infer author embeddings similar to Le and Mikolov (2014) using only the book content without revealing the actual author of the book), the accuracy improved to 95% ($\sim$ 10% above *Char* 3-*gram*), showing that our approach not only works for likability prediction but also for AA.

7 Related Work

Sapkota et al. (2015) sub-grouped character n-grams according to grammatical classes, like affixes, lexical content, and stylistic classes, like beg-punct and mid-punct. With these sub-groups, they provided empirical evidence to support the importance of character n-grams features in the task of authorship attribution. Iacobacci et al. (2015) showed that learning separate word embeddings for polysemous words yields the state-of-the-art result in word similarity and relational similarity tasks. Separating the same tokens or n-grams helps to preserve their functional and morphological information which is important for all tasks. Learning embeddings for words (Mikolov et al., 2013), n-grams (Zhao et al., 2017), and documents (Le and Mikolov, 2014) and using them as input for various NLP tasks (Samih et al., 2016;

	Overlap (%)	**Partial (%)**	**Non-Overlap (%)**
Methods	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
Word Unigrams	83.33 ± 5.27	-	-
Char 3-grams	85.00 ± 6.24	-	-
AC3	81.67 ± 6.24	83.33 ± 9.13	83.33 ± 10.54
Mean	85.00 ± 6.24	86.67 ± 8.50	83.33 ± 9.13
Weighted	81.67 ± 12.25	80.00 ± 8.50	83.33 ± 10.54
Mean + Inferred Author	83.33 ± 11.79	**95.00** ± **4.08***	90.00 ± 6.24
Weighted + Inferred Author	83.33 ± 11.79	93.33 ± 6.24	90.00 ± 6.24

Table 4: Mean and standard deviation of accuracy for 5 fold cross validation AA experiments (*statistically significant at $p < 0.05$ from t-test with *Char* 3-*grams*)

Zhang et al., 2015; Kim, 2014) has shown improvement in performance. Again, Shrestha et al. (2017) showed that applying convolutional neural network (CNN) over character bigrams embeddings improves authorship attribution of short texts like tweets. They visually showed that character bigrams were capturing important stylistic markers to distinguish between bot-like authors and other normal authors. Song and Lee (2017) jointly learned the embedding for users (senders and receivers) and showed that these user embeddings capture the semantic relationship between users through an auto-foldering of emails task. Following these research findings, we also distinguish between the same character n-grams by annotating them with categories defined by Sapkota et al. (2015) and learn separate embeddings for each of them in addition to learning embeddings for authors.

Prior works in likability prediction of books have shown that style is an important aspect (Maharjan et al., 2017; van Cranenburgh and Bod, 2017; Ashok et al., 2013). They captured the style of successful and unsuccessful books using lexical, syntactic, readability, and writing density features, and deep learning methods with only first 1K sentences. Since style is evident even with first few fragments, they obtained competitive results using only first few fragments of books. Maharjan et al. (2017) even showed that around 200 sentences are enough to perform book success prediction with reasonable accuracy. Louis and Nenkova (2013) proposed features to capture different aspects of great writing (surprising, visual and emotional content) and used them in combination with genre-specific features to predict high quality writings in science articles. Iwana et al. (2016) extracted visual features from book covers for genre classification. Also, the potential of using a computer to plot the trajectory of emotion throughout the book and its correlation with success have been discussed (Vonnegut, 1981; Reagan et al., 2016). However, learning and using stylistically aware embeddings for authors in conjunction with other relevant stylistic features extracted from books for the problem has been overlooked. Our work fills this gap by adding author's general writing style learned using an external corpus of books.

8 Conclusions and Future Work

In this paper we explored a new dimension of modeling authors for literary texts by jointly learning annotated character n-gram embeddings and author embeddings using an external corpus. We showed that a book representation using our proposed embeddings significantly improves likability prediction results. Our approach was also able to obtain competitive accuracy for authorship attribution and genre classification, two tasks where style plays a prominent role. Moreover, we also demonstrated that annotated character n-gram embeddings yield higher quality word vectors. These results in likability prediction, authorship attribution, genre classification, and word similarity further demonstrate the usability of annotated character n-grams and author embeddings in varied tasks. In the future, we will extend our method to other domains where authors' information is important, such as author profiling.

Acknowledgments

This work was partially funded by National Science Foundation under award 1462141. We would like to thank the anonymous reviewers as well as Gayle McElvain and Brandon Burroughs for reviewing the paper and providing helpful comments and suggestions.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Boulder, Colorado, pages 19–27. http://www.aclweb.org/anthology/N/N09/N09-1003.

Vikas Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1753–1764. http://www.aclweb.org/anthology/D13-1181.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 95–105. http://www.aclweb.org/anthology/P15-1010.

Brian Kenji Iwana, Syed Tahseen Raza Rizvi, Sheraz Ahmed, Andreas Dengel, and Seiichi Uchida. 2016. Judging a book by its cover. *arXiv preprint arXiv:1610.09204* .

Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. 2003. N-gram-based author profiles for authorship attribution. In *In Proceedings of the Pacific Association for Computational Linguistics*. pages 255–264.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2009. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology* 60(1):9–26.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Bejing, China, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196. http://proceedings.mlr.press/v32/le14.html.

Annie Louis and Ani Nenkova. 2013. What makes writing great? first experiments on article quality prediction in the science journalism domain. *Transactions of the Association for Computational Linguistics* 1:341–352.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 104–113. http://www.aclweb.org/anthology/W13-3512.

Suraj Maharjan, John Arevalo, Manuel Montes, Fabio A. González, and Thamar Solorio. 2017. A multi-task approach to predict likability of books. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1217–1227. http://www.aclweb.org/anthology/E17-1114.

Suraj Maharjan, Sudipta Kar, Manuel Montes, Fabio A. González, and Thamar Solorio. 2018a. Letting emotions flow: Success prediction by modeling the flow of emotions in books. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 259–265. http://www.aclweb.org/anthology/N18-2042.

Suraj Maharjan, Manuel Montes, Fabio A. González, and Thamar Solorio. 2018b. A genre-aware attention model to improve the likability prediction of books. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 3381–3391. https://doi.org/10.18653/v1/D18-1375.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations (ICLR), Workshop* .

Fuchun Peng, Dale Schuurmans, Vlado Keselj, and Shaojun Wang. 2003. Language independent authorship attribution with character level n-grams. In *10th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 267–274.

Andrew J. Reagan, Lewis Mitchell, Dilan Kiley, Christopher M. Danforth, and Peter Sheridan Dodds.

2016. The emotional arcs of stories are dominated by six basic shapes. *CoRR* abs/1606.07772. http://arxiv.org/abs/1606.07772.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM* 8(10):627–633. https://doi.org/10.1145/365628.365657.

Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. 2016. Multilingual code-switching identification via lstm recurrent neural networks. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. Association for Computational Linguistics, Austin, Texas, pages 50–59. http://aclweb.org/anthology/W16-5806.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 93–102. http://www.aclweb.org/anthology/N15-1010.

Prasha Shrestha, Sebastian Sierra, Fabio A. González, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 669–674. http://www.aclweb.org/anthology/E17-2106.

Yan Song and Chia-Jung Lee. 2017. Learning user embeddings from emails. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 733–738. http://www.aclweb.org/anthology/E17-2116.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology* 60(3):538–556. https://doi.org/10.1002/asi.21001.

Andreas van Cranenburgh and Rens Bod. 2017. A data-oriented model of literary language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 1228–1238. http://www.aclweb.org/anthology/E17-1115.

Kurt Vonnegut. 1981. Palm sunday: An autobiographical collage.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*. pages 649–657.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, pages 244–253. https://www.aclweb.org/anthology/D17-1023.

Generating Challenge Datasets for Task-Oriented Conversational Agents through Self-Play

Sourabh Majumdar[1], **Serra Sinem Tekiroğlu**[2], **and Marco Guerini**[2]

[1]The State University of New York, NY 14260, Buffalo, USA
smajumda@buffalo.edu
[2]Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento, Italy
tekiroglu@fbk.eu, guerini@fbk.eu

Abstract

End-to-end neural approaches are becoming increasingly common in conversational scenarios due to their promising performances when provided with sufficient amount of data. In this paper, we present a novel methodology to address the interpretability of neural approaches in such scenarios by creating challenge datasets using dialogue self-play over multiple tasks/intents. Dialogue self-play allows generating large amount of synthetic data; by taking advantage of the complete control over the generation process, we show how neural approaches can be evaluated in terms of unseen dialogue patterns. We propose several out-of-pattern test cases each of which introduces a natural and unexpected user utterance phenomenon. As a proof of concept, we built a single and a multiple memory network, and show that these two architectures have diverse performances depending on the peculiar dialogue patterns.

1 Introduction

In recent years, there has been an increasing research on neural approaches for conversational systems. Such approaches include the realization of full end-to-end systems (Serban et al., 2016; Bordes et al., 2017), the capacity to incorporate or query structured knowledge sources into neural architectures (Eric and Manning, 2017), the use of zero-shot learning or of synthetic dialogues generation techniques to mitigate effort of domain portability (Zhao and Eskenazi, 2018; Guerini et al., 2018). Although state-of-the-art neural models achieve high performances in many domains, the sheer size of data they require represents a bottleneck, especially for under-resourced dialogue domains. In addition, it is hard to interpret their behaviour since neural models are intrinsically opaque. In this paper we propose a novel methodology to address the aforementioned problems by synthetically creating conversation datasets with peculiar characteristics. In particular, we focus on (i) scalable and portable approaches for generating dialogues from scratch involving complex and structured knowledge, and (ii) strategies for isolating and evaluating specific reasoning capabilities of neural architectures using synthetic *challenge sets*. To this end, we utilize dialogue self-play strategies (Shah et al., 2018) simulating task oriented dialogues between a conversational agent and a user. Dialogue simulation grants complete control over the generated data allowing building special test cases, each of which introduces a natural and unexpected user utterance phenomenon, that has never been seen at training time. It should be noted that the use of natural data would require *manual* annotation/selection of examples for each phenomenon of interest, making this goal intractable.

Another problem with current datasets is the lack of exhaustiveness: they usually focus on one specific domain intent. There are few datasets covering multiple intents in the same scenario, but usually these intents are very different from one another – e.g. weather forecast and play music in a car scenario (Eric and Manning, 2017). For this reason, we choose a compelling low resource dialogue domain, i.e. Banking, that allows us to bring together multiple tasks (e.g. transfer money, check balance, block card).

2 Related Work

Our focus is on synthetic data generation techniques for low-resource domains and for inves-

Proceedings of Recent Advances in Natural Language Processing, pages 693–702,
Varna, Bulgaria, Sep 2–4 2019.

tigating the learning capabilities of neural conversational agents. For this reason, we will discuss some artificial conversational datasets, testing strategies for dialogue models, and the main dialogue systems approaches.

Artificial Datasets. Many conversational datasets rely on crowd sourcing (Jurčíček et al., 2011; Kelley, 1984), cooperating corporations (Raux et al., 2005), already available records (Lowe et al., 2015; Danescu-Niculescu-Mizil and Lee, 2011), or participation with the real world (Yu et al., 2015; Zue et al., 1994). Moreover, available task oriented dialogue datasets are mainly focused on very specific domains, such as restaurant reservation (Jurčíček et al., 2011; Henderson et al., 2014), flight booking (Zue et al., 1994), bus information systems (Raux et al., 2005) and giving directions to rooms in a hotel (Yu et al., 2015). Since the collection of whole dialogues is usually very expensive and time consuming, there have been efforts in building methodologies that allow fast and cheap data collection (Shah et al., 2018; Kelley, 1984). Artificial data generation is an effective methodology for obtaining well defined training datasets. Bordes et al. (2017) uses a simulator for this purpose. Other approaches artificially outline the conversational flow of their examples and then use crowdsourcing to convert dialogue turns into natural language (Shah et al., 2018).

Testing Neural Dialog Models. With the rise of Neural NLP, interpretability has become a major issue. Belinkov and Glass (To appear) survey various methods addressing interpretability. In particular, one line of research deals with challenge sets (Lehmann et al., 1996). While the majority of NLP datasets reflects a natural distribution of language phenomena, challenge sets are meant to restrict their focus on a specific phenomenon (quantifiers, plurals, anaphora, etc.) at a time (Cooper et al., 1996). Analyzing the ability to deal with a specific phenomenon allows evaluating the systems in a more principled way. Challenge datasets have been applied to diverse tasks, such as Natural Language Inference (Wang et al., 2018) and Machine Translation (King and Falkedal, 1990). These datasets offer insight if a model is capable of handling a specific line of reasoning from training data to specific structures at test time.

Methods for conversational scenarios. Many methods have been proposed to deal with conversational scenarios. *Rule Based Systems* are the simplest to implement for the task oriented setting when the flow of the dialogue is already known. They tend to be highly brittle to patterns not seen during their construction or to porting to new domains (Marietto et al., 2013). *Information Retrieval Methods* usually imply the two main approaches, namely TF-IDF and Nearest Neighbor models (Isbell et al., 2000; Jafarpour et al., 2010; Ritter et al., 2011; Sordoni et al., 2015). More recently, *Neural Network Models* have been heavily employed for conversational agents. Sequence-to-sequence models (Vinyals and Le, 2015) perform well for short conversations but fail in longer ones (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Hierarchical Encoder Decoder Models (Serban et al., 2016) are an extension of the sequence-to-sequence models. They handle the context in a separate RNN and use this context for response generation. Finally, Latent Variable Hierarchical Encoder Decoder models (Serban et al., 2017) represent a further improvement of the Hierarchical Encoder Decoder Model. Other recent approaches have focused on Memory Networks, that use an external memory component build into the system to store long term contexts (Weston et al., 2014). In particular, end-to-end Memory Networks (Sukhbaatar et al., 2015), an extension where every component is trained in an end-to-end fashion, showed promising results in task oriented dialogues (Bordes et al., 2017).

3 Data Generation through Self-Play

Unlike the natural challenge sets discussed in Section 2, our focus is on testing structured reasoning in conversational context, by using synthetic dialogue generation to grant that phenomena under investigation are present only at test time and not at training time. To our knowledge, this is the first attempt to build challenge datasets in an artificial way and to use them for the dialogue domain. Natural data would not be suitable for our purpose since the challenging test phenomena can also be found in the training set. In particular, our dataset is constructed using a dialogue self-play approach (Shah et al., 2018). This approach suggests that we can always simulate a conversation if we treat it like a game. During the simulations, we instantiate two bots; the system and the user. For each conversation, we pick up a user profile, then user and system bots carry on the conversation through pseudo-language ***actions*** regarding

Logical Form	Example Annotation
BOT: How can I help you today ?	*BOT: How can I help you today ?*
inform_intent = transfer	I need to send some money.
inform_intent = transfer {amount}	I want to transfer {amount}
inform_intent = transfer {partner}	Can I send money to {partner}?
inform_intent = transfer {partner} {amount}	I would like to send {amount} to {partner}.
BOT: Who is the recipient?	*BOT: Who is the recipient?*
inform {partner}	It is {partner}.
BOT: What is the transfer amount?	*BOT: What is the transfer amount?*
inform {amount}	Roughly {amount}.

Table 1: Automatically generated logical forms provided to annotators and annotated template samples for a Money Transfer intent with 2 slots. Bot request is provided to annotators to give better context.

the user profile.

Therefore, every conversation is represented as an exchange of actions between the two agents. Each action contains the information on who performed it and what values the agent provided. For each dialog for a chosen intent, the respective system bot asks the relevant slots and the user bot provides the appropriate slot values. The system bot then checks the values provided and issues the next requests accordingly. Both API calls and slot requests are performed through actions. To convert these actions into actual dialogues, we conducted an annotation task with 5 annotators. First, we converted each possible action for all the intents into logical forms (uninstantiated pseudo-code) and asked the annotators to provide natural language templates for each logical form without changing the entity placeholders. We then used the language templates to create natural language utterances by replacing the placeholders with the actual values. In Table 1, we give a few examples of the logical representations provided to annotators and the template they wrote. The conversion to the final plain text has been done by filling annotated templates with a small KB (the user profile). This approach is different from the one proposed by Shah et al. (2018) that uses instantiated pseudo-code since the beginning: it requires much more data annotation and makes it more difficult to detach surface realization from dialogue structure for building challenge sets.

The advantages of our synthetic data generation is twofold. First, it helps us to achieve a better coverage since reducing dialogues to an exchange of actions allows having a formal and simple visualization of all the generated dialogue flows - this is not the case with WoZ data collections (Kelley, 1984). Second, by instantiating each dialogue with different natural language templates, we can create dialogues that have the same structure but different wording.

3.1 The System and User Bot Design

We designed the user and the system bots using a finite state machine approach (Hopcroft et al., 2001), assuming that each dialogue is a flow between states and each utterance is a move from one state to the next. For our experiments, each of these states handles one particular slot. Whenever a user bot provides some information through its action, the system bot changes the state accordingly and performs its own action in response. The user bot then deals with the system action, performs its own and the cycle continues. The dialogue concludes when the system bot reaches an end state and issues an end call action.

3.2 Banking Domain Intents

The user intents/tasks that the dialogues are built upon are in the Banking Domain. We selected several intents to create a dataset that contains: i) conversations that are varied and diverse, and ii) conversations that use similar sentences and slots but to complete different tasks.

Each dialogue is initiated by the system bot asking the first question. To add variability we randomized the number of slots provided by the user in its response, similar to verbosity user trait in (Shah et al., 2018). After these initial turns the system bot ask the respective missing slots. Apart from asking the missing slot, the system bot also validates the value of the slots by calling respective APIs, since by design the User Bot may provide incorrect values and the System Bot is designed to handle all these possible cases. In Table 2, we give an example dialogue to demonstrate how API intensive these dialogues might become. Note that, for each interaction we have a specific user profile, i.e. the possible entities for slots are predetermined. For example, "partner list" for money transfer is fixed, and a user cannot provide a random name. Additionally, dialogues are formed with respect to policy restrictions such that i) the slots for each intent are asked in a strict order, ii) each slot value should be validated through an api call, iii) there are maximum 3 consecutive invalid user utterances after which the dialogue is terminated, iv) the final API call for intent execution randomly returns a success or failure. The valid/invalid user utterances are chosen randomly in each turn. We now list the domain intents and their descriptions as follows:

Making a Transfer. In this intent, USER wants to transfer some money. The system is tasked to fill three slots i.e *user account*, *partner name* and *amount*.

Canceling a Transfer. This intent handles cases where USER wants to cancel a transfer made earlier. The first required slot is the *transaction id* and USER may or may not recall it. If the transaction id cannot be filled, SYSTEM asks for *partner name* and checks for previous transactions.

Blocking a Card. The main slot required for this intent is the *card id*. Similar to the canceling a transfer intent, there are two possible paths for the conversation flow. If USER cannot recall the *card id*, SYSTEM asks another set of slots, which are the *user account* and the *card type*.

Transaction History Query. This intent is for finding a transaction in USER's transaction history by searching the *note* field. Notes are specific texts giving information about the transaction reason, such as "Salary" or "Electricity bill".

Account Balance Query. To search for the account balance, USER is asked to provide the *user account* name.

Account Limit Query. USER searches for the maximum transferable amount from his/her account. This intent, together with account balance, serves to evaluate the disambiguation performance of the conversational models when facing semantically similar intents - in fact their dialogue flow and required slots are identical.

S : How can i help you today?

U : I am really concerned that i lost the bank card, is its deactivation possible?

S : Please enter the card id.

U : I am not sure about the card id

S : Which account does the card belong to?

U : It is associated with my checking account.

A: CALL:CHECK_ACCOUNT, USER_ACCOUNT:CHECKING

A: RESPONSE:ACCOUNT_CHECK, RESULT:FAILED, MESSAGE:'list of available accounts joint , savings'

S : I couldn't find the account you are looking for. Your available accounts are joint , savings. Would you like to change your answer?

U : Yes

S : Which account does the card belong to?

U : It must be savings

A : CALL:CHECK_ACCOUNT, USER_ACCOUNT:SAVING

A : RESPONSE:ACCOUNT_CHECK, RESULT:SUCCESS

S : You have the following cards that belong to your account: prepaid card, credit card. which card would you like to block?

U : Can you block the prepaid card?

A : CALL:CHECK_CARD_NAME, CARD_NAME:PREPAID

A : RESPONSE:CARD_NAME_CHECK, RESULT:SUCCESS

S : Do yo confirm blocking your card?

U : yes

A : CALL:BLOCK_CARD, CARD_ID:SAVINGS-PREPAID

A : RESPONSE:BLOCK_CARD, RESULT:SUCCESS

S : Your card has been successfully blocked.

Table 2: A complete Block card example. U corresponds to User, S to System, and A to API calls.

3.3 Challenge Test Set

We generate a wide variety of test cases using the same dialogue self-play methodology employed for the training data, by modifying the structure of the original dialogues in a principled manner. Each of these test cases represents a particular conversational situation that has never been seen during training but that might happen in real interactions. While we cannot precisely quantify the exhaustiveness of the considered cases in real life scenarios due to the domain restrictions, the synthetic dialogue generation allows us to easily integrate new challenge cases when necessary. Below, we discuss in detail each test case and what function it serves to test. We also present a more conventional test set, i.e. out of templates set, to disclose how much we elevate the difficulty of the task by the challenge test cases. In our experiments, we will disregard Out Of Vocabulary (OOV) cases, i.e. cases where entities of interest at test time are not seen at training time, e.g. the partner name "Michael", (Bordes et al., 2017).

Out of Templates (OOT) Out of templates test case is constructed by sampling annotation templates into training, development and test sets. By doing so, we can test whether the agent can handle user utterances in an unseen linguistic form.

Out of Pattern (OOP) Out of pattern test cases challenge the agents through the conversations with a structure (dialogue flow) that has not been seen during training. We have constructed five out of pattern cases for the challenge test set. Together with each OOP description, we provide an example, where the arrow indicates the answer of the system bot to be predicted, and the part in italic highlights the differences between training and test sets for readability purposes.

1. Turn Compression The training dialogue structure contains a confirmation step after each attempt to fill the current slot with an invalid value. Following a confirming answer from USER for changing the slot value, SYSTEM repeats the slot filling question to USER. In the turn compression challenge test case, we concatenate the confirmation and the new slot answers as the user utterance for the change confirmation question of SYSTEM. The correct system utterance is the validation API call of the new value instead of the slot filling question again. Table 3 shows an example where SYSTEM asks if USER wants to change the *partner name*, and during testing the user gives a confirming answer together with the correct *partner name*.

Sys : Partner name is incorrect, would you like to change it?

User : *Yes*

→**Sys** : *What is the name of the recipient?*

Sys : Partner name is incorrect, would you like to change it?

User : *Yes, change it to Megan*

→**Sys** : *API PARTNER CHECK **Megan***

Table 3: Train/test excerpt of Turn Compression

2. New API The new API challenge is designed to evaluate the performance of an agent in terms of its ability of issuing the required API calls with appropriate slot values. To this end, we create intent utterances in training data either without any extra slot value or odd number of slot values while in the new API challenge set, intent utterances contain only even number of slot values. See Table 4 for an example.

Sys : How can i help you today ?

User : I want to see if my salary was credited

→ **Sys** : API CHECK SALARY

Sys : How can i help you today?

User : I want to see if my salary was credited from Facebook

→ **Sys** : API CHECK SALARY *FACEBOOK*.

Table 4: Train and test excerpt of New API

3. Reordering The training dialogues have a particular slot filling order for each intent. In the reordering challenge set, on the other hand, the order of slots in dialogues is shuffled. The purpose of reordering is to evaluate whether an agent is able to generate the system utterance to fill the correct missing slot after all other slots for the current intent have been filled. In the example given in Table 5, we show the correct system utterance of a new API challenge test case for the missing slot: *user account*.

User : I'd like to transfer some money.

Sys : From which account?

User : from Savings please.

Sys : Who is the recipient ?

User : It is Michael

→ **Sys** : *What is the amount?*

User : I'd like to transfer some money.

Sys : *What is the amount* ?

User : 100 euros.

Sys : Who is the recipient ?

User : It is Michael

→ **Sys** : *From which account?*

Table 5: Train and test excerpt of Slots Reordering

4. Another Slot In a natural conversation, USER can provide an utterance that is irrelevant to the current turn while including a slot value relevant to another one. Example dialogue excerpts for Making a Transfer intent with and without such *another slot* test case are given in Table 6.

5. Audit More In the training dialogues, after SYSTEM requests a new slot value for the value

Sys : What is the partner name ? **User** : *It is Michael.* →**Sys** : *API PARTNER CHECK* ***Michael***
Sys : What is the partner name ? **User** : *Make the amount to be 10k euros .* →**Sys** : *API AMOUNT CHECK* ***10k euros***

Table 6: Train/test excerpt for Another Slot case

changing turn, USER is expected to provide an appropriate value for the slot. In the audit more challenge set, USER provides other slot values along with the requested slot. We test if the system can recognize the changed slot value and issue the appropriate API calls for the slots given by USER. In the example test dialogue given in Table 7, USER changes the amount along with the partner name slot. The correct response of SYSTEM would be the API calls for the given partner name and the amount values.

Sys : What is the partner name ? **User** : *Change it to Michael.* →**Sys** : *API PARTNER CHECK* ***Michael***
Sys : What is the partner name ? **User** : *Change it to Michael and make the amount to be 5k euros.* →**Sys** : *API PARTNER CHECK* ***Michael,*** *AMOUNT CHECK* ***10k euros***

Table 7: Train and test excerpt of Audit more.

We should note that each OOP test can contain only one type of OOP, i.e only turn compression or only new API etc. However, in one dialog there may be more than one instance of the same OOP.

4 Neural Models

For our experiments, we focused on end-to-end Memory Networks (Sukhbaatar et al., 2015), which have been employed in various NLP tasks including non-goal-oriented (Dodge et al., 2016) and goal-oriented (Bordes et al., 2017) dialogue systems. End-to-end memory networks exploit a memory component to store the dialog history, multiple memory look-ups (hops) as an attention mechanism, and short-term context to predict the next response. They yield promising performances in dialogue tasks outperforming some other end-to-end RNN based architectures. In addition, end-to-end memory networks have been shown to be able to perform non-trivial operations such as issuing API calls to KBs (Bordes et al., 2017), which are a key element to our scenario. We implemented 2 variations of memory networks in order to test the feasibility of challenge set by analyzing the performance differences of the networks.

Single End-to-End Memory Network (SMN) replicates the end-to-end memory network presented by Bordes et al. (2017) trained on all dialogues from all intents simultaneously.

Multiple End-to-End Memory Network (MMN) is an ensemble of 6 different Memory Networks, each trained on a single intent and a 7^{th} Memory Network that has the task of selecting the appropriate one for the given dialogue. The training data for the 7^{th} memory network is produced by appending a call-memory-network action after the user utterance that informs the intent in a dialog.

We implemented MMN and tested against SMN to investigate if sharing information from different intents plays a positive role or creates interference, especially for intents that contain the same slot types and have a high semantic similarity.

5 Experiments

We used a learning rate of 0.001, a batch size of 32, and maximum 20 epochs during the training of both networks. We set the embedding size to be 128 as it has been shown to perform good enough for most NLP problems (Bai et al., 2009). For memory networks we empirically used a memory size of 40 and 3 hops.

We have generated 200 dialogues per intent and per test case through the methodology explained in Section 3. We first sampled $1/3$ of the templates to linguistically realize the logical form of training dialogues and $1/3$ for development. For the in-template test cases we then randomly sampled turns from training and development to create test dialogues. Instead, for the OOT test configurations we used the remaining $1/3$ of the templates to generate new linguistically unseen test cases. Similar to Bordes et al. (2017), the networks are evaluated in a ranking, not a generation, task: we test whether for each dialogue turn, the MNs can identify the next (correct) system utterance or API call from a list of candidates. Regarding the evaluation metric, we have used Per-Response Accuracy, which is the number of correct system responses predicted out of all dialogue turns in the whole test set. Each response under test is compared to the predicted response by the model when given a dialogue memory, i.e. context, and a new user utterance.

Considering the banking domain and task scenarios, for which it is almost impossible to record and collect real human interactions, we cannot perform an experiment on real life conversations. Although using WoZ technique and challenge pattern annotation could be applicable, it could not be guaranteed that we can collect reasonable amount of data for the same challenge pattern and we would be obliged to know every pattern in advance. Therefore, we also employ dialogue synthesis for test/challenge set dialogues, which allows for a continuous challenge design by modifying the template combinations and dialogue patterns.

5.1 Test Cases and Results

We compare the memory networks against the per-response accuracy for In Template (IT henceforth) and Out Of Template setting. In Table 8, we show the test results of SMN and MMN models for IT and OOT configurations (including non-OOP and OOP test settings).

	IT		OOT	
Test Case	*SMN*	*MMN*	*SMN*	*MMN*
Non OOP	88.62	**90.17**	87.39	**88.27**
Turn Comp.	27.80	**55.00**	27.90	**54.70**
New API	7.42	**7.83**	**8.17**	6.67
Reordering	**54.50**	45.50	**54.00**	41.50
Another Slot	**38.00**	25.00	**41.50**	27.50
Audit More	15.50	**34.00**	16.00	**35.00**
OOP Avg.	28.64	**33.47**	29.51	**33.07**

Table 8: In template non-challenge/OOP test results, OOT non-challenge/OOP test results in terms of per-response accuracy.

OOP impact. As expected, OOP cases represent a compelling challenge for our MNs, see Table 8. When we compare the results of the non-OOP and the OOP cases, we observe drastic performance differences, both in IT (88.62 vs. 28.64, 90.17 vs 33.47) and OOT (87.39 vs. 29.51, 88.27 vs. 33.07) settings. Still, in some settings both MNs are able to show reasonable performances and different behaviors on different challenge sets (reordering and another slot for SMN, turn compression and audit more for MMN).

Single vs Multiple Memory Network. Concerning the IT cases, MMN slightly outperforms SMN in the non-challenge test setting. In addition, it shows a substantial accuracy increase in turn compression and audit more OOP cases. On the other hand, SMN surpasses MMN in reordering and another slot OOP challenges. A similar outlook of performances is observed for the OOT non-challenge and OOP cases aside from new api challenge, which turns out to be the most difficult OOP challenge and will be discussed later.

We observe that on average MMN outperforms SMN both in IT and OOT cases. One possible explanation is based on how the memory network finds the correct response. The Single Memory Network does so by incorporating all the intents in its selection of the responses. Therefore, it searches for more general responses while the Multiple Memory Network assigns the same task to a specialized Memory network, which is trained on that very specific intent. The specialized memory network is better at finding the correct response since it is trained only on one particular intent data and its search space is implicitly reduced.

As a particular OOP performance comparison, we noticed that SMN is better at selecting the right response during the reordering challenge, which evaluates the ability of the model in learning the necessary slots to accomplish an intent.

In template vs Out of Template. We found out that there is not a major difference between IT and OOT test case performances (slightly better for SMN and slightly worse for MMN). One possible explanation is that the tests have not been conducted in an OOV setting. Therefore, the SMN and MMN may not learn the linguistic patterns to find entities (templates) but they directly learn to recognize the entities and to predict the API calls accordingly.

Multiple Out of pattern. As a final experiment, we wanted to inspect the effect of having the challenge phenomenon of interest appearing more than once in a dialogue, such as the example proposed in Table 10. For this last case we could use only turn compression and audit more, that have a sufficient number of slots to replicate the phenomenon of interest over different slots in the same dialogue. What we observe is that indeed it is difficult

	Per-Response Accuracy	
Test Case	SMN	MMN
Turn Compression	29.61	55.16
Turn Compr. OOT	29.07	55.13
Audit More	10.50	15.02
Audit More OOT	10.90	15.43

Table 9: Multiple Out of Pattern per Dialog

Sys : The partner name you provided is incorrect, would you like to change ?
User : *Yes, change it to Michael (first Occurrence of Turn Compression)*
Sys : Okay, your amount also exceeds the limit, would you like to change ?
***User** : Yes, It is 100 euros (The second occurrence of turn compression)*

Table 10: Example of Multiple OOP dialogue

for both the SMN and the MMN to handle the case of more than one OOP.

It can be seen that there is a drop in the performances of both MNs in Table 9 as compared to previous challenge tests, only for audit more. This drop can be attributed to the differences of the contexts of the conversations that are present in the memory while selecting the response. Since the context is the previous conversation until the chosen turn, for the first case, i.e. the first example in 10, the context is a usual sub-dialogue pattern that is seen during the training. So the responsibility of the agent is to understand the new unseen compressed user utterance and choose the correct response. However, when we test the second turn compression in the same dialogue, i.e. the second turn in 10 where USER changes the amount, the responsibility of the agent is compounded by the fact that in addition to understanding the compressed turn, it has to reason about an unseen context pattern. In other words, the context also contains a form of turn that the agent has never seen during training, which is the first turn compression of the second example in Table 10.

The Easiest and the Hardest Challenge Cases Finally, to investigate the difficulty of challenges that we have introduced with each OOP case, we should focus our attention to the easiest and the hardest cases. We observe that out of all the OOP test cases (both in-template and OOT settings) both memory networks performed quite poorly on handling new APIs. The results suggest that it is harder for the memory networks to interpret a new combination of slots and issue the related API calls. This could be partially explained by the position of the new API cases in the dialogue. By design, new API cases happen at the beginning of the conversation (i.e. giving an intent together with unexpected slots, see example in Table 4). Therefore, the system has no context (no interaction memory) to reason on while selecting the response. On the contrary, for the easier turn compression case, the memory network is already expecting a possible change in the slot value (e.g. "*do you want to change the amount?*") in the following turns, regardless of receiving it in the respective turn or in the next few. In fact, the network is already 'primed' on selecting an amount related API call. Consequently, the memory networks have a better performance on turn compression rather than new API challenge.

6 Conclusions and Future Work

In this paper, we explored some challenges connected to dataset creation for conversational agents and interpretability of neural models. In particular, we propose a methodology to create rich datasets for training end-to-end conversational agents and challenge them on unseen patterns at test time. We then experimented with Memory Networks and investigated their performance on the custom test cases. The apparently low accuracy levels on unseen challenge cases suggest that the synthetic data and challenge generation for low resource dialogue domains can act as a reasonable approximate to real life challenges in such domains. In other words, the more a dialogue model is able to handle these diverse challenges, the more it will be able to handle the unstructured or less structured dialogues in real human-machine interaction. As a future work we would like to test further neural models and create additional OOP challenge sets, even combining additional configurations.

References

Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. 2009. Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, pages 187–196.

Yonatan Belinkov and James Glass. To appear. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics (TACL)* .

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=S1Bb3D5gg.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. Association for Computational Linguistics, pages 76–87.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. http://arxiv.org/abs/1511.06931.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414* .

Marco Guerini, Simone Magnolini, Vevake Balaraman, and Bernardo Magnini. 2018. Toward zero-shot entity recognition in task-oriented conversational agents. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. pages 317–326.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. pages 263–272.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News* 32(1):60–65.

Charles Lee Isbell, Michael Kearns, Dave Kormann, Satinder Singh, and Peter Stone. 2000. Cobot in lambdamoo: A social statistics agent. In *AAAI/IAAI*. pages 36–41.

Sina Jafarpour, Christopher JC Burges, and Alan Ritter. 2010. Filter, rank, and transfer the knowledge: Learning to chat. *Advances in Ranking* 10:2329–9290.

Filip Jurčíček, Simon Keizer, Milica Gašić, Francois Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2011. Real user evaluation of spoken dialogue systems using amazon mechanical turk. In *Twelfth Annual Conference of the International Speech Communication Association*.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)* 2(1):26–41.

Margaret King and Kirsten Falkedal. 1990. Using test suites in evaluation of machine translation systems. In *COLNG 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*. volume 2.

Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, et al. 1996. Tsnlp: Test suites for natural language processing. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 711–716.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909* .

Maria das Graças Bruno Marietto, Rafael Varago de Aguiar, Gislene de Oliveira Barbosa, Wagner Tanaka Botelho, Edson Pimentel, Robson dos Santos França, and Vera Lúcia da Silva. 2013. Artificial intelligence markup language: A brief tutorial. *arXiv preprint arXiv:1307.3091* .

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 583–593.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*. volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*. pages 3295–3301.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*. pages 2440–2448.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916.

Zhou Yu, Dan Bohus, and Eric Horvitz. 2015. Incremental coordination: Attention-centric speech production in a physically situated conversational agent. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. pages 402–406.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. *arXiv preprint arXiv:1805.04803* .

Victor Zue, Stephanie Seneff, Joseph Polifroni, Michael Phillips, Christine Pao, David Goodine, David Goddeau, and James Glass. 1994. Pegasus: A spoken dialogue interface for on-line air travel planning. *Speech Communication* 15(3-4):331–340.

Sentiment Polarity Detection in Azerbaijani Social News Articles

Sevda Mammadli[1], Shamsaddin Huseynov[1], Huseyn Alkaramov[1], Ulviyya Jafarli[1], Umid Suleymanov[2], Samir Rustamov[1,3]

[1] School of Information Technologies and Engineering, ADA University, Baku, Azerbaijan

[2] E-GOV Development Center, Baku, Azerbaijan, [3] Institute of Control Systems, Azerbaijan

{smammadli2019, shuseynov2019, halkaromov2019, ujafarli2019}@ada.edu.az

umidsuleymanov96@gmail.com

srustamov@ada.edu.az

Abstract

Text classification field of natural language processing has been experiencing remarkable growth in recent years. Especially, sentiment analysis has received a considerable attention from both industry and research community. However, only a few research examples exist for Azerbaijani language. The main objective of this research is to apply various machine learning algorithms for determining the sentiment of news articles in Azerbaijani language. Approximately, 30.000 social news articles have been collected from online news sites and labeled manually as negative or positive according to their sentiment categories. Initially, text preprocessing was implemented to data in order to eliminate the noise. Secondly, to convert text to a more machine-readable form, BOW (bag of words) model has been applied. More specifically, two methodologies of BOW model, which are tf-idf and frequency-based model have been used as vectorization methods. Additionally, SVM, Random Forest, and Naive Bayes algorithms have been applied as the classification algorithms, and their combinations with two vectorization approaches have been tested and analyzed. Experimental results indicate that SVM outperforms other classification algorithms.

1 Introduction

Development of technology generates a vast amount of data flow through the internet and encourages the creation of sophisticated methodologies to store and analyze it. Analyzing such huge volumes of data manually could lead to the waste of time and investment. Therefore, currently more automated and efficient ways are implemented to solve the problem.

With the growth of the produced data, a branch of Artificial Intelligence - Natural Language Processing (NLP) had begun to evolve. Text classification is a fundamental part of NLP and has been applied in many areas. The aim of text classification is to group data into predefined categories based on the labeled data using various machine learning techniques. It requires several stages to categorize the data including data collection, preprocessing, and feature extraction. One of the implementation areas of text classification includes sentiment analysis. Sentiment analysis helps to define whether an author's opinion towards a specific topic is negative or positive. Considering the fact that people's opinion directly influences the businesses and organizations it is no surprise that sentiment analysis receives a lot of attention.

Although sentiment analysis has been applied largely worldwide, research on its usage and utility on Azerbaijani language is scarce. English language has the luxury of having numerous annotated datasets as well as having well-tuned text preprocessing techniques. Different from English language, natural language processing algorithms are not improved sufficiently in Azerbaijani which is an agglutinative language and therefore requires special pre-processing approaches. Furthermore, implementation of some well-known feature extraction approaches is not experimented enough and investigating their effectiveness in natural language processing tasks namely, in sentiment analysis for an agglutinative language is one of the main objectives of the research. Additionally, the main purpose of our research is to build supervised machine learning based, automatic sentiment polarity detection system for analyzing Azerbaijani social news articles.

Proceedings of Recent Advances in Natural Language Processing, pages 703–710,
Varna, Bulgaria, Sep 2–4 2019.

2 Literature Review

As sentiment analysis is one of the prominent topics nowadays there exists vast amount of experiments applied with different methods. Sentiment analysis of text for Azerbaijani language had been investigated by Aida-zade et al. (2013). Multi machine learning algorithms had been applied for news classification in Azerbaijani language in (Suleymanov and Rustamov, 2018; Suleymanov et al., 2018; Aida-zade et al., 2018). Cambria (2016) distinguishes three main approaches in the field of sentiment analysis: knowledge-based, statistical and hybrid. The first one is the method of classifying text using rule-based algorithm to extract the sentiment. To help the organizations to improve their decision making and improve customer satisfaction Zaw and Tandayya (2018) applied rule-based algorithm called Contrast Rule-Based sentiment analysis to classify customer reviews automatically. Another rule-based algorithm is proposed by Tan et al. (2015) for classifying financial news articles. According to the research, initial stage was to determine the sentiment of each single sentence in the given financial news. Next stage was calculating positivity/negativity for the whole content of an article.

Sentiment analysis is widely used to measure the public opinion about a given topic. Li et al. (2017) investigated relationship between Dow Jones Industrial Average and public emotions. During experiment approximately 200 million twitter data was collected that mentions 30 companies which are part of the New York Stock Exchange. Researchers applied a different methodology called SMeDA-SA. The method initially extracts all uncertain sentences from the document to create the vocabulary. As a result, the research indicated that stock price of companies can be predicted with the given methodology.

One of sentiment analysis task – subjectivity detection applied before sentiment analysis for increasing accuracy performance. Rustamov et al. applied Adaptive Neuro Fuzzy Inference System (2013a), Hidden Markov Models (2013b) and Hybrid models (2018) for detection of subjectivity analysis. Same techniques had been applied for document level sentiment analysis (Rustamov et al., 2013). The method described by Araque et al. (2017) is an example of deep learning algorithm usage for sentiment analysis. Recently, deep learning is widely used to classify the text as it is capable of extracting public opinion regarding a specific topic and also works excellently with high-level features. During the research, deep learning model was developed using word embedding and linear machine learning model was implemented.

Sentiment analysis can also be used with the combination of different methodologies in the implementation of various applications. One of the researches that benefited from application of sentiment analysis is done by Rosa et al. (2019). In the research, applications collect data about a user and give recommendations based on the data produced by the user. The research proposes an approach for a recommendation system which takes user's current psychological state into account using sentiment analysis. Based on the mood of a user system sends different messages to the user including relaxing, peaceful, calm and etc.

Bansal and Srivastava (2018) applied word2vec model with machine learning algorithms to classify user reviews. The word2vec model was used to represent 400.000 consumer reviews data from Amazon as vectors. Later, the vector representation of data was given to the classifier as an input. In order to classify the data both Continuous Bag of Words (CBOW) and Skip-Gram model were implemented in combination with 4 machine learning algorithms including SVM, Naive Bayes, random forest, and logistic regression.

Severyn and Moschitti (2015) worked on an application that does sentiment analysis of tweets with deep learning models. They have applied unsupervised natural language model to initialize word embeddings which were used as distant supervised corpus in their deep learning model. The model was initialized by using pre-trained parameters of network, then trained on supervised training data from Semeval-2015. According to official test sets' result, their model ranked first in phrase-level and second in message-level tasks.

In some researches it is observed that dictionary-based approaches are also effective to extract the sentiment from text data. Nigam and Yadav (2018) divided collected tweets into lexemes and matched the words with the terms in the dictionary. Matched words were weighted so that negative word gets -1 score and positive word gets +1 score. The overall sentiment of document was calculated by subtracting the weights of positive words from weights of negative words.

3 Methodology

3.1 Data Collection

Data collection and annotation is an essential step for training supervised machine learning algorithms. Not only the collected data should be relevant to the research objective but also the annotation process should be carefully designed in order to have no inconsistencies among labeled data as they could scale up during training phase and lead to unsatisfactory results. For conducting this research using supervised machine learning techniques, approximately 30000 news articles had been collected and monitored from online websites of famous Azerbaijani newspapers. News articles under social category have been observed to contain more sentiment polarity and therefore found to be more suitable for sentiment analysis. The inter-annotator agreement had been established and an additional procedure had been implemented as a control mechanism in order to verify that the agreement had been followed during annotation. One of the main requirements of agreement was to label only the articles in which sentiment has been explicitly expressed. For instance, an article about a social event cannot be annotated for its sentiment unless author explicitly shows its social benefits or downsides. This was done to eliminate any inconsistencies emerging from subjective assessment of annotators. The control mechanism had been implemented as follows. Firstly, each news article was given a unique identifier and after random shuffling, articles were divided into small chunks each containing 500 articles. In the first run each annotator was given a chunk. After all annotators finished the first chunk, the second run began. In the second run, each annotator was given a chunk and additionally 50 more already labeled articles without their labels. By comparing these 50 articles' new labels with old ones, we could determine in which aspects annotators did not agree and made relevant adjustment to the annotation agreement to minimize the amount of disagreement in subsequent runs. In the following runs the steps of the second run were repeated until there was no chunk remaining. 12210 articles were labelled according to above mentioned rules. Among the labeled news articles, 4565 of them were labeled as positive and 7645 were labeled as negative. In the next stages, we had applied k-folds cross validation. In this method k stands for the number of repetitions of splitting data into test and train part. Cross validation is a method in machine learning that is used for assessing the result of the applied algorithm. It helped us to estimate how accurately the model would work in real case situations. By considering that we may not have satisfactory amount of data to train the model, and while splitting data into test and train we eliminate some of the train data and it may cause underfitting problem, we have applied 10-fold cross validation. After each splitting we got the accuracy score and when the splitting ended, we obtained final accuracy by calculating the average.

3.2 Data Preprocessing

Getting clean data was first step of the experiment. In order to get sentiment from data, stop words were cleaned, which have no sentiment but are highly frequent in the dataset and could decrease accuracy. Especially while applying frequency based vectorizer, noisiness of data interrupts quality of classification process.

Dataset contained XML and HTML tags since they were taken from online resources. Especially names of websites, sources of the article, dates, URL links and JS tags were present in the dataset and they affected the prediction accuracy negatively. For example, website names which end with the domain names such as ".az", ".com", ".org", ".net", ".ru", ".edu", ".gov" had been removed. In addition, the ones that started with "http", "https" and "www" and extra time tags were also deleted from data. Furthermore, all unnecessary punctuations were cleaned except semicolon and dash, since they are used in compound words in Azerbaijani language.

Finally, to eliminate difference between same words with different cases (uppercase and lowercase), all tokens had been converted to lowercase. It is needed to mention that after preprocessing, number of words decreased, and consequently, dictionary size was reduced as well which speeded up the processing of data and the classification algorithms.

3.3 Feature Extraction

In terms of natural language processing, there is an essential need to convert text data into a specific format which is appropriate for applying statistical machine learning algorithms. The process is called feature extraction and there exists different methodologies for feature extraction. One of the commonly used methods is called bag of words model (BOW) that treats each single word as a feature. This approach takes collection of documents and converts it into a list of unordered words called vocabulary (Chen et al., 2017). Next step is to create a vector representation of documents according to the size of given vocabulary where existence of each unique word defines the size. In a basic vectorizing models, it counts occurrence of each word in text and converts it to an array of real numbers.

In the domain of machine learning, there exists various vectorization approaches one of which is counting based. Also called frequency based, it provides a sparse representation of corpus of documents as a matrix. However, frequency based vectorizer has several drawbacks. Firstly, not all words have sentiment value despite how frequently it is used in the document, namely, frequency of commonly used words could shadow other more significant and sentiment containing words. Therefore, to solve this problem term-frequency and inverse-document-frequency method (tf-idf) is widely used. In addition to the number of occurrence in the document tf-idf also takes into consideration word's density in the whole corpus of documents. It consists of two parts where term frequency provides count of each word and inverse document frequency reduces value of words that are densely used in the corpus.

Another approach of bag of words model is hashing based vectorizer. It maintains vectors representing each term as an integer value. Different from others, it does not create dictionary, still having larger matrix to encode a document. tf-idf and frequency based vectorizer generates high-dimensional vector representations. Unlike them hashing based vectorizer suggests an efficient way to reduce the dimensionality of the vector. It offers default size for feature vector and provides an option to reduce and increase vector size. However, probability of collision should be considered when choosing the size. If the number of unique words in the vocabulary exceeds the feature size it could lead to collision where several unique words could map to the same integer value.

It is not necessity to present each single unique term as an input to the vectorization method. Different word combinations can be used here including unigram, bigrams, and trigrams. In unigram each word represents one feature. Additionally, we can also take combination of two words called bigrams, and combination of three words at a time called trigrams (Bhavitha et al., 2017).

4 Classifiers

4.1 Random Forest

Random forest is one of the supervised learning algorithms, which is implemented in both regression and classification problems. This classifier is a collection of recursive, tree structured models.

In decision tree, the prediction is done by splitting root training set into subsets as nodes, and each node contains output of the decision, label or condition. After sequentially choosing alternative decisions, each node recursively is split again and at the end classifier defines some rules to predict result. Conversely, in random forest, classifier randomly generate tree without defining rules.

We have implemented random forest algorithm with the different vectorizer methodologies, and n-gram models as shown in Table 1 and got different outcomes as described below.

Feature extraction	**n-grams**	**F1-Score**	**Precision**	**Recall**
Frequency based vectorizer	Unigram	93.21	91.02	95.87
	Bigram	92.15	88.47	95.97
	Trigram	89.79	82.91	97.38
Tf-idf	Unigram	93.33	90.65	95.93
	Bigram	92.27	88.96	96.61
	Trigram	89.95	83.29	97.5

Table 1: Random Forest Classifier Result

While considering the highest F1 score, we got 93.33 percent from the combination of tf-idf vectorizer and unigram model. On the other hand, when taking highest recall and precision score separately, we obtain the highest recall score of 97.38 percent. The highest recall score yields the precision of 82.91 and F1 score of 89.79 percent. Even though the recall is the highest, we got the lowest F1 score from that combination. Additionally, the highest precision score which is 91.02 percent provides 93.21 F1 score and the lowest recall score of 95.87 percent. As visible in the Table 1 the second highest F1 score of 93.21 was obtained from the combination of frequency based vectorizer and unigram model. It is noticeable that the difference between the first highest F1 score of 93.33 percent and the second highest F1 score of 93.21 is too close to each other and therefore was not statistically important. The lowest F1 score of 89.79 percent was obtained from the combination of frequency based vectorizer and trigram model.

4.2 Naïve Bayes

The second machine learning algorithm we have applied to our data set is Naive Bayes classifier. It is a probabilistic model, which is derived from Bayes Theorem that finds the probability of hypothesis activity to the given evidence activity. According to naive Bayes rule each feature is independent from each other and because of the assumption about independence, occurrence of one feature has no impact to others. Depending on the features, Bayes classifier has several forms including Gaussian Naive Bayes classifier, Multinomial Naive Bayes, and Bernoulli Naive Bayes. In this research Multinomial Naive Bayes which is mostly used for document level analysis is implemented. In this classifier, feature representation has been generated by multinomial distribution which reflects frequency of words like vectorization. Probability of an event i happening with multinomial Naive Bayes is formulated as:

$$p(x|C_k) = \frac{(\sum x_i)!}{(\prod x_i)!}\prod_i pki^{x_i}$$

In this paper we have applied Multinomial Naive Bayes classifier using alpha=1 with frequency based vectorizer and tf-idf vectorizer as described in the following table. Alpha parameter prevents model assigning null probabilities in case of 0 term frequency. Below, table 2 clearly indicates the impact of combination of different vectorization methodologies, n-gram models on the F1, recall, and precision score.

Table 2 presents that combination of frequency based vectorizer and bigram model provided the highest F1 score result of 95.47 percent. In contrast, the lowest F1 score result we got among the above combinations was 90.9 percent and was obtained from tf-idf and trigram model. As can be seen from the table the lowest F1 score yields the lowest precision score of 90.0 percent. The highest precision score we got was 97.87 which yields the lowest recall score of 85.4 percent. The highest F1 score of 95.47 percent was obtained from the combination of frequency based vectorizer and bigram model. Additionally, it is noticeable from the table that while considering tf-idf results the highest F1 score we got 94.66 percent which was obtained from bigram model. This combination yields the recall score of 98.24 percent which is the second highest recall score and precision score of 91.34 which is the lowest precision score.

Feature extraction	**n-grams**	**F1-Score**	**Precision**	**Recall**
Frequency based vectorizer	Unigram	94.87	95.87	93.9
	Bigram	95.47	97.24	93.77
	Trigram	91.2	97.87	85.4
Tf-idf	Unigram	94.1	91.54	96.97
	Bigram	94.66	91.34	98.24
	Trigram	90.9	84.2	98.75

Table 2: Multinomial Naive Bayes Classifier Result

4.3 Support Vector Machine

The third classification algorithm that we have applied is SVM. the purpose of SVM classification algorithm is to define optimal hyperplane in N dimensional space to separate the data points from each other. N dimensional space here is number of features:

$$h(x_i) = sign(w * x_i + b)$$

$$min_w \lambda \, ||w||^2 + \sum_{i=1}^{n}(1 - y_i < x_i, w >) \quad (1)$$

Equation (1) describes the calculation of cost function and hypothesis for SVM. One of the important terms used in SVM is kernel parameter. When the data is so huge and hardly computational, kernel is used to speed up and optimize SVM. There are different types of the kernel parameter such as linear kernel, polynomial kernel, rbf kernel, and sigmoid kernel. In this research linear kernel parameter is used. In the research, we analyzed accuracy of SVM classification algorithm with different vectorization and n-gram models which is described in the following table.

Feature extraction	n-grams	F1-Score	Precision	Recall
Frequency based vectorizer	Unigram	95.51	95.4	95.63
	Bigram	95.45	94.17	96.76
	Trigram	92.82	88.73	97.31
Tf-idf	Unigram	96.79	96.48	97.1
	Bigram	95.9	94.45	97.41
	Trigram	93.35	89.19	97.93

Table 3: Linear SVM Result

Table 3 depicts the F1, recall, and precision results for SVM classifier using various feature selection and n-grams models. As can be seen from the table the best F1 score we got 96.79 percent was from the combination of tf-idf vectorizer and unigram model. While considering frequency based vectorizer the highest F1 score which is 95.51 percent was obtained from unigram model. When comparing the F1 score of unigram and bigram models we do not observe huge difference between results. Recall values for the SVM classifier ranged from 95.63 percent to 97.93 percent. The highest recall score was 97.93 which was obtained from the combination of tf-idf and trigram model. As described in table the lowest F1 score 92.82 percent and was gained from trigram model and frequency-based vectorization.

4.4 Data Skewness and Classifier Comparisons

As described in data collection section, the dataset was skewed towards negative samples with 7.6k negative samples and 4.5k positive samples. Having skewness in a dataset can be considered normal as dataset is formed as a result of some natural phenomena which can inherently be biased towards some category or other. In our case, news agencies play the role of data source which seems to be biased towards generating negative news articles. This could be explained by the fact that this type of articles is catchier and preferred more by the readers.

Having data skewness can have a direct impact on the results of a machine learning model and gaining further insights can contribute to obtaining higher results. Therefore, the impact of skewness on our dataset had been researched. Firstly, we examined the precision and recall score per class to see if skewness has a significant impact on the results.

Class	Recall	Precision
Positive	93.58	94.82
Negative	96.89	96.23

Table 4: Per class precision and recall scores of SVM classifier

Table 4 demonstrates that skewness in the dataset affects the performance of the classifier. Therefore, we explored several approaches for optimizing the results while working with skewed dataset. The considerable change came from the application of adjusting sample weights inversely to the frequencies of each class in SVM classifier. By this way, the classifier is penalized more for the mistakes made on samples from underrepresented class, namely positive. This allowed to elevate the recall score to 95.14 percent for the positive class and increase the precision score to 97.07 percent for the negative class, while slightly lowering the recall to 95.95 percent.

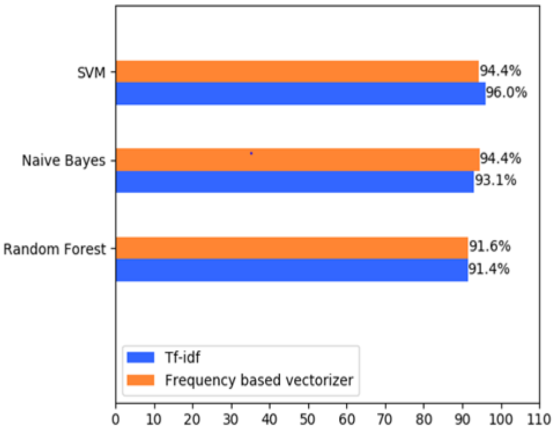


Figure 1: Accuracy of SVM, Naive Bayes, random forest classifiers with vectorization methods

Figure 1 demonstrates the accuracy results obtained from implementation of tf-idf and frequency-based vectorization methods with each classification algorithm. As can be seen from the figure, combination of SVM and tf-idf outperforms other classifiers with the 96 percent accuracy. The minimum accuracy was obtained from the combination of random forest classifier and tf-idf vectorizer with 91.4 percent accuracy. While considering accuracy result it is observable that accuracy range changes between 91 percent and 96 percent. Random forest classifier does not offer huge difference between two vectorization methods as there is only 0.2 percent accuracy difference between tf-idf and frequency-based vectorization methods. When analyzing Naive Bayes classifier, the best accuracy we got was 94.4 percent and was obtained from the frequency based vectorizer.

5 Conclusion

In conclusion, we experimented with three machine learning algorithms on news articles in Azerbaijani language. Comparing one classifier with another, depending on the n-gram model and vectorization method we obtained different results for different combinations. According to the research the highest F1-score we got was 96.79 percent with the implementation of SVM on tf-idf vectorizer and unigram model. Research also revealed that Naive Bayes classifier give its best result 95.47 percent with the combination of frequency based vectorizer and bigram model, while random forest acquires highest F1-score 93.33 percent by using tf-idf based feature extraction and unigram model.

Additionally, for future work we plan to improve our research by enlarging our dataset, adding neutral class, applying rule-based algorithms and observing their performance in our dataset. In addition to them, we are going to apply word embedding with different classification algorithms.

Acknowledgments

This work has been carried out in Center for Data Analytics Research at ADA University in partnership with E-GOV Development Center.

References

Aida-zade Kamil, Samir Rustamov, Mark A. Clements and Elshan Mustafayev. 2018. Adaptive Neuro-Fuzzy Inference System for Classification of Texts. In *Zadeh L., Yager R., Shahbazova S., Reformat M., Kreinovich V. (eds) Recent Developments and the New Direction in Soft-Computing Foundations and Applications. Studies in Fuzziness and Soft Computing*. Springer, 361(2018):63-70. https://doi.org/10.1007/978-3-319-75408-6_6

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, pages 464-469. https://doi.org/10.18653/v1/s15-2079

Ayda-zade K., Rustamov S., Mustafayev E. 2013. Sentiment Analysis: Hybrid Approach. In *Transactions of Azerbaijan National Academy of Sciences."Informatics and control problems".* Volume XXXIII №6., p. 100-108.

Barkha Bansala and Sangeet Srivastava. 2018. Sentiment classification of online consumer reviews using word vector representations. *Procedia Computer Science,* 132:1147-1153. https://doi.org/10.1016/j.procs.2018.05.029

Bing Li, Keith C.C. Chan, Carol Ou and Sun Ruifeng. 2017. Discovering public sentiment in social media for predicting stock movement of publicly listed companies. *Information Systems,* 69(1):81-92. https://doi.org/10.1016/j.is.2016.10.001

B. K. Bhavitha, Anisha P. Rodrigues and Niranjan N. Chiplunkar. 2017. Comparative study of machine learning techniques in sentiment analysis. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT).* IEEE, pages 216-221. https://doi.org/10.1109/ICICCT.2017.7975191

D Shubham, P Mithil, Meesala Shobharani and S Sumathy. 2017. Aspect level sentiment analysis using machine learning. *IOP Conference Series: Materials Science and Engineering,* 263:042009. https://doi.org/10.1088/1757-899x/263/4/042009

Erik Cambria. 2016. Affective Computing and Sentiment Analysis. *IEEE Intelligent Systems,* 31(2):102-107. https://doi.org/10.1109/MIS.2016.31

Evangelos Psomakelis, Konstantinos Tserpes, Dimosthenis Anagnostopoulos and Theodora Varvarigou. 2014. Comparing Methods for Twitter Sentiment Analysis. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval - Volume 1: KDIR*. SciTePress, pages 225-232. https://doi.org/10.5220/0005075302250232

Freha Mezzoudj and Abdelkader Benyettou. 2018. An empirical study of statistical language models: N-gram language models vs. neural network language models. *International Journal of Innovative Computing and Applications*, 9(4):189. https://doi.org/10.1504/ijica.2018.10016827

Khanvilkar Gayatri and Vor, Deepali. 2018. Sentiment Analysis for Product Recommendation Using Random Forest. *International Journal of Engineering and Technology(UAE)*, 7(3):87-89. https://doi.org/10.14419/ijet.v7i3.3.14492

Li I. Tan, Wai S. Phang, Kim O. Chin and Patricia Anthony. 2015. Rule-Based Sentiment Analysis for Financial News. In *IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, pages 1601-1606. https://doi.org/10.1109/SMC.2015.283

Mirsa Karim and Smija Das. 2018. Sentiment Analysis on Textual Reviews. *IOP Conference Series: Materials Science and Engineering*, 396:012020. https://doi.org/10.1088/1757-899x/396/1/012020

Myint Zaw and Pichaya Tandayya. 2018. Multi-level Sentiment Information Extraction Using the CRbSA Algorithm. In *15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, pages 1-6. https://doi.org/10.1109/jcsse.2018.8457328

Nitika Nigam and Divakar Yadav. 2018. Lexicon-Based Approach to Sentiment Analysis of Tweets Using R Language. *Singh M., Gupta P., Tyagi V., Flusser J., Ören T. (eds) Advances in Computing and Data Sciences. ICACDS 2018. Communications in Computer and Information Science*, 905:154-164. https://doi.org/10.1007/978-981-13-1810-8_16

Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada and Carlos A. Iglesias. 2017. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77(1):236-246. https://doi.org/10.1016/j.eswa.2017.02.002

Po-Hao Chen, Hanna Zafar, Maya Galperin-Aizenberg and Tessa Cook. 2017. Integrating Natural Language Processing and Machine Learning Algorithms to Categorize Oncologic Response in Radiology Reports. *Journal of Digital Imaging*, 31(2):178-184. https://doi.org/10.1007/s10278-017-0027-x

Renata Lopes Rosa, Gisele Maria Schwartz, Wilson Vicente Ruggiero, Demóstenes Zegarra Rodríguez. 2019. A Knowledge-Based Recommendation System That Includes Sentiment Analysis and Deep Learning. *IEEE Transactions on Industrial Informatics*,15(4):2124-2135. https://doi.org/10.1109/tii.2018.2867174

Samir Rustamov. 2018. A Hybrid System for Subjectivity Analysis. *Advances in Fuzzy Systems*. Volume 2018: 9. https://doi.org/10.1155/2018/2371621

Samir Rustamov, Elshan Mustafayev and Mark A. Clements. 2013. An Application of Hidden Markov Models in subjectivity analysis. In *2013 7th International Conference on Application of Information and Communication Technologies*. IEEE, pages 64-67. https://doi.org/10.1109/ICAICT.2013.6722756

Samir Rustamov, Elshan Mustafayev and Mark A. Clements. 2013. Sentiment Analysis using Neuro-Fuzzy and Hidden Markov Models of Text. In *2013 Proceedings of IEEE Southeastcon*. IEEE, pages 1-6. https://doi.org/10.1109/SECON.2013.6567382

Samir Rustamov and Mark Clements. 2013. Sentence-Level Subjectivity Detection Using Neuro-Fuzzy Model. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, pages 108–114, Atlanta, USA. https://www.aclweb.org/anthology/W13-1615

Sara Rosenthal, Noura Farra and Preslav Nakov. 2017. Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, pages 502-518. https://doi.org/10.18653/v1/S17-2088

Umid Suleymanov and Samir Rustamov. 2018. Automated News Categorization using Machine Learning methods. *IOP Conference Series: Materials Science and Engineering*, 459:012006. https://doi.org/10.1088/1757-899x/459/1/012006

Umid Suleymanov, Samir Rustamov, Murad Zulfugarov, Orkhan Orujov, Nadir Musayev and Azar Alizade. 2018. Empirical Study of Online News Classification Using Machine Learning Approaches. In *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, pages 1-6. https://doi.org/10.1109/ICAICT.2018.8747012

Yashaswini Hegde and S.K. Padma. 2017. Sentiment Analysis Using Random Forest Ensemble for Mobile Product Reviews in Kannada. In *2017 IEEE 7th International Advance Computing Conference (IACC)*. IEEE, pages 777-782. https://doi.org/10.1109/IACC.2017.0160

Inforex — a Collaborative System for Text Corpora Annotation and Analysis Goes Open

Michał Marcińczuk and Marcin Oleksy
G4.19 Research Group
Department of Computational Intelligence
Faculty of Computer Science and Management
Wrocław University of Science and Technology, Wrocław, Poland
{michal.marcinczuk,marcin.oleksy}@pwr.edu.pl

Abstract

In the paper we present the latest changes introduce to Inforex — a web-based system for qualitative and collaborative text corpora annotation and analysis. One of the most important news is the release of source codes. Now the system is available on the GitHub repository (https://github.com/CLARIN-PL/Inforex) as an open source project. The system can be easily setup and run in a Docker container what simplifies the installation process. The major improvements include: semi-automatic text annotation, multilingual text preprocessing using CLARIN-PL web services, morphological tagging of XML documents, improved editor for annotation attribute, batch annotation attribute editor, morphological disambiguation, extended word sense annotation. This paper contains a brief description of the mentioned improvements. We also present two use cases in which various Inforex features were used and tested in real-life projects.

1 Introduction

Development and evaluation of tools for various natural language processing task (named entity recognition, sentiment analysis, cyberbully detection and many other) require dedicated resources in a form of manually or semi-automatically annotated corpora. Corpus-based studies in the domain of Digital Humanities also require a support in the form of specialized tools and system. Both create a demand on development of tools and systems qualitative text corpora management, annotation, analysis and visualization.

Inforex is one of several web-based systems for text corpora annotation which is being developed as an open source project. The other well-known systems include, but are not limited to, WebAnno 3.0 (de Castilho et al., 2016), Brat (Stenetorp et al., 2012) and Anafora (Chen and Styler, 2013). Comparing to the other systems Inforex has some distinct features, including: support for untokenized and tokenized documents, support for both plain text and XML documents (XML tags are used to format the document layout) and integration with CLARIN-PL web services (utilizes on-demand morphological tagging).

In Section 2 we present the basic characteristic of the Inforex system. In Section the 3 we present the recents improvements and new features implemented in the system. In the Section 4 we present two projects in which the latest features were utilized.

2 Inforex Overview

Inforex is a web-based system for text corpora management, annotation and analysis. Since 2018 it is available as an open source project on the GitHub repository and is a part of the Polish CLARIN infrastructure[1] — it is integrated with the official repository for language resources in Polish CLARIN[2]. From the user perspective Inforex requires only a modern web browser to use the system.

Inforex offers several features for collaborative work on a single corpus, including concurrent access to data stored in the central database, role-based access to different modules, flag-based mechanism to track the process of various types of tasks. It support text cleanup, mention annotation, relations between annotations, morpholog-

[1] https://inforex.clarin-pl.eu
[2] https://clarin-pl.eu/dspace/

ical tagging, annotation attributes, metadata and many others. A more comprehensive list of functions can be found in (Marcinczuk et al., 2017).

3 Recent Changes and Improvements

3.1 Open Source Project

After 10 years of development the project has been finally released as an open source project. The source codes are available under the LGPL license and can be obtained from https://github.com/CLARIN-PL/Inforex.

3.2 Easy Installation

The installation process was simplified by converting the system and all required components to run withing a set of Docker containers[3] defined in a Compose file. The Compose[4] file defines four containers: (1) **www** — web server running the Inforex application with background services (see Section 3.3), (2) **db** — MySQL database server, (3) **liquibase** — Liquibase database schema control and (4) **phpmyadmin** — web-based access to the database (for development and maintenance purposes).

A new installation of Inforex boils down to running the following two lines of code:

```
sudo apt-get install composer \
        docker docker-compose
./docker-dev-up.sh
```

3.3 Background Processes

Time consuming tasks, like corpus export or morphological tagging, are handled by processes running in the background. That's how we avoid the web server timeouts and handle task queuing. The background processes have been added to the Docker container with web server and they are automatically run on the container startup.

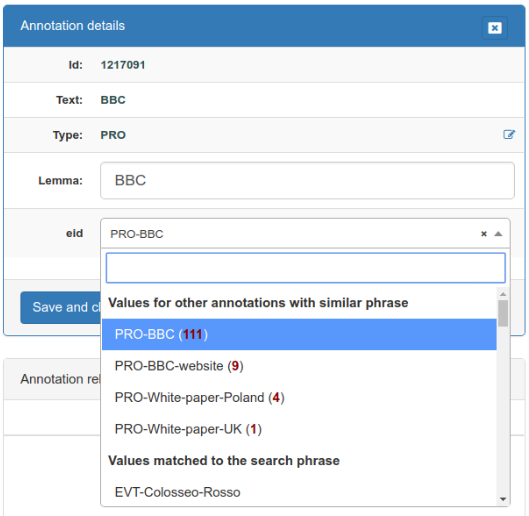
3.4 Multilingual Morphological Tagging

Inforex uses CLARIN-PL Web Service API[5] (Walkowiak, 2018) to facilitate the on-demand morphological document tagging. CLARIN-PL WS API provides access to morphological taggers for 11 languages. Seven of them are available from Inforex, i.e. Polish, English, German, Russian, Hebrew, Czech and Bulgarian (see Figure 1). Inforex automatically choose the language specific tagger based on the document language set in the metadata.

3.5 Extended Annotation Attribute Editor

We have extended the annotation attribute editor to handle dictionary-based attributes with a large number of possible values (see Figure 2). The improvements include the following:

- Filtering of the list of values,
- Feature to add a new element to the dictionary directly from the value picker level.
- Suggestions based on values assigned to other annotations. We have implemented two heuristic of generating the candidates with different levels of certainty, i.e.:
  - values for other annotations matched by the text form with the Soundex algorithm[6]. The list of candidates is sorted by their frequency,
  - attribute values matched by the annotation text (full or partial matching).



Id: 1217091
Text: BBC
Type: PRO
Lemma: BBC
eid PRO-BBC

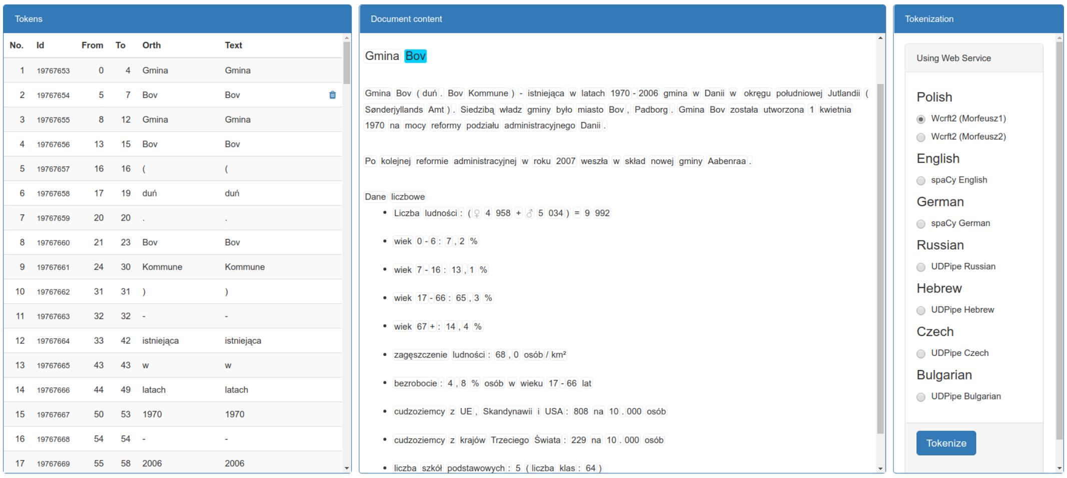
Values for other annotations with similar phrase
PRO-BBC (111)
PRO-BBC-website (9)
PRO-White-paper-Poland (4)
PRO-White-paper-UK (1)
Values matched to the search phrase
EVT-Colosseo-Rosso

Figure 2: Extended annotation attribute editor

3.6 Batch Annotation Attribute Editor

Up to now the modification of annotation attributes was available only from the *Annotator* perspective using the annotation editor (see Figure 2). When an user had to modify an attribute for each annotation the only way was to go through all the annotations one by one. This process was time

[3] https://www.docker.com/
[4] https://docs.docker.com/compose/
[5] http://ws.clarin-pl.eu/tagerml.shtml
[6] https://www.archives.gov/research/census/soundex.html



No.	Id	From	To	Orth	Text
1	19767653	0	4	Gmina	Gmina
2	19767654	5	7	Bov	Bov
3	19767655	8	12	Gmina	Gmina
4	19767656	13	15	Bov	Bov
5	19767657	16	16	(	(
6	19767658	17	19	duń	duń
7	19767659	20	20	.	.
8	19767660	21	23	Bov	Bov
9	19767661	24	30	Kommune	Kommune
10	19767662	31	31	)	)
11	19767663	32	32	-	-
12	19767664	33	42	istniejąca	istniejąca
13	19767665	43	43	w	w
14	19767666	44	49	latach	latach
15	19767667	50	53	1970	1970
16	19767668	54	54	-	-
17	19767669	55	58	2006	2006

Gmina Bov

Gmina Bov (duń . Bov Kommune) - istniejąca w latach 1970 - 2006 gmina w Danii w okręgu południowej Jutlandii (Sønderjyllands Amt) . Siedzibą władz gminy było miasto Bov , Padborg . Gmina Bov została utworzona 1 kwietnia 1970 na mocy reformy podziału administracyjnego Danii .

Po kolejnej reformie administracyjnej w roku 2007 weszła w skład nowej gminy Aabenraa .

Dane liczbowe

- Liczba ludności : (♀ 4 958 + ♂ 5 034) = 9 992
- wiek 0 - 6 : 7 , 2 %
- wiek 7 - 16 : 13 , 1 %
- wiek 17 - 66 : 65 , 3 %
- wiek 67 + : 14 , 4 %
- zagęszczenie ludności : 68 , 0 osób / km²
- bezrobocie : 4 , 8 % osób w wieku 17 - 66 lat
- cudzoziemcy z UE , Skandynawii i USA : 808 na 10 . 000 osób
- cudzoziemcy z krajów Trzeciego Świata : 229 na 10 . 000 osób
- liczba szkół podstawowych : 5 (liczba klas : 64)

Using Web Service

Polish
● Wcrft2 (Morfeusz1)
○ Wcrft2 (Morfeusz2)
English
○ spaCy English
German
○ spaCy German
Russian
○ UDPipe Russian
Hebrew
○ UDPipe Hebrew
Czech
○ UDPipe Czech
Bulgarian
○ UDPipe Bulgarian

Tokenize

Figure 1: Document tokenization perspective

consuming and error-prone because it was easy to miss some annotations. To overcame these problems we have created a page for batch annotation attribute modification. The page consists of two main components, i.e. a document content with annotation preview and a table with annotations with their attribute (see Figure 3).

3.7 Document Auto Annotation

This feature was designed to reduce user effort in annotating repeatable phrases in and across documents. *Auto annotation* works by annotating in given documents all phrases that were already annotated in other documents. This feature works for both untokenized and tokenized documents, however we advise to use it on tokenized documents as the phrases are aligned with token boundaries and we avoid matching of incomplete words. After running *auto annotation* the new annotations are presented to the user for verification. User can decide whether given annotation is correct, incorrect or the annotation type needs a change (see Figure 4). The discarded annotation are stored in the system for future run of *auto annotation*. During the next use of *auto annotation* the new annotations which were previously discarded are ignored.

3.8 Lemma and Attribute Auto Fill

These features were designed to reduce user effort in setting annotation lemmas and attribute values. They both works in a similar way — for each annotation in the document the lemma or attribute value is set based on other annotations in the corpus. For lemma we collect annotations with the same text form and category. For attribute value we collect annotations with the same text form or lemma and category. In case of ambiguity, i.e. there are more than one possible value of lemma or attribute, the value remain empty and the user has to fill it manually. The lemma auto fill feature is available in the *Annotation lemmas* perspective and the attribute auto fill feature is available in the *Annotation attributes* perspective.

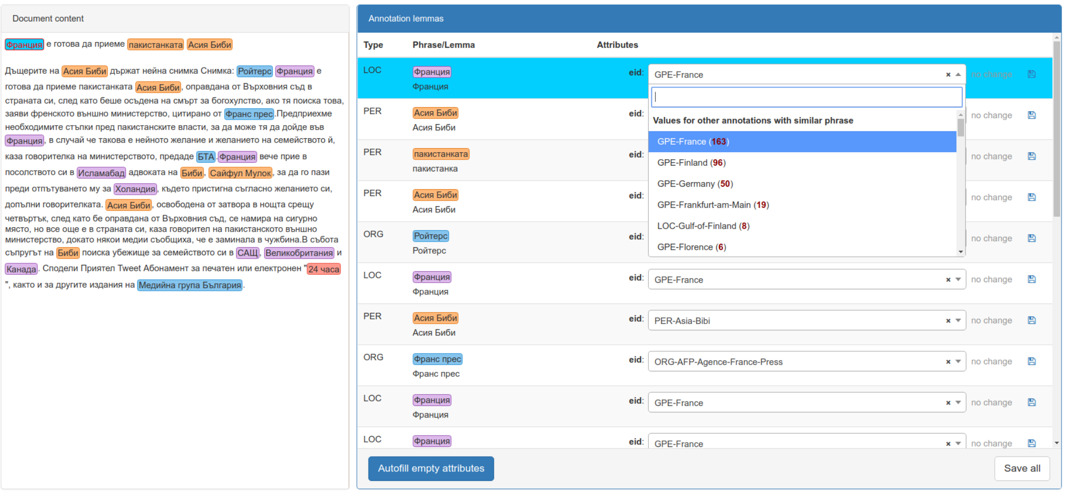
3.9 Tokenization of XML documents

Inforex allows to store documents in one of the two formats: plain text or XML. The XML format is used to encode document structure, like in the KPWr (Broda et al., 2012) and PCSN (Marcińczuk et al., 2011) corpora. During tagging the XML tags should be ignored and only the content should be processed. Thus, we made the tokenization process to be aware of the document format (see Figure 1). For XML format the document content is cleaned from XML tags, than the content is processed by the tagging service and at the end the tokenization is aligned with the original XML document.

3.10 Annotation Attribute Browser

The attribute value browser (see Figure 5) allows to browse corpus annotations by given attribute value. The page consists of three elements:

- View configuration — provides a set of filters, including: shared attribute, document language and subcorpus,
- Attribute values assigned to annotations — list of values their frequency,



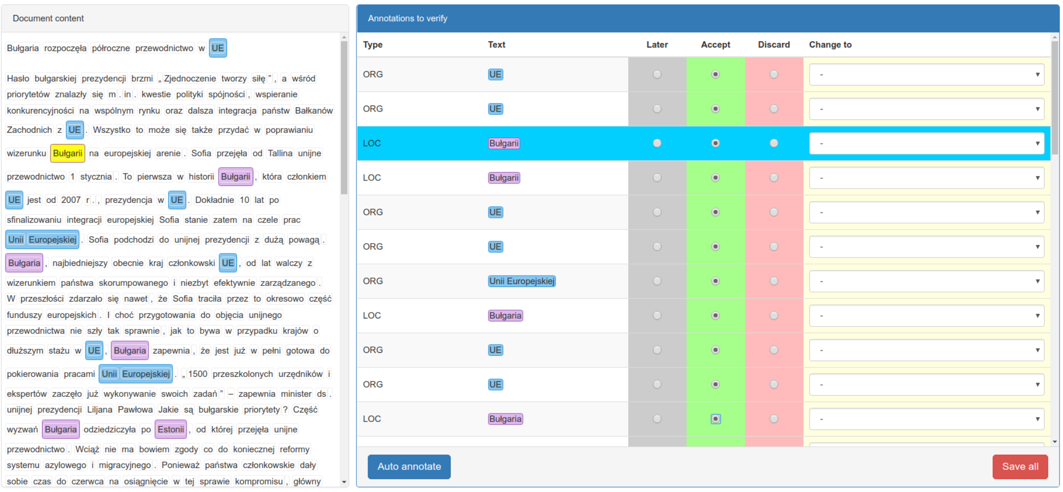
Франция е готова да приеме пакистанката Асия Биби

Дъщерите на Асия Биби държат нейна снимка Снимка: Ройтерс Франция е готова да приеме пакистанката Асия Биби, оправдана от Върховния съд в страната си, след като беше осъдена на смърт за богохулство, ако тя поиска това, заяви френското външно министерство, цитирано от Франс прес.Предприехме необходимите стъпки пред пакистанските власти, за да може тя да дойде във Франция, в случай че такова е нейното желание и желанието на семейството й, каза говорителка на министерството, предаде БТА.Франция вече прие в посолството си в Исламабад адвоката на Биби, Сайфул Мулок, за да го пази преди отпътуването му за Холандия, където пристигна съгласно желанието си, допълни говорителката. Асия Биби, освободена от затвора в нощта срещу четвъртък, след като бе оправдана от Върховния съд, се намира на сигурно място, но все още е в страната си, каза говорител на пакистанското външно министерство, докато някои медии съобщиха, че е заминала в чужбина.В събота съпругът на Биби поиска убежище за семейството си в САЩ, Великобритания и Канада. Сподели Приятел Tweet Абонамент за печатен или електронен "24 часа ", както и за другите издания на Медийна група България.

Type	Phrase/Lemma	Attributes		
LOC	Франция Франция	eid: GPE-France × ▲	no change	🖫
PER	Асия Биби Асия Биби	eid:	no change	🖫
PER	пакистанката пакистанка	eid:	no change	🖫
PER	Асия Биби Асия Биби	eid:	no change	🖫
ORG	Ройтерс Ройтерс	eid:	no change	🖫
LOC	Франция Франция	eid: GPE-France × ▼	no change	🖫
PER	Асия Биби Асия Биби	eid: PER-Asia-Bibi × ▼	no change	🖫
ORG	Франс прес Франс прес	eid: ORG-AFP-Agence-France-Press × ▼	no change	🖫
LOC	Франция Франция	eid: GPE-France × ▼	no change	🖫
LOC	Франция	eid: GPE-France × ▼	no change	🖫

Values for other annotations with similar phrase
GPE-France (163)
GPE-Finland (96)
GPE-Germany (50)
GPE-Frankfurt-am-Main (19)
LOC-Gulf-of-Finland (8)
GPE-Florence (6)

Figure 3: Batch annotation attribute editor



Bułgaria rozpoczęła półroczne przewodnictwo w UE

Hasło bułgarskiej prezydencji brzmi „ Zjednoczenie tworzy siłę ” , a wśród priorytetów znalazły się m . in . kwestie polityki spójności , wspieranie konkurencyjności na wspólnym rynku oraz dalsza integracja państw Bałkanów Zachodnich z UE . Wszystko to może się także przydać w poprawianiu wizerunku Bułgarii na europejskiej arenie . Sofia przejęła od Tallina unijne przewodnictwo 1 stycznia . To pierwsza w historii Bułgarii , która członkiem UE jest od 2007 r . , prezydencja w UE . Dokładnie 10 lat po sfinalizowaniu integracji europejskiej Sofia stanie zatem na czele prac Unii Europejskiej . Sofia podchodzi do unijnej prezydencji z dużą powagą . Bułgaria , najbiedniejszy obecnie kraj członkowski UE , od lat walczy z wizerunkiem państwa skorumpowanego i niezbyt efektywnie zarządzanego . W przeszłości zdarzało się nawet , że Sofia traciła przez to okresowo część funduszy europejskich . I choć przygotowania do objęcia unijnego przewodnictwa nie szły tak sprawnie , jak to bywa w przypadku krajów o dłuższym stażu w UE , Bułgaria zapewnia , że jest już w pełni gotowa do pokierowania pracami Unii Europejskiej . „ 1500 przeszkolonych urzędników i ekspertów zaczęło już wykonywanie swoich zadań ” – zapewnia minister ds . unijnej prezydencji Liljana Pawłowa Jakie są bułgarskie priorytety ? Część wyzwań Bułgaria odziedziczyła po Estonii , od której przejęła unijne przewodnictwo . Wciąż nie ma bowiem zgody co do koniecznej reformy systemu azylowego i migracyjnego . Ponieważ państwa członkowskie dały sobie czas do czerwca na osiągnięcie w tej sprawie kompromisu , główny

Type	Text	Later	Accept	Discard	Change to
ORG	UE	○	●	○	- ▾
ORG	UE	○	●	○	- ▾
LOC	Bułgarii	○	●	○	- ▾
LOC	Bułgarii	○	●	○	- ▾
ORG	UE	○	●	○	- ▾
ORG	UE	○	●	○	- ▾
ORG	Unii Europejskiej	○	●	○	- ▾
LOC	Bułgaria	○	●	○	- ▾
ORG	UE	○	●	○	- ▾
ORG	UE	○	●	○	- ▾
LOC	Bułgaria	○	●	○	- ▾

Figure 4: Auto annotation and candidate verification perspective

- Annotations with the selected value.

3.11 Export of Morphological Tagging and Annotations Agreements

For tokenized document Inforex can store up to three layers of morphological tags:

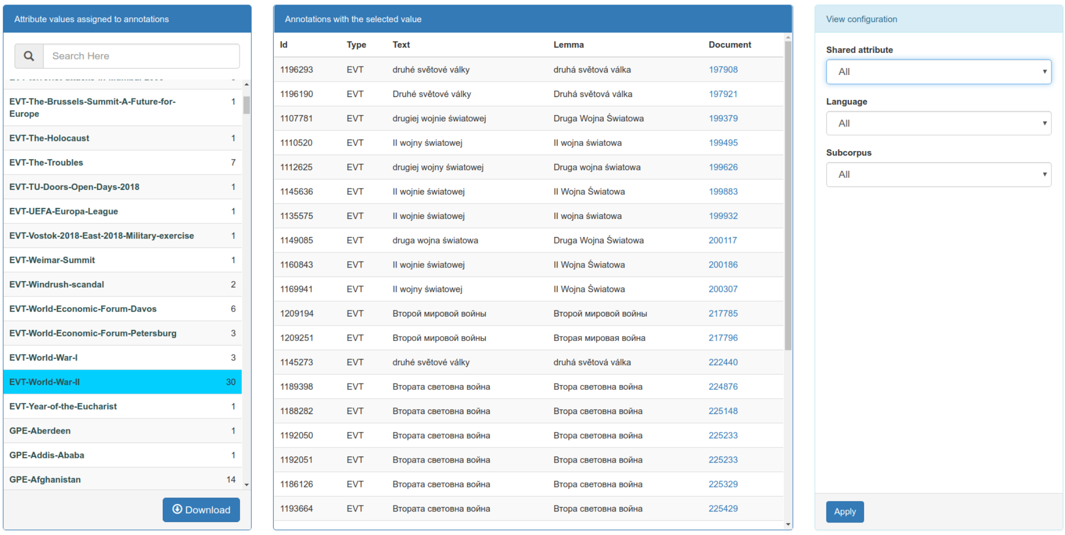
- *Tagger* — tags produced by a tool,
- *Agreement* — tags entered by a user in the agreement mode,
- *Final* — tags approved by the super user.

During export it is possible to define which layer of tags should be exported. It is possible to choose one of the following options (see Figure 6):

- *Final or tagger (if final not present)* — export the *final* tags. For tokens which does not have the *final* tag a *tagger* tag is taken.
- *Final* — export only the *final* tags. If there are tokens without final tags than the missing tags are reported as errors.
- *User (agreement)* — export tags created by selected user. For tokens which does not contain user agreement tags the tagger tags are taken.
- *Tagger* — export tagger tags.

3.12 Improved Support for Word Sense Annotation

The existing mechanism for word sense annotation was limited to a single set of words and their senses (Marcińczuk et al., 2012). We have removed the limitation and allow to define and use any number of sets of word senses in the WSD



Attribute values assigned to annotations

Search Here

EVT-The-Brussels-Summit-A-Future-for-Europe	1
EVT-The-Holocaust	1
EVT-The-Troubles	7
EVT-TU-Doors-Open-Days-2018	1
EVT-UEFA-Europa-League	1
EVT-Vostok-2018-East-2018-Military-exercise	1
EVT-Weimar-Summit	1
EVT-Windrush-scandal	2
EVT-World-Economic-Forum-Davos	6
EVT-World-Economic-Forum-Petersburg	3
EVT-World-War-I	3
EVT-World-War-II	30
EVT-Year-of-the-Eucharist	1
GPE-Aberdeen	1
GPE-Addis-Ababa	1
GPE-Afghanistan	14

Download

Annotations with the selected value

Id	Type	Text	Lemma	Document
1196293	EVT	druhé světové války	druhá světová válka	197908
1196190	EVT	Druhé světové války	Druhá světová válka	197921
1107781	EVT	drugiej wojnie światowej	Druga Wojna Światowa	199379
1110520	EVT	II wojny światowej	II wojna światowa	199495
1112625	EVT	drugiej wojny światowej	Druga wojna światowa	199626
1145636	EVT	II wojnie światowej	II Wojna Światowa	199883
1135575	EVT	II wojnie światowej	II wojna światowa	199932
1149085	EVT	druga wojna światowa	Druga Wojna Światowa	200117
1160843	EVT	II wojnie światowej	II Wojna Światowa	200186
1169941	EVT	II wojny światowej	II Wojna Światowa	200307
1209194	EVT	Второй мировой войны	Второй мировой войны	217785
1209251	EVT	Второй мировой войны	Вторая мировая война	217796
1145273	EVT	druhé světové války	druhá světová válka	222440
1189398	EVT	Втората световна война	Втора световна война	224876
1188282	EVT	Втората световна война	Втора световна война	225148
1192050	EVT	Втората световна война	Втора световна война	225233
1192051	EVT	Втората световна война	Втора световна война	225233
1186126	EVT	Втората световна война	Втора световна война	225329
1193664	EVT	Втората световна война	Втора световна война	225429

View configuration

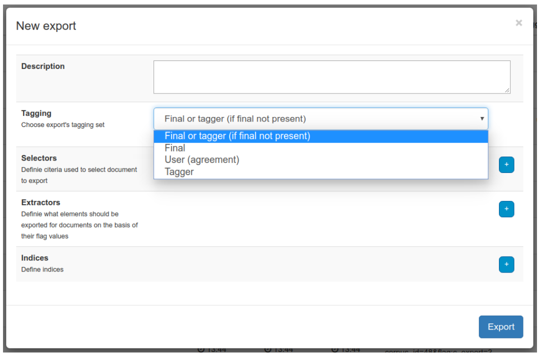
Shared attribute: All

Language: All

Subcorpus: All

Apply

Figure 5: Annotation attribute browser



New export

Description

Tagging
Choose export's tagging set

Final or tagger (if final not present)
- Final or tagger (if final not present)
- Final
- User (agreement)
- Tagger

Selectors
Definie citeria used to select document to export

Extractors
Definie what elements should be exported for documents on the basis of their flag values

Indices
Define indices

Export

Figure 6: Export configuration dialog window

perspective. We also added the option to annotate the word senses in an agreement mode for further agreement (see Figure 7). Finally, we have imported all lexical units with their senses from Słowosieć 3.2 (Piasecki et al., 2016) as an annotation set to Inforex.

3.13 Morphological Agreement

The last feature is support for morphological disambiguation agreement. Inforex provides a page with morphological tag agreement across a given set of documents (see Figure 8). The agreement is presented in a numerical form for each documents in the set and after a specific document a list of disagreements is presented. This feature is complemented by a document perspective for comparing and choosing the final morphological tags (see Figure 9).

4 Case Studies

In this section we present two uses cases in which various features of Inforex were used in real-life projects.

4.1 BSNLP 2019 Shared Task

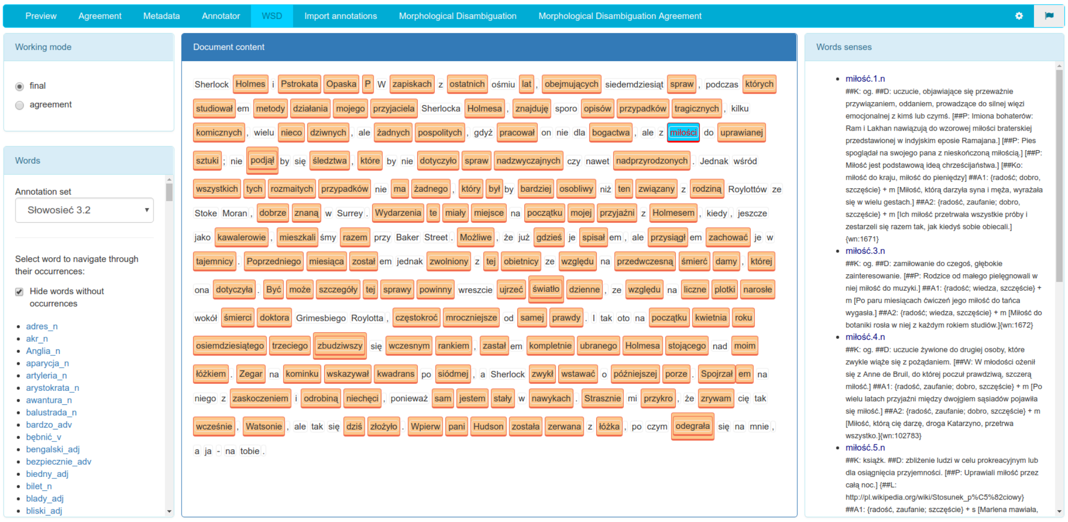
Inforex was used to create the training and testing datasets for the need of 2nd Edition of the Shared Task on Multilingual Named Entity Recognition for Slavic languages[7]. The task aims at recognizing mentions of named entities in news articles in Slavic languages, their lemmatization, and cross-language matching.

More than 10 people were involved in the annotation process for four languages, i.e. Polish, Czech, Russian and Bulgarian. There were 1–3 annotators per language. The annotation process consists of four main steps:

1. Selection of relevant documents — the document were automatically crawled and uploaded to Inforex, therefore some of them were duplicates or text not relevant to the subcorpus topic. The selected documents were marked with a flag *Valid content*.

2. Annotation of named entity mentions — the same set of five annotatation types was used to annotated all the selected documents. For Polish and Czech the annotators utilized the auto annotate feature described in Section 3.7 and we were able to evaluate the usability of

[7] http://bsnlp.cs.helsinki.fi/shared_task.html


Preview
Agreement
Metadata
Annotator
WSD
Import annotations
Morphological Disambiguation
Morphological Disambiguation Agreement

Working mode

● final

○ agreement

Words

Annotation set

Słowosieć 3.2 ▾

Select word to navigate through their occurrences:

☑ Hide words without occurrences

- adres_n
- akr_n
- Anglia_n
- aparycja_n
- artyleria_n
- arystokrata_n
- awantura_n
- balustrada_n
- bardzo_adv
- bębnić_v
- bengalski_adj
- bezpiecznie_adv
- biedny_adj
- bilet_n
- blady_adj
- bliski_adj

Document content

Sherlock Holmes i Pstrokata Opaska P W zapiskach z ostatnich ośmiu lat , obejmujących siedemdziesiąt spraw , podczas których studiował em metody działania mojego przyjaciela Sherlocka Holmesa , znajduję sporo opisów przypadków tragicznych , kilku komicznych , wielu nieco dziwnych , ale żadnych pospolitych , gdyż pracował on nie dla bogactwa , ale z miłości do uprawianej sztuki ; nie podjął by się śledztwa , które by nie dotyczyło spraw nadzwyczajnych czy nawet nadprzyrodzonych . Jednak wśród wszystkich tych rozmaitych przypadków nie ma żadnego , który był by bardziej osobliwy niż ten związany z rodziną Roylottów ze Stoke Moran , dobrze znaną w Surrey . Wydarzenia te miały miejsce na początku mojej przyjaźni z Holmesem , kiedy , jeszcze jako kawalerowie , mieszkali śmy razem przy Baker Street . Możliwe , że już gdzieś je spisał em , ale przysiągł em zachować je w tajemnicy . Poprzedniego miesiąca został em jednak zwolniony z tej obietnicy ze względu na przedwczesną śmierć damy , której ona dotyczyła . Być może szczegóły tej sprawy powinny wreszcie ujrzeć światło dzienne , ze względu na liczne plotki narosłe wokół śmierci doktora Grimesbiego Roylotta , częstokroć mroczniejsze od samej prawdy . I tak oto na początku kwietnia roku osiemdziesiątego trzeciego zbudziwszy się wczesnym rankiem , zastał em kompletnie ubranego Holmesa stojącego nad moim łóżkiem . Zegar na kominku wskazywał kwadrans po siódmej , a Sherlock zwykł wstawać o późniejszej porze . Spojrzał em na niego z zaskoczeniem i odrobiną niechęci , ponieważ sam jestem stały w nawykach . Strasznie mi przykro , że zrywam cię tak wcześnie , Watsonie , ale tak się dziś złożyło . Wpierw pani Hudson została zerwana z łóżka , po czym odegrała się na mnie , a ja - na tobie .

Words senses

- miłość.1.n
  ##K: og. ##D: uczucie, objawiające się przeważnie przywiązaniem, oddaniem, prowadzące do silnej więzi emocjonalnej z kimś lub czymś. [##P: Imiona bohaterów: Ram i Lakhan nawiązują do wzorowej miłości braterskiej przedstawionej w indyjskim eposie Ramajana.] [##P: Pies spoglądał na swojego pana z nieskończoną miłością.] [##P: Miłość jest podstawową ideą chrześcijaństwa.] [##Ko: miłość do kraju, miłość do pieniędzy] ##A1: {radość; dobro, szczęście} + m [Miłość, którą darzyła syna i męża, wyrażała się w wielu gestach.] ##A2: {radość, zaufanie; dobro, szczęście} + m [Ich miłość przetrwała wszystkie próby i zestarzeli się razem tak, jak kiedyś sobie obiecali.] {wn:1671}
- miłość.3.n
  ##K: og. ##D: zamiłowanie do czegoś, głębokie zainteresowanie. [##P: Rodzice od małego pielęgnowali w niej miłość do muzyki.] ##A1: {radość; wiedza, szczęście} + m [Po paru miesiącach ćwiczeń jego miłość do tańca wygasła.] ##A2: {radość; wiedza, szczęście} + m [Miłość do botaniki rosła w niej z każdym rokiem studiów.]{wn:1672}
- miłość.4.n
  ##K: og. ##D: uczucie żywione do drugiej osoby, które zwykle wiąże się z pożądaniem. [##W: W młodości ożenił się z Anne de Bruil, do której poczuł prawdziwą, szczerą miłość.] ##A1: {radość, zaufanie; dobro, szczęście} + m [Po wielu latach przyjaźni między dwojgiem sąsiadów pojawiła się miłość.] ##A2: {radość, zaufanie; dobro, szczęście} + m [Miłość, którą cię darzę, droga Katarzyno, przetrwa wszystko.]{wn:102783}
- miłość.5.n
  ##K: książk. ##D: zbliżenie ludzi w celu prokreacyjnym lub dla osiągnięcia przyjemności. [##P: Uprawiali miłość przez całą noc.] {##L: http://pl.wikipedia.org/wiki/Stosunek_p%C5%82ciowy} ##A1: {radość, zaufanie; szczęście} + s [Marlena mawiała,

Figure 7: The extended perspective for word sense annotation

the auto annotation feature. Table 1 contains evaluation of the automatically recognized and added annotations for two languages and two subcorpora (each on a different topic). The auto annotation feature yielded very high precision of 97-99% with relatively high recall of 66-82%. This means that in case of Polish and Czech subcorpora 10k out of 14k annotations were added automatically. It was a significant facilitation of the work.

3. Assignment of annotation lemmas — to assign lemmas the annotators utilized batch lemma editor and lemma auto fill. For the correctly added annotations the lemmas were also automatically assigned.

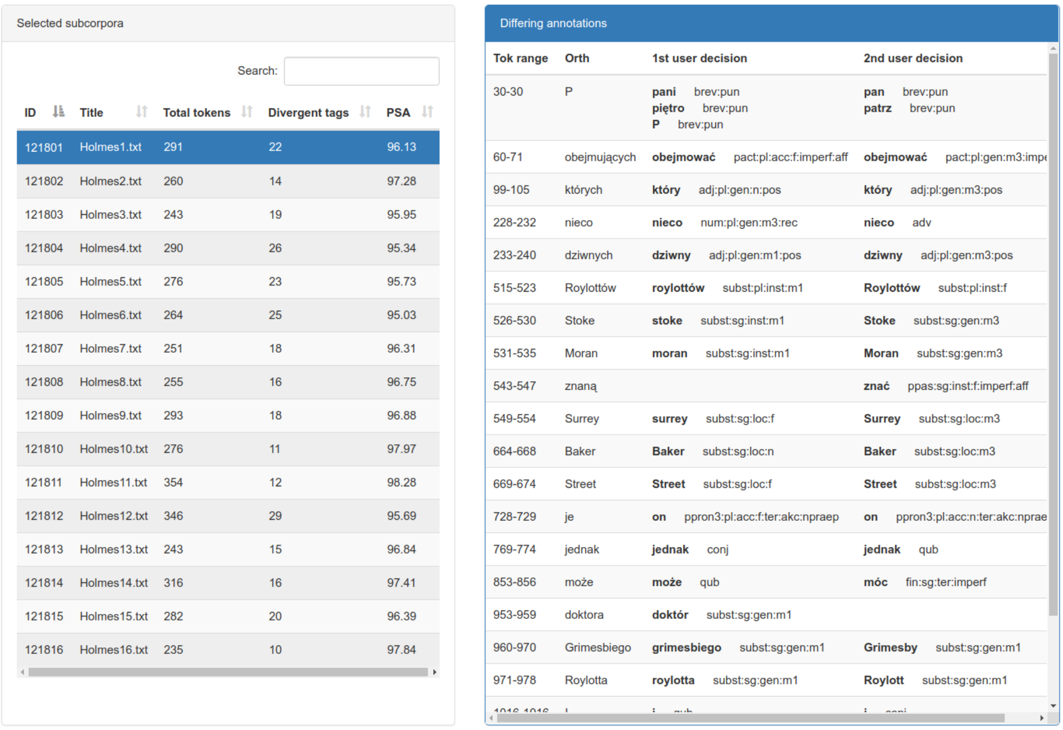
4. Assignment of cross-lingual identifier — the goal was to assign the same identifier for each mention across all languages referring to the same real-world entity. There were more than 4k identifiers. The annotators utilized the auto fill features described in Section 3.8. The attribute browser was used to validate the entity mentions.

4.2 Polish Translation of the NTU Multilingual Corpus

An ongoing project which goal is to provide Polish translation of the NTU Multilingual Corpus (Tan and Bond, 2011) which consists of two stories from the Sherlock Holmes Canon (The Adventure of the Speckled Band and The Adventure of the Dancing Men. The Adventure of the Speckled Band is the first one translated and prepared for manual annotation. The text was divided into 31 samples (txt files) of a similar size and imported directly into Inforex system. Then automatic morphological tagging was performed using WCRFT morpho-syntactic tagger for Polish (Radziszewski, 2013). The tagger provided morphological disambiguation on the basis of its context but also other possible forms for this particular word were listed. The result of the automatic annotation was then verified by two linguists independently. They were able to see morphological analysis of each token and the decision of the tagger (see Fig. 9). It could be accepted or discarded by the human annotator. It was also possible to add and assign an interpretation which was not identified by the tagger (e.g. in the case of unknown words). Inter-annotator agreement was calculated and its level was high enough (0,97) to perform further. Then, after completion the manual verification of morphological tagging by both linguists team coordinator proceeded with inconsistencies analysis. The decision was made for every token differently

Language	Polish		Czech	
Subcorpus	A	B	A	B
Total	5139	2440	4183	2504
Final	5032	2386	4128	2502
Discarded	107	53	55	2
Add by user	1015	648	696	829
Precision [%]	97.4	97.0	98.4	99.9
Recall [%]	79.9	72.8	83.1	66.9

Table 1: Evaluation of the *auto annotation* feature on the BSNLP 2019 Shared Task dataset



Selected subcorpora

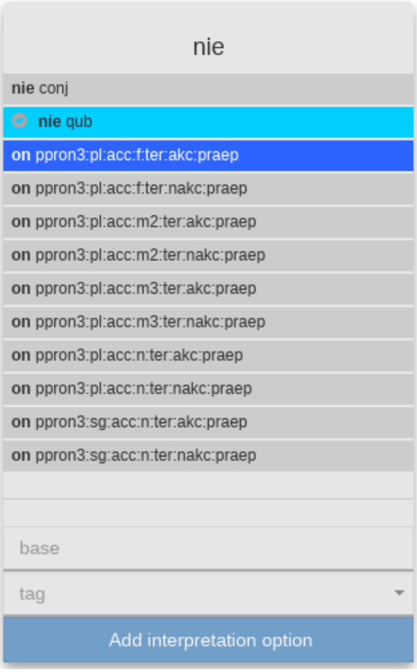
ID	Title	Total tokens	Divergent tags	PSA
121801	Holmes1.txt	291	22	96.13
121802	Holmes2.txt	260	14	97.28
121803	Holmes3.txt	243	19	95.95
121804	Holmes4.txt	290	26	95.34
121805	Holmes5.txt	276	23	95.73
121806	Holmes6.txt	264	25	95.03
121807	Holmes7.txt	251	18	96.31
121808	Holmes8.txt	255	16	96.75
121809	Holmes9.txt	293	18	96.88
121810	Holmes10.txt	276	11	97.97
121811	Holmes11.txt	354	12	98.28
121812	Holmes12.txt	346	29	95.69
121813	Holmes13.txt	243	15	96.84
121814	Holmes14.txt	316	16	97.41
121815	Holmes15.txt	282	20	96.39
121816	Holmes16.txt	235	10	97.84

Differing annotations

Tok range	Orth	1st user decision	2nd user decision
30-30	P	**pani** brev:pun **piętro** brev:pun **P** brev:pun	**pan** brev:pun **patrz** brev:pun
60-71	obejmujących	**obejmować** pact:pl:acc:f:imperf:aff	**obejmować** pact:pl:gen:m3:impe
99-105	których	**który** adj:pl:gen:n:pos	**który** adj:pl:gen:m3:pos
228-232	nieco	**nieco** num:pl:gen:m3:rec	**nieco** adv
233-240	dziwnych	**dziwny** adj:pl:gen:m1:pos	**dziwny** adj:pl:gen:m3:pos
515-523	Roylottów	**roylottów** subst:pl:inst:m1	**Roylottów** subst:pl:inst:f
526-530	Stoke	**stoke** subst:sg:inst:m1	**Stoke** subst:sg:gen:m3
531-535	Moran	**moran** subst:sg:inst:m1	**Moran** subst:sg:gen:m3
543-547	znaną		**znać** ppas:sg:inst:f:imperf:aff
549-554	Surrey	**surrey** subst:sg:loc:f	**Surrey** subst:sg:loc:m3
664-668	Baker	**Baker** subst:sg:loc:n	**Baker** subst:sg:loc:m3
669-674	Street	**Street** subst:sg:loc:f	**Street** subst:sg:loc:m3
728-729	je	**on** ppron3:pl:acc:f:ter:akc:npraep	**on** ppron3:pl:acc:n:ter:akc:nprae
769-774	jednak	**jednak** conj	**jednak** qub
853-856	może	**może** qub	**móc** fin:sg:ter:imperf
953-959	doktora	**doktór** subst:sg:gen:m1	
960-970	Grimesbiego	**grimesbiego** subst:sg:gen:m1	**Grimesby** subst:sg:gen:m1
971-978	Roylotta	**roylotta** subst:sg:gen:m1	**Roylott** subst:sg:gen:m1

Figure 8: Summary of morphological disambiguation agreement

annotated. All tags verified by the team leader obtained the status of final annotations. They were added to the version published within CLARIN-PL infrastructure (Błaszczak et al., 2019).



nie

nie conj

nie qub

on ppron3:pl:acc:f:ter:akc:praep

on ppron3:pl:acc:f:ter:nakc:praep

on ppron3:pl:acc:m2:ter:akc:praep

on ppron3:pl:acc:m2:ter:nakc:praep

on ppron3:pl:acc:m3:ter:akc:praep

on ppron3:pl:acc:m3:ter:nakc:praep

on ppron3:pl:acc:n:ter:akc:praep

on ppron3:pl:acc:n:ter:nakc:praep

on ppron3:sg:acc:n:ter:akc:praep

on ppron3:sg:acc:n:ter:nakc:praep

Figure 10: Morphological information provided for human annotators

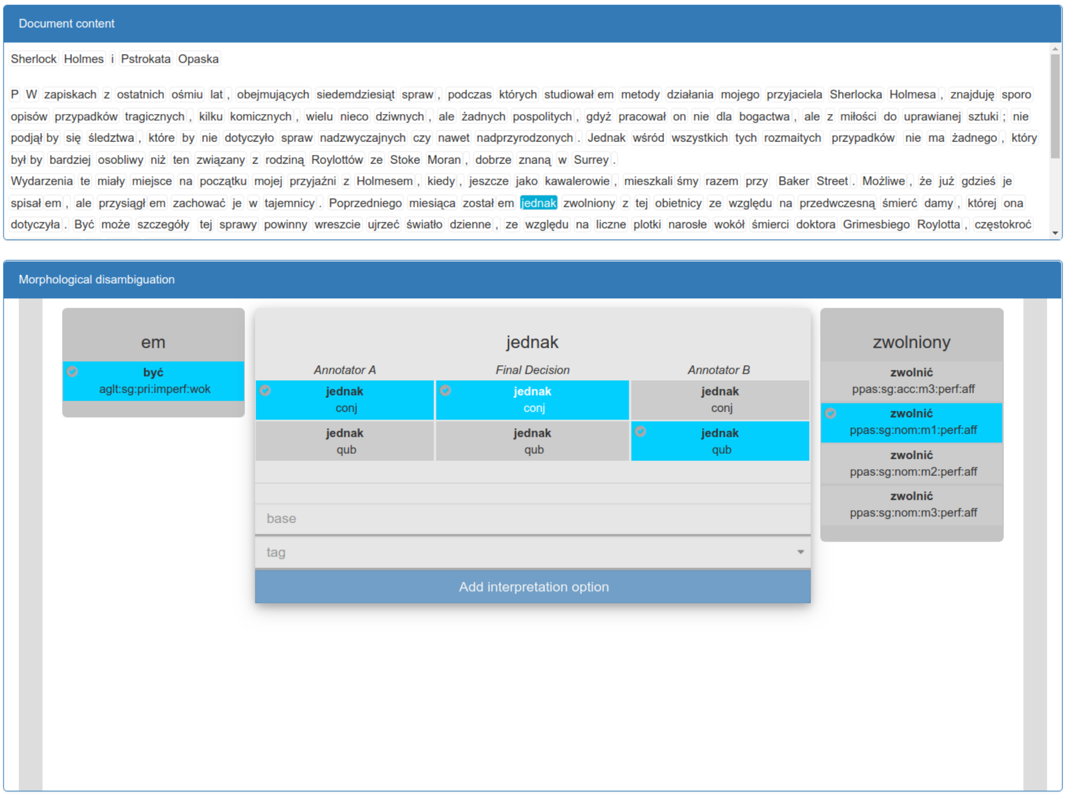
After the Inforex functionality was developed and primarily used for creation of The Adventure of the Speckled Band corpus, it was successfully applied to prepare Corpus of the colloquial Polish language (Oleksy, 2019) in another project. This corpus has been designed to address the problem of morphological tagging of user-generated content (UGC) as part of the project ”SentiCognitiveServices — next generation service for automating voice of customer and social media support based on artificial intelligence methods”[8]. The whole corpus (approximately 400000 tokens) is manually annotated with morphological information and furthermore the sample of 100 documents was prepared as a result of 2+1 annotation.

5 Summary

The last two years have been productive in the development of the Inforex system. Many new features and extensions were implemented during that time and the most important were presented in this paper. Majority of the features and improvements were dedicated by users. The most important news is the that Inforex has been finally released as an open source project.

Work financed as part of the investment in the CLARIN-PL research infrastructure funded by the

[8] https://sentione.com/knowledge/eu-research-project


Document content

Sherlock Holmes i Pstrokata Opaska

P W zapiskach z ostatnich ośmiu lat , obejmujących siedemdziesiąt spraw , podczas których studiował em metody działania mojego przyjaciela Sherlocka Holmesa , znajduję sporo opisów przypadków tragicznych , kilku komicznych , wielu nieco dziwnych , ale żadnych pospolitych , gdyż pracował on nie dla bogactwa , ale z miłości do uprawianej sztuki ; nie podjął by się śledztwa , które by nie dotyczyło spraw nadzwyczajnych czy nawet nadprzyrodzonych . Jednak wśród wszystkich tych rozmaitych przypadków nie ma żadnego , który był by bardziej osobliwy niż ten związany z rodziną Roylottów ze Stoke Moran , dobrze znaną w Surrey .

Wydarzenia te miały miejsce na początku mojej przyjaźni z Holmesem , kiedy , jeszcze jako kawalerowie , mieszkali śmy razem przy Baker Street . Możliwe , że już gdzieś je spisał em , ale przysiągł em zachować je w tajemnicy . Poprzedniego miesiąca został em jednak zwolniony z tej obietnicy ze względu na przedwczesną śmierć damy , której ona dotyczyła . Być może szczegóły tej sprawy powinny wreszcie ujrzeć światło dzienne , ze względu na liczne plotki narosłe wokół śmierci doktora Grimesbiego Roylotta , częstokroć

Morphological disambiguation

Figure 9: The perspective for morphological disambiguation agreement

Polish Ministry of Science and Higher Education.

References

Marta Błaszczak, Kacper Paszke, Ewa Rudnicka, Marcin Oleksy, Jan Wieczorek, Wioleta Kobylińska, Dominika Fikus, and Dagmara Kałkus. 2019. The adventure of the speckled band 1.0 (manually tagged). CLARIN-PL digital repository. http://hdl.handle.net/11321/667.

Bartosz Broda, Michał Marcińczuk, Marek Maziarz, Adam Radziszewski, and Adam Wardyński. 2012. KPWr: Towards a Free Corpus of Polish. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of LREC'12*. ELRA, Istanbul, Turkey.

Wei-Te Chen and Will Styler. 2013. Anafora: A web-based general purpose annotation tool. In *Proceedings of the 2013 NAACL HLT Demonstration Session*. Association for Computational Linguistics, pages 14–19. http://www.aclweb.org/anthology/N13-3004.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of the workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH) at COLING 2016*. pages 76–84. http://tubiblio.ulb.tu-darmstadt.de/97939/.

Michał Marcinczuk, Marcin Oleksy, and Jan Kocon. 2017. Inforex - a collaborative system for text corpora annotation and analysis. In Ruslan Mitkov and Galia Angelova, editors, *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, Varna, Bulgaria, September 2 - 8, 2017*. INCOMA Ltd., pages 473–482. https://doi.org/10.26615/978-954-452-049-6_063.

Michał Marcińczuk, Monika Zaśko-Zielińska, and Maciej Piasecki. 2011. Structure annotation in the polish corpus of suicide notes. In Ivan Habernal and Václav Matoušek, editors, *Text, Speech and Dialogue*, Springer Berlin Heidelberg, volume 6836 of *Lecture Notes in Computer Science*, pages 419–426.

Michał Marcińczuk, Jan Kocoń, and Bartosz Broda. 2012. Inforex – a web-based tool for text corpus management and semantic annotation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Marcin Oleksy. 2019. Corpus of the colloquial polish language. CLARIN-PL digital repository. http://hdl.handle.net/11321/637.

Maciej Piasecki, Stan Szpakowicz, Marek Maziarz, and Ewa Rudnicka. 2016. PlWordNet 3.0 – Almost There. In Verginica Barbu Mititelu, Corina Forăscu, Christiane Fellbaum, and Piek Vossen, editors, *Proceedings of the 8th Global Wordnet Conference, Bucharest, 27-30 January 2016*. Global Wordnet Association, pages 290–299.

Adam Radziszewski. 2013. A tiered crf tagger for polish. In *Intelligent Tools for Building a Scientific Information Platform*.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*. Association for Computational Linguistics, Avignon, France.

Liling Tan and Francis Bond. 2011. Building and annotating the linguistically diverse NTU-MC (NTU-multilingual corpus). In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation*. Institute of Digital Enhancement of Cognitive Processing, Waseda University, Singapore, pages 362–371. https://www.aclweb.org/anthology/Y11-1038.

Tomasz Walkowiak. 2018. Language processing modelling notation – orchestration of nlp microservices. In Wojciech Zamojski, Jacek Mazurkiewicz, Jarosław Sugier, Tomasz Walkowiak, and Janusz Kacprzyk, editors, *Advances in Dependability Engineering of Complex Systems*. Springer International Publishing, Cham, pages 464–473.

Semantic Language Model for Tunisian Dialect

Abir Masmoudi
MIRACL Laboratory
University of Sfax
masmoudiabir@gmail.com

Rim Laatar
MIRACL Laboratory
University of Sfax

Mariem Ellouze
MIRACL Laboratory
University of Sfax
Mariem.Ellouze@planet.tn

Lamia Belguith
MIRACL Laboratory
University of Sfax

Abstract

In this paper, we describe the process of creating a statistical Language Model (LM) for the Tunisian Dialect. Indeed, this work is part of the realization of Automatic Speech Recognition (ASR) system for the Tunisian Railway Transport Network. Since our field of work has been limited, there are several words with similar behaviors (semantic for example) but they do not have the same appearance probability; their class groupings will therefore be possible. For these reasons, we propose to build an n-class LM that is based mainly on the integration of purely semantic data. Indeed, each class represents an abstraction of similar labels. In order to improve the sequence labeling task, we proposed to use a discriminative algorithm based on the Conditional Random Field (CRF) model. To better judge our choice of creating an n-class word model, we compared the created model with the 3-gram type model on the same test corpus of evaluation. Additionally, to assess the impact of using the CRF model to perform the semantic labelling task in order to construct semantic classes, we compared the n-class created model with using the CRF in the semantic labelling task and the n-class model without using the CRF in the semantic labelling task. The drawn comparison of the predictive power of the n-class model obtained by applying the CRF model in the semantic labelling is that it is better than the other two models presenting the highest value of its perplexity.

1 Introduction

Generally, the development of such ASR system for a specific language requires first and foremost the construction of a large speech corpus. This corpus must be based on both an orthographic and a phonetic transcription. In addition, textual data for learning the LM of the system are also required. Nevertheless, these resources are not available directly for Arabic dialects. As a result, the ASR system development for Arabic dialects is fraught with many different kinds of difficulties that it faces. In this perspective, this work is integrated in the field of developing the Tunisian Dialect ASR system for the Tunisian Railway Transport Network. More precisely, we are interested in this paper in presenting our method for constructing a LM; one among the essential components of the ASR system. This model proposes to define the probability distribution on sets of word sequences. Due to the lack of learning data of the Tunisian dialect, it is necessary to find a method that maximizes the amount of information. This corresponds to the appearance of the n-class LM. The main idea of this model is to classify vocabulary words into lexical classes and to calculate the probability of a sequence of words, such as the probability of a sequence of lexical classes (5).

The primary contributions of this paper are as follows:

- Gathering our TARIC (Tunisian Arabic Railway Interaction Corpus) corpus to realize an ASR system for

Proceedings of Recent Advances in Natural Language Processing, pages 720–729,
Varna, Bulgaria, Sep 2–4 2019.

the Tunisian Railway Transport Network. This corpus is based on speech transcriptions. In order to obtain a standardized and normalized corpus, we employed our Tunisian Dialect CODA (Conventional Orthography for Dialectal Arabic) (30).

- We present our proposed method of creating the n-class LM. To achieve this, we show the different stages of our method.
- We evaluate the performance of the discriminative algorithm based on the CRF model in order to perform the semantic labeling task for spontaneous speech in the Tunisian Dialect in the n-class LM construction context.
- Testing the impact of using CRF model in the semantic labelling field. It is also important to assess its effect on creating semantic classes on the one hand, on developing an n-class LM and on its perplexity level on the other hand.
- We eventually disclosed the elaborated experiments we went through and the obtained results.

The remainder of this paper is organized as follows: Section 2 tries to show the main role of ASR system and its components. We also shed light on the main types LM. Section 3 discusses the related work in this field summarizing the main aspects of every work. Section 4 describes the TARIC corpus in Tunisian Dialect used for experiments. In section 5, we present the CRF discriminative model used to perform semantic labeling then, we explain our method to construct n-class LM for Tunisian Dialect. Then Conclusion is drawing in the last section.

2 ASR system and language model

An Automatic Speech Recognition (ASR) system aims to transcribe textual data of a speech signal. Indeed, in the context of statistical modeling of speech, an ASR system is composed of acoustic model, language model (LM) and phonetic dictionary. As our works aim at suggesting a method in order to construct a LM, we provided further explanations of this component by concentrating on its different types that have been proposed in the literature. As we mentioned earlier, the LM seeks to represent the language behavior in order to confirm or refute the propositions made by the acoustic module. In literature, several statistical LM types are recognized as being the most efficient models in ASR. Among these models, we can cite n-gram, sequences of n-words and factorial LM. We will present some examples of the aforementioned approaches in the following sub-sections.

2.1 N-gram model

Thanks to their simplicity and efficiency, n-gram models are the most widely used LM in the speech recognition field. They are based on the assumption that the appearance of a single word depends only on its history. In practice, estimating this probability is very difficult. In fact, no learning corpus can make it possible to observe all the sequences of possible words. As a result, the basic idea of n-gram models consists in considering only the sequences of words of length n, i. e the calculation is approached by a limited history consisting of the n-1 preceding words. The major drawback of this modeling type is that it can lead to assigning a zero probability to any n-gram that has never been encountered in the training corpus. This problem is serious especially when this n-gram could be perfectly valid in linguistics.

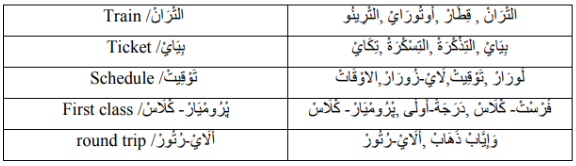
2.2 N-class model

Due to the lack of the learning corpus, it is vital to find a method that maximizes the amount of useful information on the one hand and reduces the model parameter space on the other hand. In order to meet this requirement, other methods have emerged recently. They consist in grouping words into classes. This corresponds to the appearance of n-class LM. The chief idea of this model is to group vocabulary words into lexical classes and to estimate the probability of the word sequence, such as the probability of a sequence of lexical classes (5). One of the clearest conception of motivations for n-class models is that a word of a given class, not necessarily found in the learning corpus, inherits the probability of all the other representatives of its class. In addition, it is possible to add words to classes without having to re-estimate the model probabilities. However, the problems faced by n-class models are numerous. The first major difficulty is that this type of model requires the need for a prelabeled learning corpus (24). Nevertheless, manual labeling is particularly heavy despite its exact results.

2.3 Factorial LM

The factored LM is based on the principle that a word is no longer seen as a simple graphic chain but rather as a vector of characteristics (3). These characteristics can include the lemma and the grammatical class of a word, morphs, its kinds, its numbers, or Booleans indicating the word belonging to given semantic classes. On the theoretical side, factorial models have already shown good results for some tasks, such as machine translation (14). At the practical level, few works have relied on this model, especially in the speech recognition task.

To conclude, due to the limitation of our field of work ”Tunisian Railway Transport Network”, several words with similar behaviors exist, (semantic for example) but they do not have the same probability of appearance; their class groupings will therefore be possible. Moreover, the amount of learning data is reduced. In this context, the use of the n-class model is beneficial at several levels. For these reasons, we propose to build an n-class LM that is based mainly on the integration of purely semantic data. Indeed, our method will be used to create a LM based on semantic information for the creation of word classes. The figure 1 shows some words with the same semantic behavior.



Train /التْرَانْ	التْرَانْ , قِطَارْ ,أوتُورَايْ ,التْرِينُو
Ticket /بِيَايْ	بِيَايْ ,التِذْكْرَةْ ,التِسكْرَةْ ,بِكَايْ
Schedule /تَوْقِيتْ	لُورَارْ ,تَوْقِيتْ,لَايْ-زُورَارْ,الاوْقَاتْ
First class /پْرُومْيَارْ - كْلَاسْ	فَرْسْتْ- كْلَاسْ ,دَرَجَةْ-أولَى ,پْرُومْيَارْ - كْلَاسْ
round trip /ألَايْ-رُتُورْ	وَإِيَابْ ذَهَابْ ,ألَايْ-رُتُورْ

Figure 1: some words with the same semantic behavior.

3 Related Work

In this Section, we are going to review the existing works related to classify vocabulary words for the construction of an n-type class LM. In the context of training classes of words, (10) proposes a simple word classification algorithm for statistical LM in speech recognition. The classification criterion used in this approach is the similarity of words. Indeed, the principle is based on the criterion of substitution or replacement. According to this algorithm, two words are similar since they can be substituted in the learning corpus (10). According to this automatic word classification approach, the word accuracy rate was increased by 8.6% with a reduction in perplexity of about 6.9% (10). The method proposed by (8) is essentially based on the principle of combining different sources of information at the class formation level. In his work, (8) uses two types of information: contextual information and prior information. The former is the most commonly used, corresponds to n-gram dependencies. This information can be collected not only at the words level, but also at the level of previously constructed classes of words (8). It is fundamental to take into account the contextual information in order to better distribute the words into the classes. Thus, the use of contextual information is of interest in the context of improving the predictability of the model. It makes it possible to offer a better distribution of words into classes and thus, a more balanced distribution of distributions (8). The second type, either semantic or syntactic information, is formalized by categories or grammars. In the approach proposed by (8), the used information a priori is extracted from a learning corpus labeled in grammatical categories. The approach proposed by (29) is based on contextual information (left context and right context), so words that appear frequently in similar contexts should be assigned to the same class. According to (29), different vocabulary words are classified using the k-means algorithm. The particularity of this approach is based on the fact that the number of words in a class is set to k and if there is a class whose number of words is less than k then that class will be merged with another. The main advantage of this algorithm is its simplicity to find centroids and suddenly, the cost of merging words or classes becomes less expensive. The approach developed by (2) proposes to integrate semantic information for the formation of word classes in the statistical LM of ASR system. This approach is based on a pivot language (called IF for Interchange Format), which represents the meaning of the sentence regardless of the language (2). Thus, the criterion of choice of classes is guided by the definition of the pivot language and the most used concepts in the IF.

4 Tunisian dialect corpus collection

We create our own corpus of real spoken dialogues corresponding to the information request task in railway stations in collaboration with the Tunisian National Railway Company (SNCFT). This corpus is called TARIC, for Tunisian Arabic Railway Interaction Corpus [16]. The main task of the TARIC corpus is information request in the Tunisian dialect about the railway services in a railway station. These requests are about consultation, train type, train schedule, train destination, train path, ticket price and ticket booking. The creation of the corpus was done based on two steps: the production of audio recordings and the manual transcription of these recordings.This corpus consists of 21,102 statements and 66,082 words.

The Tunisian dialect the Tunisians' mother tongue, which is used in their daily oral communication. It is becoming more and more useful not only in interviews, talk shows and public services but also in blogs, chat rooms, SMS, e-mails, etc. However until now the Tunisian dialect has no standardized spelling. Indeed, this dialect differs from MSA and it does not have a standard spelling because there are no academies of Arabic dialect. Thus, to obtain coherent learning data and to have a robust and powerful language model, it is necessary to utilize a standard spelling. Indeed, there are words with many forms. For example, the word رزرفسيون /reservation/ can be written in four different ways: رَازرفسيون, رَازَارفسيون and ريزرفسيون. As a result, each word has a unique form. Spelling transcription guidelines CODA (Tunisian Dialect writing convention), (30), were adopted.

The normalization step is essential because it presents a key point for the other steps of our method. Among the normalisation Tunisian Dialect words we have: *(i)*Number "sixteen" is written as ستطَاش. *(ii)*To define the future, we must follow the following form: بَاش + verb, for example: بَاش نمشي. *(iii)* To define the negation, we must follow the following form: مَا + verb.

The Tunisian Dialect is characterized by the presence of foreign words, such as for instance: French, English, Spanish, Italian, etc.To transcribe these words, Arabic alphabets have been used. These foreign words have a unique form, for example: رتور [Back], ترَان [train]... At the end of this step, we obtain a standardized corpus, the figure 2 represents a corpus extract before the normalization step.

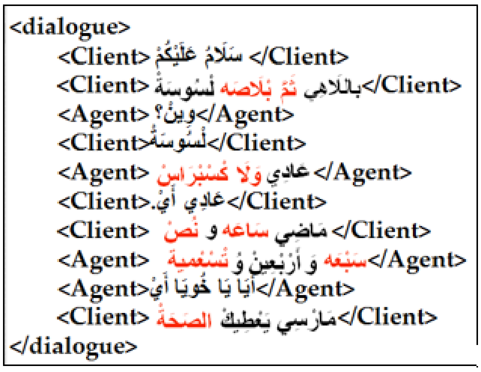

```
<dialogue>
    <Client> سَلَامْ عْلَيْكُمْ </Client>
    <Client> بِاللَّاهِي ثَمّ بْلَاصَه لْسُوسَةْ</Client>
    <Agent> وِينْ؟</Agent>
    <Client>لْسُوسَةْ</Client>
    <Agent> عَادِي وَلَا كْسْبْرَاسْ </Agent>
    <Client>عَادِي أَيْ.</Client>
    <Client> مَاضِي سَاعه و نُصْ </Client>
    <Agent> سَبْعَه وَ أَرْبْعِينْ وْ تْسْعْمِيَة</Agent>
    <Agent>أَيَا يَا خُويَا أَيْ</Agent>
    <Client> مَارْسِي يْعْطِيكْ الصَّحَّة</Client>
</dialogue>
```

Figure 2: Corpus extract before the normalisation step

The figure 3 represents a corpus extract after the normalization step.

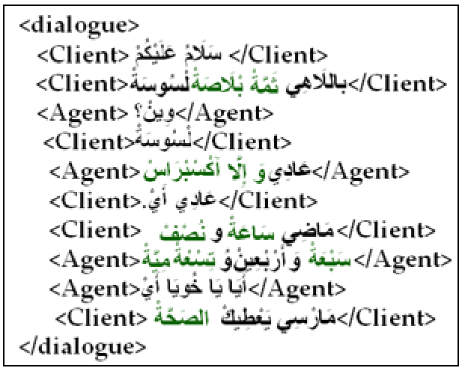

```
<dialogue>
  <Client> سَلَامْ عْلَيْكُمْ </Client>
  <Client>بِاللَّاهِي ثَمَّةْ بْلَاصَةْ لْسُوسَةْ</Client>
  <Agent> وِينْ؟</Agent>
   <Client>لْسُوسَةْ</Client>
   <Agent>عَادِي وَ إلَّا أكْسْبْرَاسْ</Agent>
   <Client>عَادِي أَيْ.</Client>
   <Client>  مَاضِي سَاعَةْ و نُصْفْ</Client>
   <Agent>سَبْعَةْ و أَرْبْعِينْ وْ تْسْعْمِيَّةْ</Agent>
   <Agent>أَيَا يَا خُويَا أَيْ</Agent>
    <Client> مَارْسِي يْعْطِيكْ الصَّحَّةْ</Client>
</dialogue>
```

Figure 3: Corpus extract after the normalisation step

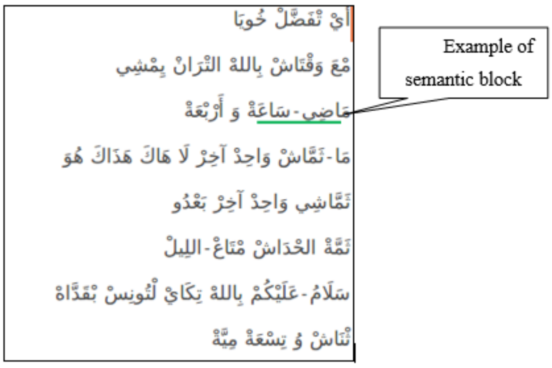
5 N-class LM construction

In this section, we are going to describe our method to create n-class LM based on semantic information for establishing word classes. This method is made up of five distinct stages.

5.1 Pre-processing stage

When we studied our corpus, we noticed that there are words that have no semantic value when they are figured all alone. Only the grouping of these words with other words can give a better semantic value to words that may be insignificant and subsequently useless for our field of work. As a result, we have decided to create semantic blocks that consist of grouping one or more words into a single word. Semantic block is defined as a group of two or more words. Indeed, this pretreatment consists of adding a (-) between two or more words to build a single word. Among the words that can be grouped together to form a semantic block, we find مَاضِي followed by another word to indicate the time for example مَاضِي سَاعَة [1 PM]. Cities whose names are composed such as سيدي بوزيد [name of Tunisian city] مَا [negation] followed by a verb with a negative form to express negation. This

step is necessary because it will be used for semantic labeling and later for the formation of word classes. Indeed, the main objective of this step is to give a better semantic value to words that may be insignificant and subsequently useless for our work. Following this step, we obtain a corpus that contains all the possible semantic blocks. Below (figure 4) is an excerpt from this corpus.

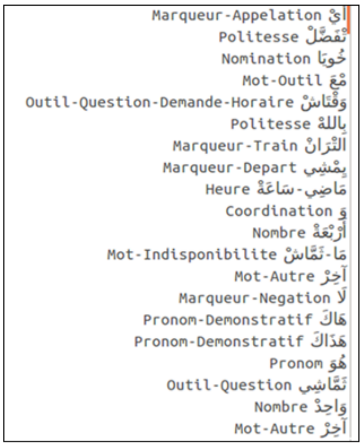

أَيْ تْفَضَّلْ خُويَا
مْعَ وَقْتَاشْ بِاللهْ التْرَانْ يِمْشِي
مَاضِي-سَاعَةْ وَ أَرْبْعَةْ
مَا-تْمَّاشْ وَاحِدْ آخِرْ لَا هَاكَ هَذَاكَ هُوَ
تْمَّاشِي وَاحِدْ آخِرْ بَعْدُو
ثَمَّةْ الحْدَاشْ مْتَاعْ-اللِيلْ
سَلَامْ-عَلَيْكُمْ بِاللهْ تِكَايْ لْتُونِسْ بْقَدَّاهْ
ثْنَاشْ وُ تِسْعَةْ مِيَّةْ

Example of semantic block

Figure 4: Extract of our corpus after semantic blocks formation

5.2 Statistical Semantic labeling

In this sub-section, we present in more details the way we integrate semantic order information for the formation of word classes. According to the previous steps, the semantic labeling step consists in giving a label to each single word or semantic block. We have performed the labeling task of words or semantic blocks extracted from a sentence and their corresponding concepts in the field of railway request information. In this task, each word or semantic block is labeled with a concept indicating its appropriate semantic nature. Thus, semantic labeling is not done word by word because we can find words that can have several meanings depending on the context in which they are used. Subsequently, the labeling of a word or a semantic block is done while taking into account its left and right neighborhood in a sentence. The figure 5 shows examples of semantic labeling.



Marqueur-Appelation أَيْ
Politesse تْفَضَّلْ
Nomination خُويَا
Mot-Outil مْعَ
Outil-Question-Demande-Horaire وَقْتَاشْ
Politesse بِاللهْ
Marqueur-Train اَلتْرَانْ
Marqueur-Depart يِمْشِي
Heure مَاضِي-سَاعَةْ
Coordination وَ
Nombre أَرْبْعَةْ
Mot-Indisponibilite مَا-تَمَّاشْ
Mot-Autre آخِرْ
Marqueur-Negation لَا
Pronom-Demonstratif هَاكَ
Pronom-Demonstratif هَدَاكَ
Pronom هُوَ
Outil-Question تَمَّاشِي
Nombre وَاحِدْ
Mot-Autre آخِرْ

Figure 5: Extract of semantically labelled corpus

In order to improve the sequence labeling task, we proposed to use a discriminative algorithm based on the Conditional Random Field (CRF) model. Thanks to their ability in learning the sequence tagging tasks efficiently, CRF have been applied to a wide range of NLP applications, such as morpho-syntactic tagging (POS), chunking and language modeling. CRF are probabilistic models for computing the conditional probability of a possible output giving an input sequence also called the observation sequence. To train semantic labeling associations and some predefined feature sets, CRF learns a set of weights w. Learning the parameter set w is usually done by a maximum likelihood learning for $p(\bar{x}|\bar{y};w)$:

$$p(\bar{x}|\bar{y};w) = \frac{1}{z(\bar{x}|w)} exp \sum_{j} w_j F_j(\bar{x},\bar{y}) \tag{1}$$

According to these equations, $\bar{x}$ represents the sequence of words or semantic blocks (observation), $\bar{y}$ represents the sequence of labels, and w stands for the weights. F_j corresponds to a feature function. The latter depends on the sequence of words, the current label, the previous label and the current position in an utterance. We utilized the CRF++ toolkit (16) in our experiments. It is a customizable and open source, which implement the CRF for segmenting and labeling sequential data. It is written in C++, employs fast training based on gradient descent and generates n-best candidates. In order to measure the performance of the labelling task, three evaluation metrics were generally adopted. The latter allow expressing Recall and Precision. These measures could be calculated as follows:

$$Precision = \frac{\#word\ correctly\ Labelling}{\#word\ Labelling} \tag{2}$$

$$Recall = \frac{\#word\ correctly\ Labelling}{\#total\ of\ word} \tag{3}$$

Our purpose is to create training and testing sets decently. We outlined the available datasets for the languages under investigation. We randomly selected 15826 sentences and 49562 words for training, 5276 sentences and 1652 words for testing. The outcomes of the CRF system show a Recall of 88% and a Precision of 87%. Based on the manual examination of the automatic labeling result using CRF, we found that the CRF have the ability to detect composed token specific to the task and label them correctly.

5.3 Construction of semantic classes

The present work, being mainly dedicated to building n-class-based LM, focuses essentially on the formation of semantic classes. Each class represents an abstraction of similar labels. In fact, a semantic class may correspond to a label or a group of labels, whereas a label cannot belong to only one class. Thus, if we consider the two statements: **Sentence 1:** تُونِس دُوزيَام كَلاس [Tunis second class]. **Sentence 2:** سوسَة پروميَار كَلاس [Sousa first class]. They become similar from this point of view because the words تُونِس [name of city] and سوسَة [name of city] belong to the same semantic class grouping ”city” and the words دُوزيَام and پروميَار belong to the same semantic class grouping ”Class”. Indeed, the number of classes must be limited while the number of occurrences of words belonging to same class must be large enough to come up with correct learned probabilities. We therefore choose to limit ourselves to the selection of classes most frequently encountered in our corpus, corresponding to our Field of study ”Tunisian Railway Transport Network”. In this step, we identify the most common label and group them into classes. A class then corresponds to a set of words leading to the same semantic labelling representation.

Table 1: Examples of Semantic Classes

Semantic classes	**Variants semantic tags**
City	Destination-Station..
Schedule	Trip-time, Arrival-hour ..
Response	Confirmation, Negation..

In our case, this classification provides about 20 semantic classes such as [City], [Response], [Request-Concept], [Comparison], [Schedule]. After obtaining the list of semantic classes, as shown in Table 1, we can then directly associate each word of our corpus with the class to which it belongs.

5.4 N-class calculation

We have already done the first experience for n-class LM calculation for Tunisian Dialect. After obtaining the list of semantic classes, we use it in combination with the data of LM to build our new ”semantic” model. In the LM learning corpus all words (or semantic blocks) are replaced by class names. The result is a ”prepared” corpus that contains both words (or semantic blocks) and Semantic classes. Finally, we use the SRILM [1] toolbox to learn LM including semantic classes. SRILM is a toolkit for building and applying statistical LM, primarily for use in speech recognition, statistical tagging and segmentation, and machine translation.

5.5 Evaluation of a LM

Several measures are used to evaluate the quality of LM. We present perplexity as the most used method. Perplexity (PPL) is a quick method to evaluate LM. It is commonly used for several years to judge the quality of LM (15). This evaluation metric is used to measure the prediction ability of LM on a test corpus not seen during learning. The principle of perplexity is to check how much LM can predict the word sequences of the language it is supposed to model. Perplexity is defined by:

$$PPL = 2^{-\frac{1}{n}} \sum_{t=1}^{n} logP(w_t|h) \qquad (4)$$

Where $P(w_t|h)$ represents the probability proposed by the LM for the word knowing the history h. Indeed, the perplexity of LM is between 1 and V, V is the size of vocabulary, that is to say the number of words that compose it. A reduced value of perplexity leads to better LM prediction capability.

As we have already mentioned, a low value of perplexity reflects a strong predictive power of LM. Thus, to better judge our choice of creating an n-class word model, we compared the created model with the 3-gram type model on the same test corpus of evaluation. Additionally, to assess the impact of using the CRF model to perform the semantic labelling task in order to construct semantic classes,

[1] http://www.speech.sri.com/projects/srilm/

we compared the n-class created model with using the CRF in the semantic labelling task and the n-class model without using the CRF in the semantic labelling task. The table below shows a comparison between the n-class model based on the CRF model in the Semantic labeling task and the n-class model without applying the CRF model in the Semantic labeling task together with the n-gram model in terms of perplexity.

Table 2: Value of perplexity calculated on the same test corpus

Type of model	Perplexity
3-gram	74.46
n-class without CRF	4.17
n-class with CRF	3.87

Table 2 shows the very significant relative reduction in perplexity by applying the CRF model in the semantic labelling compared to the other models. These results are consistent with what could be expected: (1) it is the classification based on semantic data that has minimized the perplexity of the obtained LM. The value of the n-class model perplexity remains well below that of the 3-gram model on the test corpus. Interestingly, the same models as for the learning corpus have the lowest perplexity value on the test corpus. (2) the application of CRF model to perform the semantic labelling task affects the improvement in creating semantic classes, by taking into account the n-class LM, and calculating the perplexity rates.

6 Conclusion

In this work, we have interested in constructing a statistical LM as one of the components of ASR system. The main role of this model is to give the probability of distribution on sets of word sequences. In particular, we are interested in n-class LM by using semantic information for the creation of word classes. Our choice is justified by the fact that some words are similar but they do not have the same probability of appearance, so their class groupings will be possible because of the limitation of our field of work ”the Tunisian Railway Transport Network”. The main idea of this model is to group vocabulary words into semantic classes and to consider mainly the calculation of the probability of a sequence of words such as the probability of a sequence of these semantic classes. To do this, we have followed these steps:We firstly construct semantic blocks that consists in grouping one or more words into a single word that we call ”semantic blocks”. Secondly, we do the semantic tagging. So in order to obtain a labeled corpus, the semantic labeling step consists in giving a label for each single word or for each semantic block. To improve the sequence labeling task, we proposed to use a discriminative algorithm based on the Conditional Random Field (CRF) model. Thirdly, we form semantic classes. In fact, a semantic class may correspond to a label or a group of labels, whereas a label cannot belong to only one class.LM calculation based on the SRILM tool. Evaluating the constructed model by calculating its perplexity. In order to test the model generated by our statistical LM system, we compared the created model with the 3-gram type model on the same test corpus. Secondly, to better judge the impact of using the CRF model to perform the semantic labelling task in order to construct semantic classes, we compared the n-class created model based on the CRF in the semantic labelling task and the n-class model without using the CRF in the semantic labelling task. According to this evaluation, we can say that the classification based on semantic data has minimized the perplexity of the LM obtained as compared to the rapport 3-gram LM. Moreover, the use of the CRF model to perform the semantic labelling task has an impact on the improvement in creating semantic classes, by taking into account the n-class LM and calculating the perplexity rates.

References

[1] Bahl L.R., Brown P.F., Souza P.V., Mercer R.L.: A tree-based Statistical Language Model for Natural Language Speech Recognition, IEEE Transactions on Acoustics, Speech and Signal Processing (1989).

[2] Bigi B : Modle de langage smantique pour la reconnaissance automatique de parole dans un contexte de traduction, Laboratoire Parole et Langage - Aix-en-Provence (2012).

[3] Bilmes J. A., Kirchhoff K.: Factored Language Models and Generalized Parallel Backo, Proc. of Human Language Technologies, North American (2003).

[4] Bouagres F. : Attelage de systmes de transcription automatique de la parole, thse de doctorat, universit du Maine-Le Mans-France (2012).

[5] Brown P. F., DellaPietra V. J., Souza P. V., Lai J. C., Mercer R. L.: Class-Based N-Gram Models of Natural Language, Computational Linguistics (1992).

[6] Chelba C., Engle D., Jelinek F., Jimenez V., Khudanpur S., Mangu L., Printz H., Ristad E., Rosenfeld R., Stolcke A., Wu D.: Structure and Performance of a Dependency Language Model, Dans Proc. of the European Conf. on Speech Communication and Technology (Eu-rospeech) (1997).

[7] Chelba C., Jelinek F.: Structured Language Modeling , Computer Speech and Language (2000).

[8] Damnati G. : Modles de langage et classification automatique pour la reconnaissance de la parole continue dans un contexte de dialogue oral homme-machine, thse de doctorat, uni-versit dAvignon et des pays du vaucluse (2000).

[9] Graja M., Jaoua M. and Belguith L.: Discriminative Framework for Spoken Tunisian Dialect Understanding. SLSP(2013)

[10] Farhat A., Isabelle J.F., O'Shaughnessy D.: Clustering Words for Statistical Language Mod-els based on Contextual Word Similarity, Dans Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, USA (1996).

[11] Jelinek F.: Continuous Speech Recognition by Statistical Methods, Proc. Of the IEEE (1976).

[12] Ji G., Bilmes J.: Dialog Act Tagging Using Graphical Models, Proc. Of the IEEE Intl Conf. on Acoustics, Speech, and Signal ProcessingICASSP (2005).

[13] Habash, N., Diab, M. and Rambow, O.: Conventional Orthography for dialectal Ara-bic,Proceedings of the Eighth International Conference on Language Resources and Evalua-tion, LREC-2012, (2012).

[14] Kirchhoff K., Yang M.: Improved Language Modeling for Statistical Machine Translation, Proc. of the ACL Workshop on Building and Using Parallel Texts (ParaTex), Morristown, NJ, USA. Association for Computational Linguistics (2005).

[15] Kneser R., Ney H.: Improved Clustering Techniques for Class-Based Statistical Language Modeling. Proc. European Conference on Speech Communication and Technology, Berlin, Allemagne (1993).

[16] Kudo, T.: crf++, http://chasen.org/ taku/software/CRF++/

[17] Lecorv G. : Adaptation thmatique non supervise d'un systme de reconnaissance automatique de la parole, thse de doctorat, Universit europenne de Bretagne (2010).

[18] Masmoudi A., Ellouze M., Estve Y., Hadrich Belguith L., Habash N.: A Corpus and Pho-netic Dictionary for Tunisian Arabic Speech Recognition, LREC'2014, Reykjavik, Iceland (2014).

[19] Masmoudi A Ellouze M., Estve Y., Bougares F HadrichBelguith L: Phonetic tool for the Tunisian Arabic, SLTU'2014, Russia, (2014).

[20] Masmoudi, A., Habash, N., Khmekhem, M., Esteve, Y. and Belguith, L.: Arabic Transliteration of Romanized Tunisian Dialect Text: A Preliminary Investigation, Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing 2015, Cairo, Egypt, p.608-619, (2015).

[21] Masmoudi, A., Bougares,F., Ellouze, M., Estve, Y., Belguith, L.: Automatic speech recognition system for Tunisian dialect. Language Resources and Evaluation 52(1): 249-267 (2018).

[22] Abir Masmoudi, Rim Laatar, Mariem Ellouze, Lamia Hadrich Belguith: N-Class Language Model for Tunisian Dialect Automatic Speech Recognition System. LPKM (2018)

[23] Rosenfeld R.: Adaptive Statistical Language Modeling: A Maximum Entropy Approach, PhD Thesis, (1994).

[24] Smali K., Conception et ralisation dune machine dicter entre vocale destine aux grands vocabulaires : Le systme MAUD, Thse de doctorat, Universit de Nancy 1, 1991.

[25] Smaili K., Jamoussi S., Langlois D., Haton J.-P.: Statistical feature language model, Proc. of the 8th Intl Conf. on Spoken Language ProcessingICSLP, (2004).

[26] Vergyri D., Kirchhoff K., Duh K., Stolcke A.: Morphology-Based Language Modeling for Arabic Speech Recognition, Proc. of the 8th Intl Conf. on Spoken Language Processing (ICSLP), Jeju Island, South Korea, (2004).

[27] Wang, Y., Acero, A., Mahajan, M., Lee, J.: Combining statistical and knowledge-based spoken language understanding in conditional models. In: Proceeding COLING/ACL, Sydney, Australia (2006)

[28] Xu P., Chelba C., Jelinek F.: A study on richer syntactic dependencies for structured lan-guage modeling, Proc. of the 40th Annual Meeting on Association for Computational Linguistics (ACL), Morristown, NJ, USA. Association for Computational Linguistics, (2002).

[29] Zitouni I.: Linearly Interpolated Hierarchical N-gram Language Models for Speech Recogni-tion Engines, IBM T.J. Watson Research Center, NY, Bell Labs Alcatel-Lucent, NJ,USA, (2008).

[30] Zribi I., Boujelben R., Masmoudi A., Ellouze M., Belguith L., Habash N.: A conventionnal Orthography for Tunisian Arabic, LREC'2014, Reykjavik, Iceland, (2014).

Automatic diacritization of Tunisian dialect text using Recurrent Neural Network

Abir Masmoudi
MIRACL Laboratory
University of Sfax
masmoudiabir@gmail.com

Mariem Ellouze Khmekhem
MIRACL Laboratory
University of Sfax
Mariem.Ellouze@planet.tn

Lamia Hadrich Belguith
MIRACL Laboratory
University of Sfax
l.belguith@fsegs.rnu.tn

Abstract

The absence of diacritical marks in the Arabic texts generally leads to morphological, syntactic and semantic ambiguities. This can be more blatant when one deals with under-resourced languages, such as the Tunisian dialect, which suffers from unavailability of basic tools and linguistic resources, like sufficient amount of corpora, multilingual dictionaries, morphological and syntactic analyzers. Thus, this language processing faces greater challenges due to the lack of these resources. The automatic diacritization of MSA text is one of the various complex problems that can be solved by deep neural networks today. Since the Tunisian dialect is an under-resourced language of MSA and as there are a lot of resemblance between both languages, we suggest to investigate a recurrent neural network (RNN) for this dialect diacritization problem. This model will be compared to our previous models models CRF and SMT (24) based on the same dialect corpus. We can experimentally show that our model can achieve better outcomes (DER of 10.72%), as compared to the two models CRF (DER of 20.25%) and SMT (DER of 33.15%).

1 Introduction

Modern Standard Arabic (MSA) as well as Arabic dialects are usually written without diacritics(24). It is easy for native readers to infer correct pronunciation from undiacritized words not only from the context but also from their grammatical and lexical knowledge. However, this is not the case for children, new learners and non-native speakers as they dont have a good mastery of rich language derivations. Moreover, the absence of diacritical marks leads to ambiguity that affects the performance of NLP tools and tasks. This may generally bring a considerable awkward ambiguity at the data-processing level for NLP applications. Hence, we can notice that automatic diacritization has been shown to be useful for a variety of NLP applications, such as Automatic Speech Recognition (ASR), Text-to-Speech (TTS), and Statistical Machine Translation (SMT)(24).

In this paper, we present our method to automatic dialectical Arabic diacritization. In fact, both previous experiences and works have shown that the use of Recurrent Neural Network (RNN) could give better results for such an MSA diacritization system as compared to the other approaches, like the Lexical Language Model (16), a hybrid approach combining statistical and rule-based techniques (28). For instance, the authors (1) demonstrated in their study that the RNN gave the least DER, compared to the other MSA diacritization works.

Based on the huge similarity between MSA and Tunisian dialect, we decided to benefit from its advantages by testing the RNN performance in the automatic diacritization of Tunisian Arabic dialects. To the best of our knowledge, it is the first work that investigates the RNN for the diacritization of the Tunisian dialect.

In this respect, we performed the task of restorating diacritical marks without taking into account any previous morphological or contextual analysis. Moreover, we diagnosed different aspects of the proposed model with various training options. The latter include the choice of transcription network (long short-term memory (LSTM) networks, bidirectional LSTM (B-LSTM)) and the impact of RNN sizes. The size of the neural network is a function of the number of hidden layers. Our goal is to choose the most pertinent layers

Proceedings of Recent Advances in Natural Language Processing, pages 730–739,
Varna, Bulgaria, Sep 2–4 2019.

in the Tunisian dialect based on the final findings provided by various experiments.

This model will be compared to our previous CRF and SMT models (24) by utilizing the same training and testing corpus.

The remaining of this paper is structured as follows: In Section 2 we describe the linguistic background of the Tunisian dialect, we try to show the complexity of diacritization tasks based on examples and we present the main level of diacritization. Section 3 introduces our proposed model and experiments. Section 4 provides an exhaustive experimental evaluation that illustrates the efficiency and accuracy of our proposed method. Section 5 summarizes the key findings of the present work and highlights the major directions for future research.

2 Linguistic background

2.1 Tunisian dialect

The language situation in Tunisia is "polyglossic", where distinct language varieties, such as the normative language (MSA) and the usual language (the Tunisian dialect) coexist (24).

As an official language, MSA is extensively present in multiple contexts, namely education, business, arts and literature, and social and legal written documents. However, the Tunisian dialect is the current mother tongue and the spoken language of all Tunisians from different origins and distinct social belongings. For this purpose, this dialect occupies a prominent linguistic importance in Tunisia.

Another salient feature of the Tunisian dialect is that it is strongly influenced by other foreign languages. In fact, it is the outcome of the interaction between Berber, Arabic and many other languages such as French, Italian, Turkish and Spanish. The manifestation of this interaction between these languages is obvious in introducing borrowed words. As a result, the lexical register of the Tunisian dialect seems to be more open and contains a very rich vocabulary.

The Tunisian dialect has other specific aspects. Indeed, since this dialects spoken rather than written or taught at school, there is neither grammatical, nor any orthographical or syntactic rules to be followed.

2.2 Challenges in the absence of diacritization in Tunisian dialect

The absence of diacritics signs in the Tunisian dialect texts often increases the morphological, syntactic and semantic ambiguity in the Tunisian dialect. Some of them are presented as follows:

- **Morphological ambiguity:** The absence of the diacritical marks poses an important problem at the association of grammatical information of the undiacritized word (24). For example, the word لعب /lEb/ admits several possible words that correspond to different solutions at the grammatical labeling level. We can find the plural noun "toys" and the verb "play" in 3rd person masculine, singular of passive accomplishment.

- **Syntactic ambiguity:** It should be noted that the ambiguities in the association of grammatical information, related to the undiacritic words, pose difficulties in terms of syntactic analysis (24). For example, the undiacritic expression كتب الولَيِّد عَالبيرو can admit two different diacritization forms that are syntactically accepted.
  - We find the diacritization form كتِب الولَيِّد عَالبيرو [The boy wrote on the desk] whose syntactic structure corresponds to a verbal sentence.
  - In addition, we find the diacritization form whose syntactic structure corresponds to a nominal sentence كتُب الولَيِّد عَالبيرو[The boy's books are on the desk].

- **Semantic ambiguity:**Tthe different diacritization of a word can have different meanings, even if they belong to the same grammatical category. For example, among the possible dicaritization of the word قريت /qryt/ we find the following dicaritization:
  - قرِيت /qryt/ [I read]
  - قَرِيت /qaryt/ [I taught].

  These two diacritic words have the same grammatical category: verb but they have two different meanings.

2.3 Diacritization level

The use of diacritic symbols in several instances is quite crucial in order to disambiguate homographic words. Indeed, the level of diacritization refers to the number of diacritical marks presented on a word to avoid text ambiguity for human readers. There are four levels of possible diacritization.

- **Full Diacritization:** this is the case where each consonant is followed by a diacritic. This type of diacritization is used mainly in classical Arabic, especially in religion-related books and educational writings.

- **Half Diacritization:** the objective of this category is to add diacritics of a word except the letters that depend on the syntactic analysis of the word. Often, it is the before last letter that depends on syntactic analysis of a word but it is not always the case due to the use of suffixes.

- **Partial Diacritization:** is the case of adding only lexical vowels. The latter can be defined as the vowels with which the meaning of words changes. The goal of marking these vowels is to remove ambiguity from the meaning of words.

- **No Diacritization:** This level is completely underspecified. The script is subject to ambiguity, especially with homographs (4).

3 Methodology and experiment step

In recent years, RNN has received a lot of interest in many NLP tasks of sequence transcription problems, such as speech recognition, handwriting recognition and diacritics restoration. So, we select the RNN to evaluate its performance on the diacritization of the Tunisian dialect. In this work, we adopted the full diacritization level, at which all diacritics should be specified in a word.

3.1 Recurrent neural networks

RNN can be mapped from a sequence of input observations to a sequence of output labels. The mapping is defined by activation weights and a non-linear activation function as in a standard MLP. However, recurrent connections allow to access past activations. For an input sequence x_1^T, RNN computes the hidden sequence h_1^T and the output sequence y_1^T by performing the following operations for $t = 1$ to T (13):

$$h_t = H(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

where H is the hidden layer activation function, W_{xh} is the weight matrix between input and the hidden layer, W_{hh} is the recurrent weight matrix between the hidden layer and itself, Why is the weight matrix between the hidden and output layers, b_h and b_y are the bias vectors of the hidden and output layers, respectively. In a standard RNN, H is usually an element-wise application of sigmoid function. Such a network is usually trained using the back-propagation through time (BPTT) training (27).

- **Long short-term memory: LSTM**

An alternative RNN called Long Short-Term Memory (LSTM) is introduced where the conventional neuron is replaced with a so-called memory cell controlled by input, output and forget gates in order to overcome the vanishing gradient problem of traditional RNNs (12). In this case, H can be described by the following composite function (13):

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t\ tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{6}$$

$$h_t = o_t\ tanh(c_t) \tag{7}$$

where the σ is the sigmoid function. *i*, *f*, *o* and *c* correspond to, the input, forget, output gates and cell activation vectors respectively.

- **Bidirectional Long short-term memory: B-LSTM**

A BLSTM processes the input sequence in both directions with two sub-layers in order to account for the full input context. These two sub-layers compute forward and backward hidden sequences $\overrightarrow{h}$, $\overleftarrow{h}$ respectively, which are then combined to compute the output sequence y as follows (13):

$$\overrightarrow{h}_t = H(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \quad (8)$$

$$\overleftarrow{h}_t = H(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \quad (9)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \quad (10)$$

3.2 Model architecture

In our diacritization task, the basic idea is to attribute a corresponding diacritical label to each character. Hence, we apply RNN to model our sequence data, where a sequence of characters constitutes the input and the probability distribution over diacritics forms the output. Schematically, our RNN structure is employed in this work as the following figure:

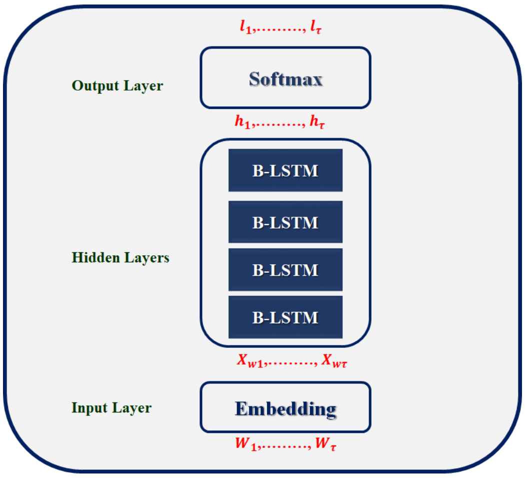


Figure 1: Architecture of our neural network

This can be statistically expressed in this way: Given that $W = (w_1...w_T)$, w indicates a sequence of characters, where each character is related to a label l_t. In this respect, a label may stand for 0, 1 or more diacritics. Furthermore, a real-valued vector x_w is a representation of each character w in the alphabet.

We can state that our neural network consists of 3 layers, namely an input layer, a hidden layer and an output layer. Each layer fulfils a particular purpose. In what follows, we will explain the advantages of each layer.

- **Input layer:** This level consists in mapping the letter sequence w to a vector sequence x. We have checked and prepared data of our corpus. In combining the gemination mark with another diacritic, each character in the corpus has a label corresponding to 0,1 or 2 diacritics. Character embedding input, which is initialized randomly and updated during training, means that each character in the input sentence is represented by a vector of *d* real numbers. It is worth pointing out that adding a linear projection after the input layer affects the learning of a new representation for the latter embedding.

- **Hidden layer:** This layer consists in mapping the vector sequence x to a hidden sequence h. Several types of hidden layers have been used to choose the best performance and the best result in the automatic diacritization of the Tunisian dialect. Hence, these experiments were based on LSTM different layers ranging from one layer to multiple B-LSTM layers.

- **Output layer:** This last layer focuses on mapping each hidden vector h_t to a probability distribution over labels l. In this layer, we use a softmax activation function to produce a probability distribution over output alphabet at each time step.

$$P(l|w_t) = \frac{exp(y_t[l])}{\sum_{l'} exp(y_t[l'])} \quad (11)$$

where $y_t = W_{hy}h_t + b_y$ and $y_t[l]$ is the l^{th} element of y_t.

3.3 Experience

As mentioned above, in order to train the RNN to achieve high accuracy, we apply our experiment based on several training options. These options include the choice of the number of layers in the hidden layer. The aim of this experiment is to determine which options will give optimal accuracy. Indeed, we applied these experiences, in which several types of hidden layers were tested. These layers are ranging from one LSTM layer to multiple B-LSTM layers.

The network is trained using Gradient Descent optimizer with learning rate 0.0003 and a mini-batch size of 200. For dropout, a rate of 0.2 is used both on embedded inputs and after each type of hidden layers; either LSTM or B-LSTM. Weights are randomly-initialized with normal distribution of zero mean and 0.1 standard deviation and weight updates after every batch. The loss function is the cross-entropy loss summed over all outputs. The GPU used is Nvidia GTX 580 that has 16 streaming multiprocessors and 1.5 GB of memory

4 Results and discussion

4.1 Evaluation Metric

In order to measure our model performance, an evaluation metric, known as Diacritic Error Rate (DER) is generally used. DER indicates how many letters have been incorrectly restored with their diacritics. The DER can be calculated as follows (24):

$$DER = \frac{(1 - |TS|)}{|TG|} * 100 \qquad (12)$$

Where $|TS|$ is the number of letters assigned correctly by the system, and $|TG|$ is the number of diacritized letters in gold standard texts.

4.2 Datasets

This section shows a breakdown of different sizes of our data sets, which were gathered from various sources. So far, we have used four existent types of corpora for our teamwork.

- We made use of our TARIC corpus (Tunisian Arabic Railway Interaction Corpus) (24). The latter collected information about the Tunisian dialect used in a railway station. This corpus was recorded in the ticket offices of the Tunis railway station. We recorded conversations in which there was a request for information about the train schedules, fares, bookings, etc. This corpus consists of several dialogues; each dialogue is a complete interaction between a clerk and a client. All the words are written using the Arabic alphabet with diacritics. The diacritics indicate how the word is pronounced. The same word can have more than one pronunciation.

- The second corpus is called STAC (Spoken Tunisian Arabic Corpus)(35). This corpus is a representation of spontaneous discourses in Tunisian dialect. This dialect corpus of transcribed discourses deals with multiple themes, such as social affairs, health, religion, etc.

- We utilized another type of corpus that is the result of a conversion tool from Latin written texts (also called Arabizi) into Arabic scripts following the CODA conversion. The Arabizi corpus is collected from social media like Facebook, Twitter and SMS messaging (22).

- In order to solve the problem of the lack of resources for the Tunisian dialect, we have chosen to gather corpora from blog sites written in this dialect using Arabic alphabets (24). (For more details see (24))

As mentioned above, the Tunisian dialect differs from MSA and it does not have a standard spelling because there are no academies of Arabic dialect. Thus, to obtain coherent learning data, it is necessary to utilize a standard spelling. Indeed, there are words with many forms. For example, the word رزرفسيون /reservation/ can be written in four different ways: رَازرفسيون, رَازَارفسيون and ريزرفسيون.

In this work, spelling transcription guidelines CODA (Tunisian Dialect writing convention), (36), were adopted. CODA is a conventionalized orthography for Dialectal Arabic. In CODA, every word has a single orthographic representation. It uses MSA-consistent and MSA-inspired orthographic decisions (rules, exceptions and ad hoc choices). CODA preserves, also, dialectal morphology and dialectal syntax. CODA is easily learnable and readable. CODA has been designed for the Egyptian Dialect (11) as well as the Tunisian Dialect (36) and the Palestinian Levantine Dialect (20). For a full presentation of CODA and an explanation of its choices, see ((11), (36)).

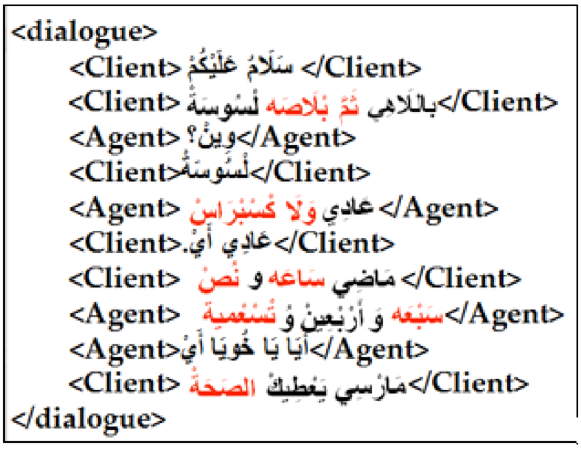
The normalization step is essential because it presents a key point for the other steps of our method. Among the normalisation Tunisian Dialect words we have:

- Number ”sixteen” is written as ستطَاش.

- To define the future, we must follow the following form: بَاش + verb, for example: بَاش نمشي.
- To define the negation, we must follow the following form: مَا + verb.

Let's remember that the Tunisian Dialect is characterized by the presence of foreign words, such as for instance: French, English, Spanish, Italian, etc. To transcribe these words, Arabic alphabets have been used. These foreign words have a unique form, for example: رتور [Back], ترَان [train]...

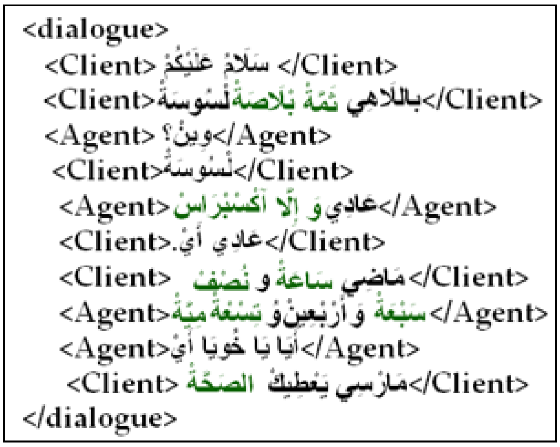
At the end of this step, we obtain a standardized corpus. The figure 2 represents a corpus extract before the normalization step.



```
<dialogue>
    <Client> سَلَامْ عَلَيْكُمْ </Client>
    <Client> بِاللَّاهِي ثَمّ بْلَاصَه لْسُوسَةْ</Client>
    <Agent> وِينْ؟</Agent>
    <Client>لْسُوسَةْ</Client>
    <Agent> عَادِي وَلَا كُسْنْبْرَاسْ </Agent>
    <Client>عَادِي أَيْ.</Client>
    <Client> مَاضِي سَاعَه و نُصّ </Client>
    <Agent> سَبْعَه وَ أَرْبْعِينْ وُ تُسْعْمِية</Agent>
    <Agent>أَيَا يَا خُويَا أَيْ</Agent>
    <Client> مَارْسِي يْعَطِيكْ الصَّحَّة</Client>
</dialogue>
```

Figure 2: Corpus extract before the normalisation step

The figure 3 represents a corpus extract after the normalization step.



```
<dialogue>
  <Client> سَلَامْ عَلَيْكُمْ </Client>
  <Client>بِاللَّاهِي ثَمَّةْ بْلَاصَةْ لْسُوسَةْ</Client>
  <Agent> وِينْ؟</Agent>
   <Client>لْسُوسَةْ</Client>
    <Agent>عَادِي وَ إِلَّا آكْسْنْبْرَاسْ</Agent>
    <Client>عَادِي أَيْ.</Client>
    <Client>  مَاضِي سَاعَةْ و نُصْفْ</Client>
    <Agent>سَبْعَةْ وَ أَرْبْعِينْ وْ تِسْعَةْ مِيَّةْ</Agent>
    <Agent>أَيَا يَا خُويَا أَيْ</Agent>
     <Client> مَارْسِي يْعَطِيكْ الصَّحَّةْ</Client>
</dialogue>
```

Figure 3: Corpus extract after the normalisation step

Since there are no automatic diacritization tools for the Tunisian dialect, and because the MSA tools are unable to treat this dialect due to the differences between the MSA and the Tunisian dialect, we were obliged to diacritize the corpus manually.

Below we provide the most important characteristics in Table 1.

Table 1: Characteristics of our corpus.

	# **statements**	#**words**
TARIC	21,102	71,684
STAC	4,862	42,388
Arabizi	3,461	31,250
Blogs	1582	27,544
Total	31,007	172,866

We aimed to decently create training, development and test sets in order to judge our diacritization models. We outlined the available datasets for the language under investigation. We randomly selected 23,255 sentences for training, 1,550 for development and 6,202 for testing.

Table 2 reports some quantitative information for the datasets.

Table 2: Tunisian dialect diacritization corpus statistics.

	Train	Dev	Test
# Statements	23,255	1,550	6,202
# words	129,649	8,643	34,574
# Letters	64,8247	43,216	172,867

4.3 Result

In this section, we present the evaluation outcome of our established diacritization models. We use DER as an evaluation metric. The adopted RNN has from 1 to 4 hidden layers, each with 250 neurons. This number is chosen after different experiments. We come up with the conclusion that a smaller number of neurons (less than 250) have an impact on accuracy rate and a greater number do not improve it in a significant way. Table 3 gives an overview of the RNN models outcomes in terms of diacritic error rate (DER).

Table 3: Diacritization Error Rate Summary for the Tunisian dialect RNN model

Model	DER
LSTM	13.86%
B-LSTM	12.31%
2-layer B-LSTM	11.53%
3-layer B-LSTM	10.72%
4-layer B-LSTM	10.83%

Table 3 shows the effect of using LSTM and B-LSTM models, and the number of hidden layers on the DER. According to this table, results show a DER of 13.56% for LSTM and 12.31% for B-LSTM. Based on the results of our RNN, we detected an enhancement of 1.55% in DER of the B-LSTM model as compared to LSTM model. This means that B-LSTM is more performant than LSTM.

Moreover, we noticed that increasing the number of B-LSTM layers from one hidden layer to three layers improves accuracy. But, we applied the 3-layer BLSTM because its accuracy is not only closer but also fasther than the 4-layer BLSTM. Indeed, the training time rises from 3:52 to 6:78 hours when the number of layers progresses monotonically from 3 to 4 and the testing time icreases from 3.65 to 5.41 minutes. Hence, the 3-layer B-LSTM configuration was adopted.

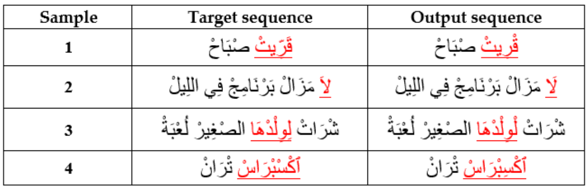
To conclude, a 3-layer BLSTM models achieved the best results.

4.4 Error Analysis

In order to reveal the weaknesses of our automatic diacritic restoration RNN models, we analyzed all errors that are mainly due to the following reasons:

- We noticed that these errors are due to the presence of foreign words in our corpus.
- Some error words with prefixes, or suffixes or both can be significantly perceived. It is hard to diacritize these complex words in a correct way, as the in flection diacritical mark is related to the last letter of the stem rather than to the last letter of the suffix.
- Errors due to form/spelling diacritization errors. Errors caused by ”Shadda” (consonant doubling), or Tanween (nunation). We perceived that restoring ”shadda” is harder than restoring the other diacritics.
- Errors due to missing and incorrect short vowels (i.e. lexical diacritics).

We have manually checked 150 error samples of our input RNN model. The following figure shows an example of 4 sample sequences that have errors. The words that have errors are underlined and in red.



Sample	Target sequence	Output sequence
1	قَرّيتْ صْبَاحْ	قْرِيتْ صْبَاحْ
2	لاْ مْزَالْ بَرْنَامِجْ فِي اللِيلْ	لَا مْزَالْ بَرْنَامِجْ فِي اللِيلْ
3	شْرَاتْ لِولْدْهَا الصْغِيرْ لُعْبَة	شْرَاتْ لْولْدْهَا الصْغِيرْ لُعْبَة
4	آكْسْبْرَاسْ تْرَانْ	آكْسِبْرَاسْ تْرَانْ

Figure 4: Sample sequences with errors

In about 21% of the samples, we have remarked that the absence of ”shadda” in some words can lead to a semantic ambiguity of the verb. For instance, sample 1 shows that target verb قَرِّيت [I taught] is output as قْرِيت [I read]. These two diacritic words have the same grammatical category: verb but they have two different meanings.

Diacritization errors in test samples can cause about 4% of errors. For example, sample 2 displays that the "Fatha" in the word لاَ was mistakenly entered after the last letter rather than the first letter.

Some error words with prefixes, suffixes or both can be significantly perceived, in about 41% of the samples. In another illustration, the error word in sample 3 has both the prefix "la" لِ and the pronoun suffix "haA" هَا.

We also noticed a significant fraction of error words (34%) due to the presence of foreign words in our corpus as in sample 4..

4.5 Comparison with State-of-art Systems

In this section, we compare our proposed RNN model with two other models, namely SMT and a discriminative model as a sequence classification task based on CRFs (24). These two models were realized in our previous works in order to carry on the dialect restoration for the Tunisian dialect. To achieve this comparison, we employed the same dialectical corpus and evaluation metrics.

Concerning the first model, we regarded the diacritization problem as a simplified phrase-based SMT task. The source language is the undiacritic text while the target language is the diacritic text. The basic idea of SMT is to analyze automatically existing human sentences called parallel corpus in order to build translation model. The alignment from words without diacritics to words with diacritics is a monotonic mapping. To do this, we employed moses (21) a SMT tool.Word alignment was done with GIZA++ (25). We implemented a 5-gram language model using the SRILM toolkit (31). We decoded using Moses (21).

In the second model, we decided to get the diacritical marks restoration by focusing on diacritization based on grammatical information. We intended to build dependency relations between words and "POS" tags and to perceive their effects on word diacritizations. In fact, we proposed to scrutinize the integration of grammatical information "POS" for the diacritization with the aid of Conditional Random Fields (CRF)(24).

The following table reviews the accuracy results to restore diacritics automatically from our previous published researches in the Tunisian dialect field.

Table 4: Diacritization results of related work (CRF and SMT models) and our RNN model

Model	**DER**
3-layer B-LSTM	10.72%
CRF	20.25%
SMT	33.15%

As depicted in Table 4, our RNN model (3-layer B-LSTM) provides the best results(DER of 10.72%) compared to both SMT (DER of 33.15%) and CRF (DER of 20.25%) models.

5 Conclusion

The absence of short vowels gives rise to a great ambiguity which influences the results of such NLP applications. An outcome of this study was the development of RNN diacritic restoration model for Tunisian dialect. To the best of our knowledge, this is the first work that deals with the problem of Tunisian dialect diacritizers using RNN.

In order to choose the best configuration of the RNN network, we did several preliminary experiments with different training options. These options concern the hidden layer where we tested the impact of the change of the neural network size and the topology on its performance. Several types of hidden layers are tested, ranging from one layer LSTM to multiple B-LSTM layers. The best accuracy is obtained when using the 3-layer B-LSTM model (DER of 10.72%). We compared our RNN diacritization model with two major models, namely a SMT and CRF models (24). These two models were realized in our previous works in order to carry on the dialect restoration for the Tunisian dialect. During this comparison, we employed the same dialectical corpus and evaluation metrics. About 9.53% DER improvement of RNN model was achieved over the best reported CRF model.

We have two future plans for the diacritization problems of Tunisian dialect. The first plan consists in expanding a rule-based diacritizer system and integrating it into our RNN model in order to ameliorate the outcomes. The second plan focuses on providing morphological analysis of such words to the RNN in order to achieve higher accuracy. The presence of significant fraction of errors in complex words that contain prefixes, suffixes, or both open up new perspectives for future research.

Abandah, G., Graves, A., Al-Shagoor, B., Arabiyat, A., Jamour, F. and Al-Taee. M. 2015. Automatic diacritization of arabic text using recurrent neural networks. International Journal on Document Analysis and Recognition (IJDAR).

Alotaibi, Y. A., Meftah, A. H. and Selouani. S.A. 2013. Diacritization, Automatic Segmentation and Labeling for Levantine Arabic Speech. In Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE).

Alqudah,S., Abandah,G., Arabiyat, A., 2017.Investigating Hybrid Approaches for Arabic Text Diacritization with Recurrent Neural Networks. 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies.

Al-Badrashiny, M., Hawwari, A. and Diab, M. 2017. A Layered Language Model based Hybrid Approach to Automatic Full Diacritization of Arabic. Third Arabic Natural Language Processing Workshop.

Ameur, M. Moulahoum, Y. and Guessoum, A. 2015. Restoration of Arabic Diacritics Using a Multilevel Statistical Model. In IFIP International Federation for Information Processing.

Ayman, A. Z., Elmahdy, M., Husni, H. and Al Jaam, J. 2016. Automatic diacritics restoration for Arabic text. International Journal of Computing & Information Science..

Azmi, A. and Almajed, R. 2015. A survey of automatic Arabic diacritization techniques. Natural Language Engineering, 21, pages:477495.

Belinkov, Y. and Glass. J. 2015. Arabic diacritization with recurrent neural networks. Conference on Empirical Methods in Natural Language Processing.

Bouamor, H., Zaghouani, W., Diab, M., Obeid, O., Kemal, O., Ghoneim, M. and Hawwari, A. 2015. A pilot study on Arabic multi-genre corpus diacritization annotation. The Second Workshop on Arabic Natural Language Processing.

Diab, M., Ghoneim, M. and Habash. N. 2007. Arabic Diacritization in the Context of Statistical Machine Translation. In Proceedings of MTSummit, Denmark.

Diab, M., Habash, N., Owen, R.: Conventional Orthography for Dialectal Arabic. In Proceedings of the Language Resources and Evaluation Conference , Istanbul,2012

Gers,F. : Long short-term memory in recurrent neural networks, Ph.D. dissertation, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Switzerland, 2001.

Graves, A., Mohamed, A., Hinton, G: Speech recognition with deep recurrent neural networks, IEEE International Conference on Acoustics, Speech, and Signal Processing, Canada,2013.

Fashwan, A., Alansary, S. 2017. SHAKKIL: an automatic diacritization system for modern standard Arabic texts. The Third Arabic Natural Language Processing Workshop (WANLP).

Habash, N., Shahrour, A., and Al-Khalil, M. 2016, Exploiting Arabic Diacritization for High Quality Automatic Annotation, the Tenth International Conference on Language Resources and Evaluation, LREC 2016.

Habash, N. and Rambow, O. 2007. Arabic diacritization through full morphological tagging. The Conference of the North American Chapter of the Association for Computational Linguistics.

Hamed, O and Zesch, T. 2017. A Survey and Comparative Study of Arabic Diacritization Tools. Journal of Language Technology and Computational Linguistics, volume 32, number 1.

Harrat, S., Abbas, M., Meftouh, K., Smaili, K., Bouzareah, E.N.S. and Loria, C. 2013. Diacritics restoration for Arabic dialect texts. 14th Annual Conference of the International Speech Communication.

Hifny, Y. 2012. Higher order n-gram language models for Arabic diacritics restoration. In Proceedings of the 12th Conference on Language Engineering.

Jarrar, M., Habash, N., Akra, D. and N. Zalmout, N.: Building a Corpus for Palestinian Arabic: a Preliminary Study :In Proceedings of the Arabic Natural Language Processing Workshop, EMNLP, Doha,2014

Koehn, P. Hoang; H. Birch, A. Callison-Burch, C. Federico, M. and Bertoldi, N. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. ACL 2007, demonstration session.

Masmoudi, A., Habash, N., Khmekhem, M., Esteve, Y. and Belguith, L.: Arabic Transliteration of Romanized Tunisian Dialect Text: A Preliminary

Investigation, Computational Linguistics and Intelligent Text Processing - 16th International Conference, CICLing 2015, Cairo, Egypt, p.608-619, (2015).

Masmoudi, A., Bougares,F., Ellouze, M., Esteve, Y., Belguith, L.: Automatic speech recognition system for Tunisian dialect. Language Resources and Evaluation 52(1): 249-267 (2018).

Masmoudi, A., Mdhaffer,S., Sellami, R. Belguith, L.: Automatic Diacritics Restoration for Tunisian Dialect. TALLIP2018: Transactions on Asian and Low-Resource Language Information Processing.

Och, F. and Ney, H. 2003. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, 29(1):19–52.

Rashwan, M., Al Sallab, A., Raafat, H. and Rafea, A. Deep Learning Framework with Confused Sub Set Resolution Architecture for Automatic Arabic Diacritization. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015.

Rumelhart, D., Hinton,G., and Williams, R. Learning representations by back-propagating errors, Nature, no. 323, pp. 533 536, 1986.

Said, A., El-Sharqwi, M., Chalabi, A , and Kamal, E. A hybrid approach for Arabic diacritization, Application of Natural Language to Information Systems, pp. 53-64, Jun 2013.

Shaalan, K., Abo Bakr, M. and Ziedan. I. 2009. A hybrid approach for building Arabic diacritizer. In Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages.

Shaalan, K., Abo Bakr, H. and Ziedan. I. 2008. A statistical method for adding case ending diacritics for Arabic text. In Proceedings of Language Engineering Conference.

Stolcke A. 2002. SRILM an Extensible Language Modeling Toolkit. Proceedings of ICSLP.

Zaghouani, W., Bouamor, H., Hawwari, A., Diab, M., Obeid, O., Ghoneim, M., Alqahtani, S and Oflazer, K. 2016. Guidelines and framework for a large-scale Arabic diacritized corpus. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016).

Zitouni, I., Sorensen, J. and Sarikaya, R. 2006. Maximum entropy based restoration of Arabic diacritics. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics.

Zitouni, I. and Sarikaya, R. 2009. Arabic Diacritic Restoration Approach Based on Maximum Entropy Models. In Journal of Computer Speech and Language.

Zribi, I., Ellouze, M., Belguith, L.H. and Blache, P. 2015. Spoken Tunisian Arabic Corpus ”STAC”: Transcription and Annotation. Res. Comput. Sci. 90.

Zribi I., Boujelben R., Masmoudi A., Ellouze M., Belguith L., Habash N.: A conventionnal Orthography for Tunisian Arabic, LREC’2014, Reykjavik, Iceland, (2014).

Comparing MT Approaches for Text Normalization

Claudia Matos Veliz, Orphée De Clercq and Véronique Hoste
LT^3, Language and Translation Technology Team - Ghent Univeristy
Groot-Brittanniëlaan 45, 9000, Ghent, Belgium
Firstname.Lastname@UGent.be

Abstract

One of the main characteristics of social media data is the use of non-standard language. Since NLP tools have been trained on traditional text material, their performance drops when applied to social media data. One way to overcome this is to first perform text normalization. In this work, we apply text normalization to noisy English and Dutch text coming from different genres: text messages, message board posts and tweets. We consider the normalization task as a Machine Translation problem and test the two leading paradigms: statistical and neural machine translation. For SMT we explore the added value of varying background corpora for training the language model. For NMT we have a look at data augmentation since the parallel datasets we are working with are limited in size. Our results reveal that when relying on SMT to perform the normalization, it is beneficial to use a background corpus that is close to the genre to be normalized. Regarding NMT, we find that the translations - or normalizations - coming out of this model are far from perfect and that for a low-resource language like Dutch adding additional training data works better than artificially augmenting the data.

1 Introduction

Probably one of the most persistent characteristics of social media texts is that they are full of non-standard words (Eisenstein, 2013). Several sources of noise can influence the way people write. For example, the different kind of social media platforms available nowadays provide a diverse range of ways to communicate and particular forms of language variations (Ke et al., 2008). Social variables such as gender, age and race can also influence communication style (Schwartz et al., 2013; Blodgett et al., 2016). Location is also an important variable, since it can lead to the use of dialect and non-standard words. (Vandekerckhove and Nobels, 2010; Blodgett et al., 2016).

Very typical for this User-Generated Content (UGC) is the expression of emotions by the use of symbols or lexical variations of the words (Van Hee et al., 2017). This can be done in the form of flooding or the repetition of characters, capitalization, and the productive use of emoticons. In addition, the use of homophonous graphemic variants of a word, abbreviations, spelling mistakes or letter transpositions are very typical, since people tend to write as they speak and/or write as fast as possible.

One can imagine that all those characteristics contribute to an increased difficulty of automatically processing and analyzing UGC. Since Natural Language Processing (NLP) tools have originally been developed for and trained on standard language, these non-standard forms adversely affect language analysis using these tools. One of the computational approaches which has been suggested to tackle this problem is text normalization (Sproat et al., 2001). This approach envisages transforming the lexical variants to their canonical forms. In this way, standard NLP tools can be applied in a next step, after normalization. Kobus et al. (2008) introduced three metaphors to refer to these normalization approaches: the spell checking, automatic speech recognition and machine translation metaphors. In section 2 these approaches are discussed in more depth.

In this work, we follow the third metaphor and tackle text normalization as a Machine Translation (MT) task. We test the two leading paradigms,

Proceedings of Recent Advances in Natural Language Processing, pages 740–749,
Varna, Bulgaria, Sep 2–4 2019.

statistical machine translation (SMT) and neural machine translation (NMT), on English and Dutch parallel corpora with data coming from three genres (text messages, message board posts and tweets). For SMT we explore the added value of varying background corpora for training the language model. For NMT we have a look at data augmentation since the parallel datasets we are working with are limited in size. Our results reveal that when relying on SMT to perform the normalization it is beneficial to use a background corpus that is close to the genre to be normalized. Regarding NMT, we find that the translations - or normalizations - coming out of this model are far from perfect and that for a low-resource language, like Dutch, adding additional training data works better than artificially augmenting the data.

In the following section, we discuss related work on text normalization. In section 3, we give more information about our parallel data and describe the methodology we used to perform the SMT and NMT experiments. Section 4 gives an overview of the results, whereas section 5 concludes this work and offers some prospects for future work.

2 Related Works

Previous research on UGC text normalization has been performed on diverse languages using different techniques ranging from hand-crafted rules (Chua et al., 2018) to deep learning approaches (Ikeda et al., 2016; Sproat and Jaitly, 2016; Lusetti et al., 2018). Kobus et al. (2008) introduced three metaphors to refer to these normalization approaches: the spell checking, automatic speech recognition and translation metaphors.

In the **spell checking metaphor**, corrections from noisy to standard words occur at the word level. As in conventional spelling correction one has to deal with both non-word and real-word errors (Clark and Araki, 2011). The disadvantage of this approach is that all non-standard words (NSWs) have to be represented in the dictionary in order to obtain the corresponding normalization. Therefore, the success of this kind of systems highly depends on the dictionary coverage. However, as UGC is a very generative language and new variants of canonical words and phrases appear constantly, it is very difficult and expensive to maintain a high coverage lexicon. Other works have approached the problem using a noisy channel model. In this model, the goal is to find the intended word w given a word x where the letters have been changed in some way. Correct words in the text remain untouched. This model is probably the most popular and successful approach to spelling correction (Dutta et al., 2015; Goot, 2015). Although spelling correction is mostly performed on languages which are morphologically simple and with a fairly strict word order, like English, there has been some progress for normalization applied to other languages as well, such as Russian (Sorokin, 2017) and French (Beaufort and Roekhaut, 2010).

Text found in social media shares features with spoken language and the **automatic speech recognition metaphor** exploits this similarity. This approach starts by converting the input message into a phone lattice, which is converted to a word lattice using a phoneme-grapheme dictionary. Finally, the word lattice is decoded by applying a language model to it and using a best-path algorithm to recover the most likely original word sequence. This metaphor has mainly been merged with the machine translation (infra) and spell checking (supra) metaphors to improve the quality of the normalization. Kobus et al. (2008), for example, incorporated ideas from speech recognition to text message normalization and combined it with a machine translation system. Beaufort and Roekhaut (2010); Xue et al. (2011) and Han and Baldwin (2011) also combined the automatic speech recognition approach with spell checking and machine translation techniques.

The **machine translation metaphor** treats social media text as the source language and its normalized form as the target language. Several works have tackled the problem of text normalization using this approach. Statistical Machine Translation (SMT) models, especially those trained at the character-level, have proven highly effective for the task because they capture well intra-word transformations. One of the first works following this approach was presented by Aw et al. (2006). They adapted phrase-based SMT to the task of normalizing English SMS producing messages that collated well with manually normalized ones. Besides, they studied the impact of the normalization on the task of SMS translation, showing that SMS normalization, as a preprocessing step of MT, can boost the translation performance. Kaufmann (2010) used a two-step approach for

Source sentence	Target sentence	Translation
iz da muzieksgool vnavnd ? kwt da niemr .	is dat muziekschool vanavond ? ik weet dat niet meer .	is that music school tonight? I don't know that anymore.
wa is je msn k en e nieuwe msn omda k er nie meer op graal . xxx	wat is je msn ik heb een nieuwe msn omdat ik er niet meer op geraak . xx	what is your msn i have a new msn because i can't get it anymore. xx
@renskedemaessc dm me je gsmnummer eens ;-)	<user> doormail me je gsm-nummer eens <emoji>	<user> mail me your cellphone number once <emoji>

Table 1: Source and target pairs as parallel data for a machine translation approach.

the normalization of English tweets: he first preprocessed the tweets to remove as much noise as possible and then used a machine translation approach to convert them into standard English. MT approaches when used at the character level, also have the advantage of being effective when small training data is provided, thanks to their small vocabulary size. De Clercq et al. (2013) proposed a phrase-based method to normalize Dutch UGC comprising various genres. They performed experiments at several levels of granularity: character and word level. In a preprocessing step they handled emoticons, hyperlinks, hashtags and so forth. Then they worked in two steps: first at the word level and then at the character level. This approach revealed good results across various genres of UGC; however a high number of phonetic alternations still remained unresolved. Schulz et al. (2016) made a modification to the previous work by combining the three metaphors (machine translation, spell checking and speech recognition) in a multi-modular system and by using a novel approach for decoding. This led to an improvement in the selection of the best normalization option. Furthermore, they showed a performance improvement of state-of-the-art NLP tools on UGC data when normalization is used as a previous step.

Recently, neural networks have proven to outperform many state-of-the-art systems in several NLP tasks (Young et al., 2018). The encoder-decoder model for recurrent neural networks (RNN) was developed in order to address the sequence-to-sequence nature of machine translation and it obtains good results for this task (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015). The model consists of two neural networks: an encoder and a decoder. The encoder extracts a fixed-length representation from a variable-length input sentence, and the decoder generates a correct translation from this representation. Some works on text normalization have followed the same approach. Ikeda et al. (2016) performed text normalization at the character level for Japanese text and proposed a method for data augmentation using hand-crafted rules. They proved that the use of the synthesised corpus improved the performance of Japanese text normalization. Mandal and Nanmaran (2018) presented an architecture for automatic normalization of phonetically transliterated words to their standard forms in a code-mixed scenario improving the accuracy of a pre-existing sentiment analysis system by 1.5%. Lusetti et al. (2018) performed text normalization over Swiss German WhatsApp messages and compared it to a state-of-the-art SMT system. They showed that integrating language models into an encoder-decoder framework can reach and even improve the performance of character-level SMT methods for that language.

In this work, we also consider the normalization task as a MT problem and test both statistical and neural machine translation. For SMT, we explore the added value of varying background corpora for training the language model. For NMT, we investigate whether we can overcome the limited data set size by using data augmentation.

3 Methodology

Our objective is to go from noisy to standard text and we tackle this normalization problem using a Machine Translation (MT) approach. As in general MT, a translation model is trained on parallel data consisting of pairs (x, y) of source sentences/words (= noisy text) and their corresponding target equivalents (= standard). Table 1 lists some examples of the noisy data we are dealing with.

3.1 Parallel Corpora

We relied on existing Dutch (Schulz et al., 2016) and English (De Clercq et al., 2014) corpora that were manually normalized[1] (Table 2). Three genres were included for both languages:

Tweets (TWE) which were randomly selected for both languages from the social network.

Message board posts (SNS) which were in both languages sampled from the social network Netlog, which was a Belgian social networking website targeted at youngsters.

Text messages (SMS) which were sampled from the Flemish part of the SoNaR corpus (Treurniet et al., 2012) for the Dutch language and from the NUS SMS corpus (Chen and Kan, 2013) for English.

Genre	# Sent.	# Words		%
English		Ori	Tgt	
TWE	810	13477	13545	0.50
SNS	2592	26881	27713	3.00
SMS	1435	20663	22946	**3.94**
Dutch				
TWE	842	13013	13024	0.08
SNS	770	11670	11913	2.04
SMS	801	13063	13610	**4.19**

Table 2: Parallel corpora data statistics in both languages.

Table 2 shows the number of parallel sentences in each corpus and the number of words before and after normalization. Regarding the level of noise, we observe that the text messages required most normalization operations in both languages (an increase of 3.94% for English and one of 4.19% for Dutch). We also notice that the Dutch tweets required hardly any normalization (0.08%). This can be explained by the fact that this platform has been mainly adopted by professionals in Belgium who write in a more formal style (Schulz et al., 2016).

3.2 SMT Approach

The core idea behind SMT relies on the noisy channel model. In this task, two basic components are integrated:

$$\underset{y \in W}{\operatorname{argmax}}\ P(y|x) = \underset{y \in W}{\operatorname{argmax}}\ P(x|y)P(y)$$

The translation model $P(y|x)$ is responsible for the correctness of the translation from the source $x = x_1, x_2, ..., x_m$ to the target sentence $y = y_1, y_2, ..., y_n$. The language model $P(y)$ is responsible for the fluency of the sentence in the target language. W is the set of all target sentences.

To achieve better context-sensitive source-target mappings, traditional SMT systems rely on phrase-level translation models. These models allow to build a phrase table to store aligned phrase pairs in the source and target language. This is a difficult task since one word in one language may correspond to several words in the other language. However when translating from noisy to standard text we can assume that most of the words have a one-to-one mapping. Figure 1 illustrates the architecture of an SMT system.

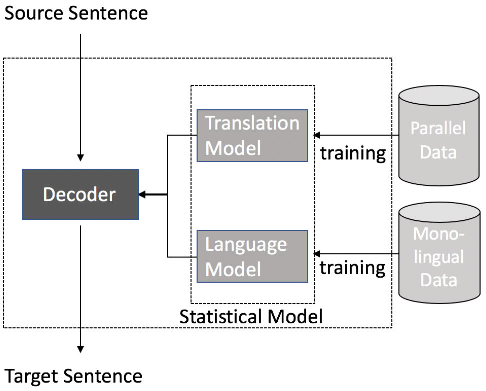


Figure 1: SMT architecture.

For social media translation we suspect that depending on the level of noise of the parallel data, the use of different monolingual corpora for training the language model should lead to better results. Due to the unavailability of a monolingual social media text corpus, we needed to find a resource that somewhat resembles this specific domain. We chose to work with existing corpora comprising two flavors of transcribed speech, namely subtitles and transcriptions of parliamentary debates, because we believe that these can better represent, to some extent, the user-generated content that we can find in social media texts. Table 3 represents the different corpora we used for our experiments. For the English experiments we relied on three different background corpora for constructing our language models: the OpenSubtitles corpus (OPUS) (Tiedemann, 2012b) which is a collection of documents from http://www.opensubtitles.org/; the Europarl corpus (Koehn et al., 2006), extracted from the proceedings of the European Parliament; and the combination of both. A similar approach was

[1] All data was normalized following the procedure described in Schulz et al. (2016)

taken for Dutch, for which we used an in-house subtitles dataset, Europarl, and the combination of both.

Corpus	Sentences
English	
OPUS	22,512,649
Europarl	2,005,395
Combined	OPUS+Europarl
Dutch	
Subtitles	8,056,693
Europarl	2,000,113
Combined	OPUS+Europarl

Table 3: Size (expressed in sentences) of the monolingual corpora used for training our LMs.

3.3 NMT Approach

Neural Machine Translation incorporates the advantages of newly developed deep learning approaches into the task. Sequence-to-Sequence (seq2seq) models have been used for a variety of NLP tasks including machine translation obtaining state-of-the-art results (Luong et al., 2015; Young et al., 2018). In this approach both input and output sentences are going in and out of the model. As described in the literature overview, the model consists of two neural networks: an encoder and decoder (See Figure 2). The encoder extracts a fixed-length representation from a variable-length input sentence (*A B C D*), and the decoder generates a correct translation from this representation (*X Y Z*). In the figure <eos> marks the end of a sentence. The encoder-decoder model is trained on a parallel corpus consisting of aligned source sentences and their normalized forms (see Table 1).

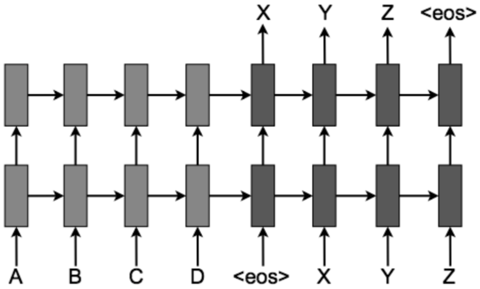


Figure 2: Encoder-decoder architecture. The light-color nodes represent the encoder and the dark-color ones the decoder. Image taken from Luong et al. (2015).

Neural systems, however, require huge amounts of data in order to perform properly. The training data we have available for text normalization amounts to only a few hundred sentences, as can be derived from Table 2. Moreover, manually annotating more training is highly time-consuming. Under these conditions, we decided to experimentally verify whether it is more beneficial to use a data augmentation technique (step A) which possibly resolves the data scarcity problem (Saito et al., 2017) or to annotate more data (step B). We tested this on the Dutch corpus and one particular genre, namely text messages (SMS). For step A, we augmented the parallel data by duplicating monolingual subtitles data on both the source and target side. For step B, we sampled and manually annotated ten thousand extra tokens from the Flemish part of the SoNaR corpus (Treurniet et al., 2012)[2].

We relied on OpenNMT[3] to train our encoder-decoder model. OpenNMT is an open source (MIT) initiative for neural machine translation and neural sequence modeling (Klein et al., 2017). The main system is implemented in the Lua/Torch mathematical framework, and can easily be extended using Torch's internal standard neural network components. We used the version of the system with the basic architecture which consists of an encoder using a simple LSTM recurrent neural network. The decoder applies attention over the source sequence and implements input feeding (Luong et al., 2015).

3.4 Evaluation

For evaluating the results of the normalization we calculated Word Error Rate (WER), a commonly used machine translation evaluation metric. WER is derived from the Levenshtein distance (Levenshtein, 1966), working at the word level instead of the character level. It takes into account the number of insertions (INS), deletions (DEL) and substitutions (SUBS) that are needed to transform the suggested string into the manually normalized string. The metric is computed as follows:

$$WER = \frac{INS + DEL + SUBS}{N}$$

where N in the number of words in the reference.

Table 4 reports WER computed between the original and target parallel sentence pairs that were

[2] Following the same annotation guidelines as Schulz et al. (2016)

[3] http://opennmt.net

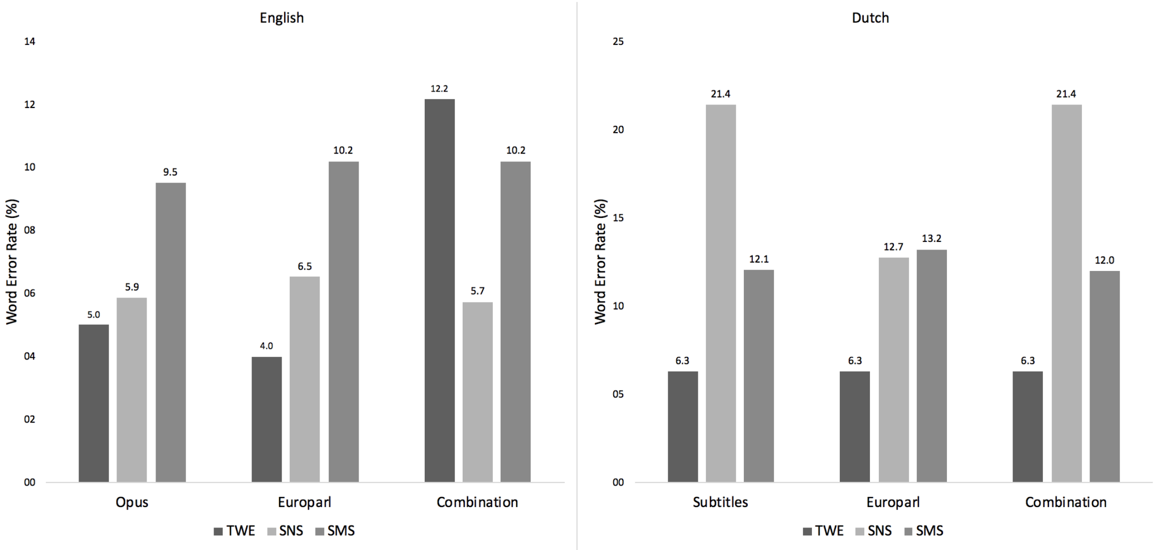


Figure 3: Normalization results at the token level. The left chart presents the results on the English datasets and the right one the results on the Dutch dataset.

used for training our models.

Word Error Rate (%)		
Genre	English	Dutch
TWE	12.160	10.592
SNS	15.400	21.390
SMS	17.190	25.130

Table 4: WER values (in percentage) at the sentence level

WER values were calculated per sentence and averaged within the document. The higher the value, the more operations are needed to obtain the target sentence. Looking at the values, we again notice that genres requiring the most and least normalization are the text messages (SMS) and tweets (TWE), respectively.

4 Experiments

4.1 Varying Background Corpora for SMT

With the first round of experiments we want to research the influence of varying the monolingual data that are used to construct the language models. We trained LMs at the character level using unigrams and bigrams and at the token level. All LMs were built using the SRILM toolkit (Stolcke, 2002) with Witten-Bell discounting which has proven to work well on small data sets (Tiedemann, 2012a; Mahmudul Hasan et al., 2012; De Clercq et al., 2013). To evaluate the performance of each LM, Word Error Rate (WER) was calculated.

The parallel data (Table 2) used for training the translation model were divided using 80% for training the model and 10% for development and testing, respectively. The target sentences from the training set were also added to the monolingual corpus for training the language model.

Despite several works reporting better results when using a character-level approach (De Clercq et al., 2014; Lusetti et al., 2018) our experiments revealed the best performance with SMT at the token level. The bar charts in Figure 3 present the results of SMT at the token level with the different LMs.

Regarding the monolingual background corpora, we notice that Europarl leads to the best results for the tweets (TWE) genre which was actually the genre with the least noise (see Section 3.1). Our experiment shows WERs of 4% and 6.3% for English and Dutch, respectively. This result was to be expected as the word usage in Europarl is mostly standard and therefore close to the word usage in the tweets. The same is true for the genre comprising the most noise, i.e text messages. The word usage in the Subtitles/OPUS dataset is less standard and closer to spoken language and, indeed, also in this case we obtained a WER of 9.5% for English using OPUS, and a WER of 12% for Dutch using a combination of the Subtitles dataset and Europarl.

In addition, we also computed the number of insertions (INS), deletions (DEL) and substitutions

(SUBS) in the original sentence pairs (Ori) and after normalization (Norm) (Table 5).

English						
	INS		DEL		SUBS	
Genre	Ori	Norm	Ori	Norm	Ori	Norm
TWE	91	36	15	**30**	34	22
SNS	354	64	66	43	109	62
SMS	377	63	44	**57**	135	54
Dutch						
TWE	22	19	0	**5**	39	**45**
SNS	386	159	18	**36**	76	47
SMS	0	0	2	**3**	94	85

Table 5: Number of operations required before (Ori) and after (Norm) normalization.

Ideally, the number of operations after normalization should be reduced to zero. As can be derived from the table many operations were strongly reduced; however, many cases still need to be solved. We will have a closer look at some of these cases in the next section.

4.2 Error Analysis of the SMT Results

The number of remaining insertions is mostly linked to the problem of abbreviation expansions. Very common abbreviations like *lol* or *omg* are always corrected, whereas others like *r.e.* and *p.e.* for *religious* and *physical education* or *cum on den* for *come on then* are not corrected since they never appear in the training data.

When we consider the deletions, we can observe that flooding or repetition of characters is often not solved with SMT. For example, tokens like *okkk*, *awwwww* or sentences like *immaaa dooiin fiine !* remained unchanged. A straightforward way to overcome this problem could be to reduce the number of repetitions to two or three in some cases as a pre-normalization step. The second factor that affects the number of deletions is the hypernormalization of some words. This leads to an increase in the number of operations since sometimes we will have to perform more deletion operations on the predicted sentences than on the original ones (these instances are indicated in bold in Table 5). This is for example the case with the name *al* which was incorrectly normalized to *all* or the normalization of *i can 't really think...* into *i can not really not think*. These problems also affect the number of substitutions. In general, we also notice that the normalization of Dutch presents a higher number of errors.

4.3 NMT Approach to UGC Normalization

As we explained before, neural approaches have obtained state-of-the-art results for the task of Machine Translation. Neural approaches however, are well-known to require big amounts of parallel data in order to perform properly. Especially for Dutch, which can be considered a low-resource language, it is difficult to find freely available parallel data and annotating new data is both money and time-consuming.

Under these conditions we decided to experimentally verify whether it is more beneficial to use a data augmentation technique (step A) which possibly resolves the data scarcity problem (Saito et al., 2017) or annotate more data (step B). We tested this on the Dutch corpus and one particular genre, the most noisy one, namely text messages (SMS).

For Step A, our idea was to make use of the monolingual subtitles corpus in both sides of the parallel data in order to augment the number of sentences available for training our model. Doing this, we obtain a bigger dataset consisting of the Dutch SMS parallel corpus and the Dutch Subtitles dataset which is duplicated in the source and target data. That is a total of 8,057,334 parallel sentences for training.

src sent.	wa gaat je doen ? xxx
norm sent.	wat gaat je doen ? xxx
src sent.	oeiiii misterieus <emoji> xxx
norm sent.	oeiiii misterieus <emoji> xxx
src sent.	dne dvd vn is ni goe ze . ge kunt nx zien . mt betale . x
norm sent.	het dvd hem is niet niet het maar wat wat in ik . niet betale . x x x x x

Table 6: Original (src) and predicted (norm) sentences using the NMT approach.

Unfortunately, as can be derived from the table above, results following this approach are very poor. It is common in the output to find repetition of words like in the third sentence (*niet, wat* and *x*). Besides, some sentences that needed normalization like the second sentence in the table, were not normalized at all. For example, the words *wat* and *niet* in the first and last sentence respectively, were correctly normalized. These results may have been determined to a great extent by the unbalance in the parallel sentences. However, we could see a slight improvement compared to the results using only the small parallel data in this architecture. For that case, the system output consisted of sentences of the type <*emoji*> *, de , , , . . .* <*emoji*>. These are random repetitions of the most represented tokens like *ik* (*I* in English), punctuation marks or <*emoji*> labels.

In order to corroborate our other hypothesis, we collected and manually normalized more data for step B. In order to check how this system works at different levels of granularity, we also performed experiments using bigram and unigrams of characters. Regarding the results, also for NMT the results are better at the word level than at the character level, with WERs of 15% instead of 29% and 26% for bigram and unigram, respectively.

4.4 Error Analysis of the NMT Results

Using the new data configuration the system is capable to correctly translate sentences like the first one in Table 7.

src sent.	aahn , ok ma cva dan kzal dan wel wa zoeken xp merci eh x
norm sent.	ah , oké maar ça va dan ik zal dan wel wat zoeken <emoji> merci h x
tgt sent.	ah , oké maar ça va dan ik zal dan wel wat zoeken <emoji> merci h x
src sent.	zal dan eentje **v** mezelf sturen . zorgen we morgenavond dan voor verrassing **v** kareltje ?
norm sent.	zal dan eentje van mag sturen . zorgen we morgenavond dan voor cocktail van droomt ?
tgt sent.	zal dan eentje van mezelf sturen . zorgen we morgenavond dan voor verrassing voor kareltje ?

Table 7: Original (src), predicted (norm) and target (tgt) sentences using the NMT approach trained on the extended dataset.

However, the system still produces a large number of odd normalizations. In the second sentence in Table 7, for example, only the bold words should have been normalized. However, only one of those two words was correctly normalized, the other one was normalized but not into its correct form. On the other hand, the system also produces odd translations of words that already were in their standard form. For example the word *mezelf* is changed to *mag* and we got *cocktail van droomt* instead of the desired normalization, ie. *verrassing voor kareltje*.

In general, while the results using this approach are very poor, the experiments revealed that having a bigger parallel training corpus could improve the performance of this system.

5 Conclusions and Future Work

In this article, we have presented two different approaches to text normalization of social media text: statistical and neural machine translation. We applied text normalization to English and Dutch text from different genres: text messages, message board posts and tweets. Best results were achieved at the token level for all genres and for both SMT and NMT.

For the SMT experiments, regarding the different corpora that were used to construct the LM, we found that Europarl gave the best results for the least noisy genre (tweets). The same is true for the noisiest genre (text messages). Considering our results, it seems to be important to make variations in the background data for building the LM, depending on the amount of noise and vocabulary that is present in the genre. With respect to the remaining errors we believe that following a modular approach instead of only using SMT could lead to a much better performance.

Our NMT approach performs poorly due to the scarcity of the data, although we did find that for a low-resource language like Dutch adding additional training data works better than artificially augmenting the data. The data augmentation technique used, however, was very basic and we believe that other techniques could lead to better results, such as hand-crafted rules for the production of abbreviations or the use of previously trained embedding in order to build similar sentences helping to generalize better.

Exploring those other data augmentation techniques is a first avenue for future work. Besides we also want to test the benefits of the integration of a neural LM in the encoder-decoder model to help with the translation of out-of-vocabulary words.

References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for SMS text normalization. *Proceedings of the COLING/ACL on Main conference poster sessions* - (July):33–40. https://doi.org/10.1109/TCST.2005.854339.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint* pages 1–15. http://arxiv.org/abs/1409.0473.

Richard Beaufort and Sophie Roekhaut. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics* 1(July):770–779. https://doi.org/10.1097/ICO.0b013e318245c02a.

Su Lin Blodgett, Green Lisa, and OĆonnor Brendan. 2016. Demographic Dialectal Variation in Social Media: A Case Study of African-American English. *arXiv preprint* .

Tao Chen and Min Yen Kan. 2013. *Creating a live, public short message service corpus: The NUS SMS corpus*, volume 47. https://doi.org/10.1007/s10579-012-9197-9.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *International Conference on Learning Representations ICLR* http://arxiv.org/abs/1406.1078.

Mason Chua, Daan Van Esch, Noah Coccaro, Eunjoon Cho, Sujeet Bhandari, and Libin Jia. 2018. Text Normalization Infrastructure that Scales to Hundreds of Language Varieties. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*. European Language Resource Association, Miyazaki, Japan, pages 1353–1356. http://www.lrec-conf.org/proceedings/lrec2018/pdf/8883.pdf.

Eleanor Clark and Kenji Araki. 2011. Text normalization in social media: Progress, problems and applications for a pre-processing system of casual English. *Procedia - Social and Behavioral Sciences* 27(Pacling):2–11. https://doi.org/10.1016/j.sbspro.2011.10.577.

Orphée De Clercq, Sarah Schulz, Bart Desmet, and Véronique Hoste. 2014. Towards Shared Datasets for Normalization Research. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* pages 1218–1223.

Orphée De Clercq, Sarah Schulz, Bart Desmet, Els Lefever, and Véronique Hoste. 2013. Normalization of Dutch User-Generated Content. *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)* pages 179–188.

Sukanya Dutta, Tista Saha, Somnath Banerjee, and Sudip Kumar Naskar. 2015. Text Normalization in Social Media. In *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*. IEEE, Kolkata, India, c, pages 378–382. https://doi.org/0.1109/ReTIS.2015.7232908.

Jacob Eisenstein. 2013. What to do about bad language on the internet. *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference* (June):359–369.

Rob Van Der Goot. 2015. Normalizing Social Media Texts by Combining Word Embeddings and Edit Distances in a Random Forest Regressor. In *Normalisation and Analysis of Social Media Texts (NormSoMe)*. 1.

Bo Han and Timothy Baldwin. 2011. Lexical Normalisation of Short Text Messages : Makn Sens a #twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics* pages 368–378.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese Text Normalization with Encoder-Decoder Model. *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)* pages 129–137.

Max Kaufmann. 2010. Syntactic Normalization of Twitter Messages. *International conference on natural language processing* 2:1–7.

Jinyun Ke, Tao Gong, and William S-y Wang. 2008. Language Change and Social Networks. *Communications in Computational Physics* 3(4):935–949.

Guillaume Klein, Yoon Kim, Yuntian Deng, Josep Crego, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source Toolkit for Neural Machine Translation. *arXiv preprint* http://arxiv.org/abs/1709.03815.

Catherine Kobus, Francois Yvon, and Geéraldine Damnati. 2008. Normalizing SMS: are two metaphors better than one? *Proceedings of the 22nd International Conference on Computational Linguistics* 1(August):441–448. https://doi.org/10.14288/1.0064682.

Philipp Koehn, Wade Shen, Marcello Federico, Nicola Bertoldi, Chris Callison-Burch, Brooke Cowan, Chris Dyer, Hieu Hoang, Ondrej Bojar, Richard Zens, Alexandra Constantin, Evan Herbst, and Christine Moran. 2006. Moses: Open Source Toolkit for Statistical Machine Translation. *Proceedings of ACL* (June):177–180. https://doi.org/10.3115/1557769.1557821.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions. *Soviet physics doklady* 10(8):707–710.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint* http://arxiv.org/abs/1508.04025.

Massimo Lusetti, Tatyana Ruzsics, Anne Göhring, Tanja Samardi Samardžic, and Elisabeth Stark. 2018. Encoder-Decoder Methods for Text Normalization. In *Proceedings of the Fifth*

Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial). pages 18–28. http://www.aclweb.org/anthology/W18-3902.

A. S. M. Mahmudul Hasan, Saria Islam, and M. Mahmudul Rahman. 2012. A Comparative Study of Witten Bell and Kneser-Ney Smoothing Methods for Statistical Machine Translation. *Journal of Information Technology (JIT)* 1(June):1–6.

Soumil Mandal and Karthick Nanmaran. 2018. Normalization of Transliterated Words in Code-Mixed Data Using Seq2Seq Model & Levenshtein Distance. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*. Association for Computational Linguistics, Brussels, Belgium, pages 49–53. https://en.wikipedia.org/wiki/ISO http://arxiv.org/abs/1805.08701.

Itsumi Saito, Jun Suzuki, Kyosuke Nishida, and Kugatsu Sadamitsu. 2017. Improving Neural Text Normalization with Data Augmentation at Character- and Morphological Levels. *Proceedings of the The 8th International Joint Conference on Natural Language Processing* pages 257–262. http://aclweb.org/anthology/I17-2044.

Sarah Schulz, Guy De Pauw, Orphée De Clercq, Bart Desmet, Véronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular Text Normalization of Dutch User-Generated Content. *ACM Transactions on Intelligent Systems and Technology* 7(4):1–22. https://doi.org/10.1145/2850422.

Andrew H. Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. 2013. Personality, Gender, and Age in the Language of Social Media : The Open-Vocabulary Approach. *PLoS ONE* 8(9). https://doi.org/10.1371/journal.pone.0073791.

Alexey Sorokin. 2017. Spelling Correction for Morphologically Rich Language: a Case Study of Russian. *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing* (April):45–53. https://doi.org/10.18653/v1/w17-1408.

Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech and Language* 15(3):287–333. https://doi.org/10.1006/csla.2001.0169.

Richard Sproat and Navdeep Jaitly. 2016. RNN Approaches to Text Normalization: A Challenge. *Computing Research Repository (CoRR)* abs/1611.0. http://arxiv.org/abs/1611.00068.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing.*. Denver, Colorado, pages 901–904. https://doi.org/10.1.1.157.2429.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*, pages 3104–3112. http://arxiv.org/abs/1409.3215.

Jörg Tiedemann. 2012a. Character-Based Pivot Translation for Under-Resourced Languages and Domains. *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* pages 141–151.

Jörg Tiedemann. 2012b. Parallel Data, Tools and Interfaces in OPUS. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* pages 2214–2218.

Maaske Treurniet, Henk van den Heuvel, Nelleke Oostdijk, and Orphée De Clercq. 2012. Collection of a corpus of Dutch SMS. *Proceedings of the Eight Conference of International Language Resources and Evaluation.* pages 2268–2273.

Cynthia Van Hee, Marjan Van De Kauter, Orphe De Clercq, Els Lefever, Bart Desmet, and Vronique Hoste. 2017. Noise or music? Investigating the usefulness of normalisation for robust sentiment analysis on social media data. *Revue Traitement Automatique des Langues* 58(1):63–87.

Reinhild Vandekerckhove and Judith Nobels. 2010. Code eclecticism : Linguistic variation and code alternation in the chat language of Flemish teenagers. *Journal of Sociolinguistics* 14(5):657–677.

Zhenzhen Xue, Dawei Yin, and Bd Davison. 2011. Normalizing Microtext. *Analyzing Microtext* pages 74–79.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine* 13(3):55–75. https://ieeexplore.ieee.org/abstract/document/8416973/.

Sentiment and Emotion Based Text Representation for Fake Reviews Detection

Alimuddin Melleng
Queen's University Belfast
amelleng01@qub.ac.uk

Anna-Jurek Loughrey
Queen's University Belfast
a.jurek@qub.ac.uk

Deepak P
Queen's University Belfast
deepaksp@acm.org

Abstract

Fake reviews are increasingly prevalent across the Internet. They can be unethical and harmful. They can affect businesses and mislead customers. As opinions on the Web are increasingly relied on, the detection of fake reviews has become more critical. In this study we explore the effectiveness of sentiment and emotions based representations for the task of building machine learning models for fake reviews detection. The experiment performed with three real-world datasets demonstrate that improved data representation can be achieved by combining sentiment and emotion extraction methods, as well as by performing sentiment and emotion analysis on a part-by-part basis by segmenting the reviews.

1 Introduction

The Internet has evolved into a content creation platform where people express their opinions and experiences. Online reviews written by users have significant impact on customers and companies. Potential customers often consult reviews before making a purchase. Reviews help potential customers to gain insights from other people's experiences, particularly in making choices on purchasing products or services. At the same time, companies need reviews on their products or services in order to get feedback and maintain good reputation. However, not all reviews available in the Internet are genuine. Profusion of reviews of questionable quality increase concerns about their trustworthiness. Moreover, users with malintent often post fake reviews (FR) to mislead customers by promoting or demoting products or target stores. Authors of FR can sway customer choices towards companies with which they are associated, or against competitors making fake reviews a lucrative business. There has been an increase in FR profusion lately. According to the report of the Harvard Business School (Luca and Zervas, 2016) the percentage of fake reviews on YELP[1] increased from 5 % in 2006 to 20% in 2013. This makes FR detection an important challenge to be addressed.

FR were firstly categorized by Jindal et al. (2008) into three groups: (1) Untruthful opinions: mislead readers by giving positive reviews to promote or demote target object, (2) Reviews on brands only: the reviewer focus on the brands, producers or sellers of a product or service without commenting on the product or service, (3) Non-reviews: the reviews are irrelevant to the product and do not contain opinions but advertisements or questions. The first category is the most challenging type to detect, and that is the focus of our paper. Given the large numbers of reviews posted daily, automatic methods would be preferred over manual ones as illustrated in (Ott et al., 2011). Recent years have witnessed an increased impetus on machine learning methods for data-driven FR detection (Mukherjee et al., 2013; Ott et al., 2011; Rout et al., 2017)

The performance of machine learning models for detecting FR is heavily influenced by the data representation (or features) in their application (Bengio et al., 2013). Text analytics has conventionally focused on domains such as labelling news stories or grouping disease reports based on severity where the human authors of text documents are largely passive to the usage of downstream analytics. FR mitigation methods, on the other hand, are in direct conflict with the intents of FR peddlers, generating interest-

[1] https://www.yelp.co.uk/

ing gamification dynamics. This makes it important for data-driven FR solutions to rely on more generic or higher-level data representations rather than simple lexical ones based on words, phrases and sentences. This is because FR filters using higher-level generic features may naturally be expected to be more robust and resistant to simple workarounds by FR authors such as word and phrase replacements. Further, higher-level features may have limited volatility across domains; thus, FR detection methods based on them may be more transferable across domains.

In this paper, we evaluate the effectiveness of emotion and sentiment based representations for the task of building machine learning models for FR detection. In particular, we illustrate that improved data representations can be achieved by leveraging a plurality of emotion and sentiment extraction methods, as well as by estimating emotions and sentiments on a part-by-part basis by segmenting the reviews. We illustrate the improved effectiveness of multiple emotion and sentiment features as well as review-segmented features by evaluating over real-world datasets.

2 Related Work

Representation learning focuses on developing a more instructive feature set for training a classification model that helps to boost the FR detection process (Li et al., 2017; Yilmaz and Durahim, 2018). Within past research, diverse features selection methods have been employed to detect FR. These may be divided into two classes: review-centric and reviewer-centric features. Reviewer-centric features are related to the reviewer's behaviour (Fontanarava et al., 2017) rather than the review itself. Those features include textual features, rating features, and temporal features. Review-centric features are derived from the content of a review. Commonly used review-centric features include Bag-of-words, TF-IDF (Term-frequency inverse-document- frequency), POS (part of speech) tags, word n-grams (Ahmed et al., 2018), and word embedding vectors (e.g. Word2vec, Doc2vec) (Krishnamurthy et al., 2018; Yilmaz and Durahim, 2018). A recent study by Jia et al. (2018) explored the application of linguistic features to distinguish between fake and non-fake reviews. They used Yelp filter dataset in their study and applied Term Frequency, Word2vec, and Latent Topic Distribution for data representation. They trained three machine learning models i.e. SVM, Logistic Regression, and Multi-layer Perceptron and found that LDA+Logistic Regression and LDA+Multi-layer Perceptron performed better with 81.3% of accuracy.

With representations being only a means to enable better FR identification, it is useful to briefly outline the classification techniques that have been employed for FR detection. Ott et al., (2011) used word n-gram features in combination with a SVM classifier. Banerjee and Chua (2014) employed a Logistic Regression classifier over POS tags and writing style features (e.g., tense of words) for FR detection. Algur et al., (2010) explored a similarity-oriented method for FR detection over domain-specific product features.

As mentioned earlier, our representations are centred on emotion and sentiment based features. There has been very little prior work on using such features for FR detection. An early work in sentiment analysis for FR detection was conducted by Peng and Zhong (2014), whereas (K et al., 2019) explore utility of emotions in health fake news detection. Peng and Zhong (2014) chose SentiWordNet and MPQA lexicons and analysed sentiment on review and product features. In our experiment, we used IBM, Afinn, SenticNet, and Biu Liu lexicons. To our knowledge, this is the first study detecting FR by means of combination emotion and sentiment analysis. Taking cue from the previous work of FR detection, we use Random Forest classifier, in our experiments.

3 Methodology

In this section we describe our proposed approach to online FR detection using emotion and sentiment based text representation.

3.1 Emotion and Sentiment Analysis

For the purpose of sentiment and emotion analysis, we apply three different sentiment lexicons and one emotion analysis API.

- **IBM Watson Natural Language Understanding.** Natural Language Understanding (NLU)[2] is a collection of APIs that offer text analysis through natural language processing. One of the feature of IBM Watson NLU is emotion analysis. The API takes a text as an input and returns the category which the

[2] https://www.ibm.com/services/natural-language-understanding/

text belongs to, stored in a list variable: $< KeyValuePair < String, Double >>$ e.g. ”emotion” : {”sadness”:0.336228}. Each item in the list contains the category (emotion) name and the categorization score. IBM Watson NLU can detect five emotions: anger, disgust, fear, joy, and sadness. For example, for an an input ’I love apples! I don’t like oranges’, the NLU API returns (sadness: 0.32665, joy: 0.563273, fear: 0.033387, disgust: 0.022637, anger: 0.041796).

- **SenticNet lexicon.** SenticNet[3] performs tasks such as polarity detection and emotion recognition. Instead of merely relying on word co-occurrence frequencies, it leverages semantics and linguistics. This lexicon contains a list of words with their polarity and intensity values. The intensity is a float number between -1 and +1. For example, according to the SenticNet lexicon ’abandoned’ is a negative word with intensity of -0.85. Each word in the lexicon is assigned with only one polarity and intensity value.
- **AFINN lexicon.** AFINN[4] lexicon is a list of English terms rated with valence on a scale -5 (negative) and +5 (positive). This lexicon has been manually labelled by Finn Årup Nielsen (2011). AFINN provides two versions of lexicon: the newest version AFINN-111 with 2477 words and phrases and AFINN-96 with 1468 unique words and phrases on 1480 lines. Our experiment use AFINN-111 as it is the most up-to-date version.
- **Biu Liu lexicon.** Biu Liu[5] lexicon consists of 6789 words including 2006 positive and 4783 negative words (Hu and Liu, 2004). This lexicon does not provide any sentiment scores and only provides positive/negative labels.

3.2 Representation Learning

In this work we explore whether sentiment and emotions extracted from a review can be used to train machine learning models for distinguishing between fake and non-fake reviews. We perform the sentiment/emotion analysis with different levels of granularity on a part-by-part basis by segmenting the reviews.

3.2.1 Sentiment Based Representation

The process of constructing sentiment based representation of a review is presented in Algorithm 1. We first split a review into P segments, each one containing the same number of sentences. For example, if P=4, then we split a review into 4 segments. For each segment we identify all positive and all negative words using the lexicons. In the next step, all positive sentiment values and all negative sentiment values within the segment are accumulated together. In the case of AFINN and SenticNet, all positive and negative values are summed in each segment. For Biu Liu lexicon, all positive and all negative words are counted. Following this, the segment is represented by a two dimensional vector $[pos(s_i), neg(s_i)]$, where $pos(s_i)$ and $neg(s_i)$ represent the accumulated/counted positive and negative sentiment values. Finally, all P vectors (one generated for each segment) are concatenated. The concatenated vector is returned as the sentiment representation of the entire review. The process looks the same for all sentiment lexicons.

Algorithm 1 Sentiment Based Representation

```
Input: Review $R$, number of segments $P$, sentiment lexicon $L$
Output: Sentiment representation of $R$
Split $R$ into $P$ equal segments $s_1, \ldots, s_P$
for all $s_1, \ldots, s_P$ do
    Tokenise $s_i$ into set of words $W$
    Retrieve sentiment values for all words in $W$ using $L$
    Accumulate all positive sentiment values in $W$ as $pos(s_i)$
    Accumulate all negative sentiment values in $W$ as $neg(s_i)$
    $v_i = [pos(s_i), neg(s_i)]$
end for
$v(R) := [v_1, \ldots, v_P]$
return $v(R)$
```

3.2.2 Emotion Based Representation

The process of generating emotion based representation is presented in Algorithm 2. As in the case of the sentiment based representation, a review is first divided in P segments. All sentences in each segment is then passed to the IBM Watson API. As the output we obtain vector with the five emotions’ scores. Finally, the emotion vectors obtained for all the segments are concatenated. The

[3] https://sentic.net/

[4] https://pypi.org/project/afinn/

[5] http://www.cs.uic.edu/l̃iub/FBS/opinion-lexicon-English.rar

output vector is returned as the emotion representation of the entire review.

Algorithm 2 Emotion based Representation

```
Input: Review $R$, number of segments $P$, emotion lexicon $L$
Output: Emotion representation of $R$
Split $R$ into $P$ equal segments $s_1, \ldots, s_P$
for all $s_1, \ldots, s_P$ do
    Get vector $v_i$ with emotions scores from $L$
end for
$v(R) := [v_1, \ldots, v_P]$
return $v(R)$
```

3.2.3 Multi-Segment Based Representation

The process of multi-segment representation learning is presented in Algorithm 3. With this technique, the sentiment/emotion based representation is first generated for different numbers of segments $1 \ldots P$. Following this, all vectors obtained for $p = 1 \ldots P$ are concatenated to form the final representation. In this way, the output vector contains more granular information on the distribution of sentiment or emotions within a review.

Algorithm 3 Multi-Segment Representation

```
Input: Review $R$, maximum number of segments $P$, lexicon $L$
Output: Vector representation of $R$
for all $p \in 1 \ldots P$ do
    Obtain $v_p(R)$ calling Algorithm 1 or 2 and passing $R$, $p$ and $L$ as parameters
end for
$v(R) := [v_1(R), \ldots, v_P(R)]$
return $v(R)$
```

3.2.4 Combined Sentiment and Emotion Based Representation

The last representation type that we explore is the combined sentiment and emotion based representation. The process is presented in Algorithm 4. First, a review is divided into P segments. The representation of each segment is generated by concatenation of sentiment and emotion representations obtained with Algorithms 1 and 2 respectively. Finally, representations of all segments are merged together.

Algorithm 4 Combined sentiment and Emotion Based Representation

```
Input: Review $R$, number of segments $P$, sentiment lexicon $L_s$, emotion lexicon $L_e$
Output: Vector representation of $R$
Split $R$ into $P$ equal segments $s_1, \ldots, s_P$
for all $s_1, \ldots, s_P$ do
    Get sentiment representation $V_s(R)$ applying Algorithm 1 with $R$, $P$ and $L_s$
    Get emotion representation $V_e(R)$ applying Algorithm 2 with $R$, $P$ and $L_e$
    $v_i = [V_s(R), V_e(R)]$
end for
$v(R) := [v_1, \ldots, v_P]$
return $v(R)$
```

4 Experimental Results and Discussion

In this section we present the experimental evaluation of the proposed four different sentiment/emotion based representations. Each of the representations are separetely used to build a machine learning model for FR detection. We conducted an extensive set of experiments in order to answer the following key questions:

- Do sentiment/emotion based representations help in FR detection?
- Which of the proposed representations is the most effective for FR detection?
- Can higher sentiment/emotion granularity level improve the data representation?

4.1 Experimental Setup

Datasets. We collected our datasets from two different sources. We used gold standard spam review dataset from Ott et al., (2011), and Yelp dataset from Rayana and Akoglu (2015). The Ott dataset contains reviews about hotels. Yelp Zip and Yelp NYC are extracted from Yelp filtered dataset. Yelp NYC is a collection of reviews from restaurants located in New York City (NYC) while Yelp Zip is a collection of restaurant's reviews in zip code area in NY State. Each of the datasets contains true labels of the reviews, i.e. fake or non-fake label assigned to each review. Table 1 shows the size and class distribution for each of the datasets. In our experiment, we only consider reviews that contain more than 10 sentences. We presume that proposed representation learning techniques would not be effective for short reviews since sparse text does not allow to identify emotions and sentiments well. Table 2 demonstrates the statistics of the datasets after filtering.

Dataset	Non-fake	Fake
YELP ZIP	528019	80439
YELP NYC	322097	36860
Ott	800	800

Table 1: Statistics of the datasets

Dataset	Non-fake	Fake
YELP ZIP	170261	15108
YELP NYC	105080	6185
Ott	340	270

Table 2: Statistics of the datasets after filtering

Learning. As the machine learning algorithm we used Random Forest (RF) given that it was reported as one of the most effective in FR detection (Chowdhary and Pandit, 2018; Saumya and Singh, 2018; Viviani and Pasi, 2017). However, any other learning algorithm can be applied instead. We set n_estimator=100 and random state=42 for the RF parameter. All the experiments are performed with 5-fold cross-validation and the prediction performance is evaluated with application of F-measure. Given the very high class imbalance in the Yelp NYC and Yelp Zip, we randomly select number of non-fake reviews equal to the number of FR in order to balance the training data.

4.2 Sentiment and Emotion Granularity

In this section we investigate what level of granularity in terms of sentiment and emotion is the most representative for FR detection. Tables 3-5 demonstrate the F-measure obtained by RF with each of the datasets and sentiment and emotion based representations for reviews. For the parameter P we used values from 1 to 4. For each table, the first row represents results obtained by RF applied with the emotion based representation obtained with the IBM Watson API. The three bottom columns contain results obtained for the sentiment based representation generated with each of the three sentiment lexicons. Each column refers to a different value of parameter $P = 1 \times 4$. The last column presents results obtained for multi-segment based representation.

We can observe from the tables that in the majority of cases, the higher the granularity (P) the better the prediction performance. It can also be noted that the multi-segment based representation tends to perform better than when a single segmentation is applied. The only exception is the Biu Liu lexicon, which for Yelp, Zip, and Ott obtained the best results for $P = 1$.

Lexicon	P=1	P=2	P=3	P=4	P1-4
IBM	0.570	**0.584**	**0.589**	**0.584**	**0.597**
SenticNet	0.506	0.510	0.522	0.523	0.524
Biu Liu	**0.574**	0.540	0.547	0.558	0.557
AFINN	0.550	0.542	0.549	0.555	0.563

Table 3: RF's F-measure over Yelp ZIP dataset.

Lexicon	P=1	P=2	P=3	P=4	P1-4
IBM	**0.554**	**0.569**	**0.578**	**0.569**	**0.584**
SenticNet	0.511	0.520	0.523	0.525	0.526
Biu Liu	0.546	0.523	0.543	0.543	0.555
AFINN	0.524	0.529	0.541	0.544	0.557

Table 4: RF's F-measure over Yelp NYC dataset.

Lexicon	P=1	P=2	P=3	P=4	P1-4
IBM	**0.620**	**0.605**	0.543	0.533	0.590
SenticNet	0.533	0.529	0.483	0.525	0.570
Biu Liu	0.618	0.561	**0.592**	**0.576**	**0.600**
AFINN	0.523	0.560	0.580	0.545	**0.600**

Table 5: RF's F-measure over Ott dataset.

4.3 Sentiment vs. Emotion

In this section we compare the results obtained by RF applied with the sentiment and the emotion based representations of data. We can see from Tables 3-5 that IBM emotion lexicon obtained the best performance in comparison to the three sentiment lexicons in the Yelp Zip and NYC datasets. This may be considered unsurprising since emotions provide more fine grained information for the classifiers to work with. For the Ott dataset, Biu Liu and AFINN lexicon obtained better results for some of the greater values of P.

In order to perform better comparison between the sentiment and emotion based representations we calculated average of the results obtained for each of the granularity levels: $P1, P2, P3, P4, P1 - 4$. The results are demonstrated in Figure 1. We can observe from the graphs that the IBM emotion lexicon performs significantly better than any of the other sentiment lexicons apart from the Ott dataset where it is outperformed by the Biu Liu lexicon.

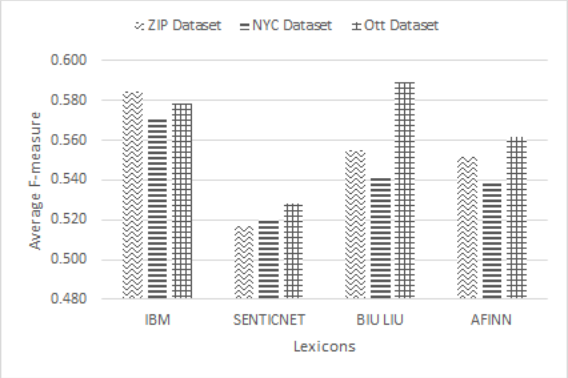


Figure 1: Average F-measure obtained for all values of P

4.4 Combined Sentiment and Emotion Based Representation

Table 6 demonstrates results obtained for the combined sentiment and emotion based representation generated according to the Algorithm 4. We can observe that for Zip and NYC datasets the best results were obtained when multi-segment based representation was applied. With Ott the best performance was obtained for $P1$.

Dataset	P1	P2	P3	P4	P1-4
ZIP	0.589	0.596	0.599	0.599	**0.602**
NYC	0.580	0.580	0.589	0.584	**0.588**
Ott	**0.653**	0.624	0.606	0.604	0.640

Table 6: F-measure obtained with combined sentiment and emotion representation learning.

In Figures 2-4 we compare the performance of the combined sentiment and emotion based representation with the emotion based representation, which so far obtained the most promising results. We can observe that the combined approach obtained better results in each case, with the difference in F-measure being quite significant for Zip and NYC datasets. This demonstrate that improved data representation can be achieved by applying combination of different emotion and sentiment extraction methods.

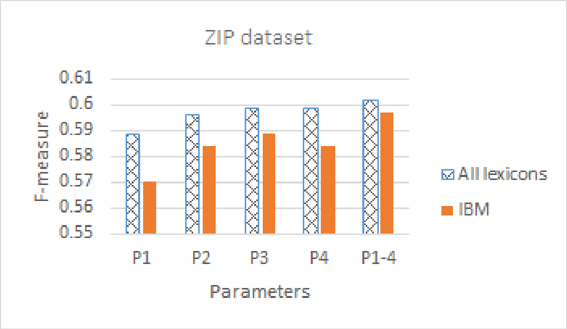


Figure 2: Combined sentiment-emotion vs. emotion representation learning for Zip

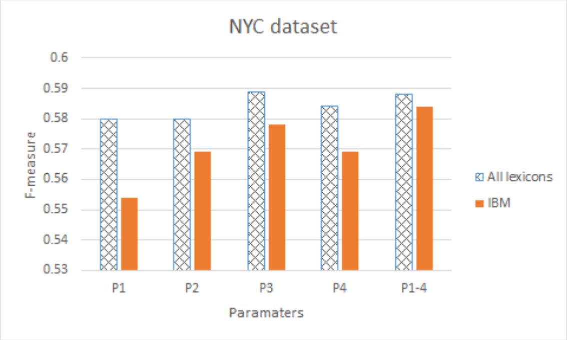


Figure 3: Combined sentiment-emotion vs. emotion representation learning for NYC

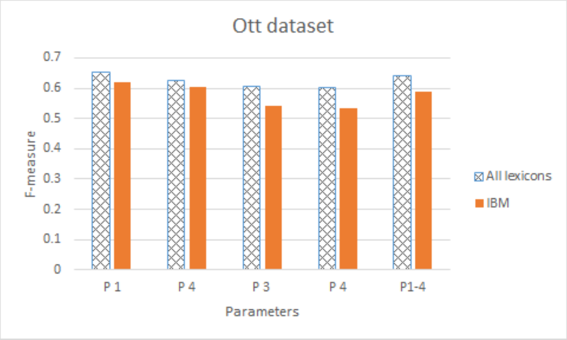


Figure 4: Combined sentiment-emotion vs. emotion representation learning for Ott

5 Conclusions and Future Work

In this paper, we analyzed the effectiveness of emotion and sentiment based representations estimated over varying text grabularities, for the task of fake review classification. Through an empirical study across three real-world datasets, we find consistent evidence that combinations of emotions and sentiments work better than either of them separately. Further, we observe that combining emotion and sentiment representations obtained across different text granularities yields better accuracies over the restaurant review datasets. As future work, we plan to carry on research on cross domain between different datasets. We also want to observe how sentiment and emotion work on neural network model such as CNN and LSTM using generic as well as custom-built lexicons (Bandhakavi et al., 2017).

Acknowledgments

The authors thank LPDP (Indonesia Endowment Fund for Education) for funding this research.

References

Ahmed, Hadeer, et al. “Detecting Opinion Spams and Fake News Using Text Classification.” *Security and Privacy*, vol. 1, no. 1, 2017, p. e9, doi:10.1002/spy2.9.

Algur, Siddu P., et al. “Conceptual Level Similarity Measure Based Review Spam Detection.” *Proceedings of the 2010 International Conference on Signal and Image Processing, ICSIP 2010*, IEEE, 2010, pp. 416–23, doi:10.1109/ICSIP.2010.5697509.

Anoop, K., et al. “Emotion Cognizance Improves Fake News Identification.” *ArXiv Preprint ArXiv:1906.10365*, 2019.

Bandhakavi, Anil, et al. “Lexicon Generation for Emotion Detection from Text.” *IEEE Intelligent Systems*, vol. 32, no. 1, IEEE, 2017, pp. 102–08, doi:10.1109/MIS.2017.22.

Banerjee, Snehasish, and Alton Y. K. Chua. “Applauses in Hotel Reviews: Genuine or Deceptive?” *Proceedings of 2014 Science and Information Conference, SAI 2014*, The Science and Information (SAI) Organization, 2014, pp. 938–42, doi:10.1109/SAI.2014.6918299.

Bengio, Yoshua, et al. “Representation Learning: A Review and New Perspectives.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, IEEE, 2013, pp. 1798–828.

Chowdhary, Neha S., and Anala A. Pandit. “Fake Review Detection Using Classification.” *International Journal of Computer Applications*, vol. 180, no. 50, 2018, pp. 16–21.

Fontanarava, Julien, et al. “Feature Analysis for Fake Review Detection through Supervised Classification.” *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2017, pp. 658–66.

Hu, Minqing, and Bing Liu. “Mining and Summarizing Customer Reviews.” *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 50, no. 08, 2004, pp. 50-4466-50–4466, doi:10.5860/choice.50-4466.

Jia, Shaohua, et al. “Fake Reviews Detection Based on LDA.” *2018 4th International Conference on Information Management, ICIM 2018*, IEEE, 2018, pp. 280–83, doi:10.1109/INFOMAN.2018.8392850.

Jindal, Nitin, and Bing Liu. “Opinion Spam and Analysis.” *Proceedings of the 2008 International Conference on Web Search and Data Mining*, ACM, 2008, pp. 219–30.

Krishnamurthy, Gangeshwar, et al. “A Deep Learning Approach for Multimodal Deception Detection.” *ArXiv Preprint ArXiv:1803.00344*, 2018.

Kumar, Abhinav, et al. “Spotting Opinion Spammers Using Behavioral Footprints.” *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2013, doi:10.1145/2487575.2487580.

Li, Luyang, et al. “Document Representation and Feature Combination for Deceptive Spam Review Detection.” *Neurocomputing*, vol. 254, Elsevier B.V., 2017, pp. 1339–51, doi:10.1016/j.neucom.2016.10.080.

Luca, Michael, and Georgios Zervas. “Fake It till You Make It: Reputation.” *Competition, and Yelp Review Fraud., SSRN Electronic Journal*, 2016.

Nielsen, Finn Årup. “A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs.” *CEUR Workshop Proceedings*, vol. 718, 2011, pp. 93–98, doi:10.1016/j.knosys.2015.06.015.

Ott, Myle, et al. “Finding Deceptive Opinion Spam by Any Stretch of the Imagination.” *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, 2011, pp. 309–19.

Peng, Qingxi, and Ming Zhong. “Detecting Spam Review through Sentiment AnalysisPeng, Q. and Zhong, M. (2014) ‘Detecting Spam Review through Sentiment Analysis’, Journal of Software. Doi: 10.4304/Jsw.9.8.2065-2072.” *Journal of Software*, 2014, doi:10.4304/jsw.9.8.2065-2072.

Rayana, Shebuti. “Collective Opinion Spam Detection : Bridging Review Networks and Metadata.” *Proceedings of the 21th ACM SIGKDD*, 2015.

Rout, Jitendra Kumar, et al. “Deceptive Review Detection Using Labeled and Unlabeled Data.” *Multimedia Tools and Applications*, vol. 76, no. 3, Multimedia Tools and Applications, 2017, pp. 3187–211, doi:10.1007/s11042-016-3819-y.

Saumya, Sunil, and Jyoti Prakash Singh. “Detection of Spam Reviews: A Sentiment Analysis Approach.” *CSI Transactions on ICT*, vol. 6, no. 2, Springer, 2018, pp. 137–48.

Viviani, Marco, and Gabriella Pasi. “Quantifier Guided Aggregation for the Veracity Assessment of Online Reviews.” *International Journal of Intelligent Systems*, vol. 32, no. 5, Wiley Online Library, 2017, pp. 481–501.

Yilmaz, Cennet Merve, and Ahmet Onur Durahim. “SPR2EP: A Semi-Supervised Spam Review Detection Framework.” *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018*, 2018, pp. 306–13, doi:10.1109/ASONAM.2018.8508314.

Turning Silver into Gold: Error-Focused Corpus Reannotation with Active Learning

Pierre André Ménard, Antoine Mougeot
Centre de recherche informatique de Montréal
pierre-andre.menard@crim.ca, antoine.mougeot@crim.ca

Abstract

While high quality gold standard annotated corpora are crucial for most tasks in natural language processing, many annotated corpora published in recent years, created by annotators or tools, contains noisy annotations. These corpora can be viewed as more *silver* than *gold* standards, even if they are used in evaluation campaigns or to compare systems' performances. As upgrading a silver corpus to gold level is still a challenge, we explore the application of active learning techniques to detect errors using four datasets designed for document classification and part-of-speech tagging.

Our results show that the proposed method for the seeding step improves the chance of finding incorrect annotations by a factor of 2.73 when compared to random selection, a 14.71% increase from the baseline methods. Our query method provides an increase in the error detection precision on average by a factor of 1.78 against random selection, an increase of 61.82% compared to other query approaches.

1 Introduction

As machine learning is increasingly predominant in natural language processing tasks, the need for annotated data, and more specifically linguistic annotations, intensify greatly. Tasks like bias detection, named entity detection (NER) and recognition, part-of-speech (POS) tagging, semantic role labeling, assessment evaluation in discourse, dependency parsing and sentiment analysis can all be modeled as machine learning problems. They require a large amount of human annotated information to enrich raw textual resources. Creating large annotated resources for these tasks requires a lot of time and effort to insure a carefully planned and well-defined annotation protocol, obtain and encode expert knowledge, do curation steps and so on. Even when using highly qualified human annotators, errors can always make their way into the final annotated corpus. Another way to obtain annotations is to apply existing state-of-the-art algorithms (also trained on previously annotated data) on a raw corpus.

While some researchers publish new versions of their hard earned annotated resources based on the feedback obtained from users, others often have to set them aside to work on other projects and resources, leaving the corpora with their original errors. One way to improve these resources without the same level of effort would be to reannotate them using an active learning process to quickly find errors and discrepancies and resubmit them to expert annotators.

This article relates an experiment which explores the potential application of active learning for corpus reannotation, which context is detailed in Section 2 with related works listed in Section 3. Two new approaches and multiple baselines are then explained in Section 4, followed by the list of datasets used (Section 5) and the experiments combining all these elements (Section 6). We then comment on the experiment (Section 7) and close with a review of the work done and some future works in the final section.

2 Context

This section gives an overview of the two main aspects of this research, namely corpus reannotation and active learning, with their challenges and possibilities. The current work is at the crossroad of these two topics.

Proceedings of Recent Advances in Natural Language Processing, pages 758–767,
Varna, Bulgaria, Sep 2–4 2019.

2.1 Corpus Reannotation

The errors present in annotated corpora can have many sources. They can be simple typographical errors created during transcription or from the begining on uncurated information. On a higher cognitive level, another frequent cause of error is annotators' discrepancies over expert knowledge, in which disagreement over labels are not solved in a consistent way. Protocol inconsistencies are yet another source of noise in annotated corpora, often encountered when unforeseen cases fail to be brought to light, managed, or added to the protocol. Moreover, there is also all the possible causes from the annotator side, like misunderstanding the protocol, errors caused by fatigue, solving ambiguities with the most favorable case instead of reporting it, and many others. All these issues can diminish the quality of the annotations.

The nature of an annotation task can also influence the risk of generating errors. Intuitively, fine-grained classifications are often considered more error-prone than those with a small number of class values. This can be attributed to fuzzy frontiers between close-by values, overlapping values (i.e. choosing between ”entertainer” and ”comedian”) or failing to recall a specific classification option in a very large set of values.

Errors might also occur based on the interpretation of different annotators on the same case. This can be the case in named entity classification tasks, when an occurrence can be considered both an organization and a location (i.e. “I went to register at the office of University X”). These are edge cases that are often overlooked in annotation protocols and might result in inconsistencies in the final annotated corpus.

A silver standard corpus (Rebholz-Schuhmann et al., 2010) is usually defined as a noisy set of ground truth annotations provided automatically by state-of-the-art algorithms, while *gold* labels are the higher quality annotations created by expert annotators. The silver labels are normally produced manually by human agents, but can also be obtained automatically by tools or trained prediction models. Of course, the gold labels might still contain some degree of noise, but they are considered of better overall quality than the silver version.

Corpus reannotation can be a tedious undertaking, as it not only requires the same expert domain knowledge to correct the annotations, but also a deep understanding of the original protocol, often created by another team. It may also require to modify the protocol or classification values, and also require an additional effort of the annotators to assess if they are not creating more errors when modifying an original answer. For all these reasons, it is important to explore ways by which some of the effort might be lowered, such as the application of active learning on noisy corpora.

2.2 Active Learning

Active learning (Cohn et al., 1996) has been used to lower the annotation effort needed to train a prediction model in natural language processing tasks. It traditionally involves a dialogue between one (or more) human annotator(s) and a machine learning algorithm, the former being proficient at annotating instances that the latter is providing based on relevance measures. The process can be split into three distinct steps.

The first one, called *seeding*, is where the algorithm must choose instances without being able to rely on any annotation from the expert. The chosen instances are submitted to be annotated by the expert. This starts the second and longest step, the *querying* phase. The active learning engine iterates between training a prediction model with the gold values (training set) from the expert, choosing new instances relevant to training a better model, submitting them to the expert and adding the expert's answers back into the training set. The third step, *stopping*, is applied after each training of the querying phase. It checks if there is enough information in the model to annotate the rest of the corpus automatically so that the annotation can stop. A bad stopping criterion might overfit the model, lowering its predictive power.

One challenge of active learning is to balance between specializing the classification on known cases (for example, annotation errors), thus improving the performance for current classes, or exploring the problem space to find unknown but relevant instances that could improve the overall performance of the model.

The goal of the current experiment is not to produce the best prediction model, but to explore if the reannotation effort on a corpus can be reduced by using the active learning process with different algorithms. These algorithms should not target the most informative instances for a prediction model, but instead choose those that are more likely to be errors. If successful, the model should then be able to extract more errors as it progresses.

3 Related Works

While different flavors of reannotation have been used for natural language processing tasks on text corpora, the work of (Rehbein and Ruppenhofer, 2017) is the most similar to our contribution from an error finding perspective with active learning. They evaluate the query-by-committee (QBC) and variational inference (VI) active learning methods to perform error detection on part-of-speech tagging and named entity recognition (NER) tasks. These approaches were tested in four contexts: in-domain (same type of training and testing data) on English POS, out-domain (different training and testing) on English POS, new task and language on German NER, and real-world context with human annotators on POS task. After 1,000 iterations, the VI approach generally gives a better noise reduction than QBC. While the POS task is similar to the one used in our experiment (although on a different language), the NER task uses only 4 class values (location, organization, person and miscellaneous) while documents in our classification task can be categorized into a high number of types. They do not specify the experimental seeding methods used to choose the first examples.

Another similar research is Skeppstedt (2013) which uses active learning with two sources of tools generated annotation in order to tag and classify named entities in Swedish clinical text documents. The challenge of combining multiple pre-annotated sources and active learning is to provide the right quality of information. If the sources are too noisy, the task will be more difficult and unreliable. On the other hand, providing high-quality sources might lower the attention and interest of the annotators. The proposed method tries to overcome these two points by showing the sources without specifying which one is most likely to be correct. No performance evaluation was done for the proposed approach.

Other experiments make usage of preannotated information without applying active learning methods, like (Chou et al., 2006) who use a semantic role labeling tool trained on PropBank to pretag a biomedical corpus called BioProp. After the automatic annotation step, a human annotator manually checks the silver values and corrects them as needed. Other reannotation efforts may be conducted by adding human resources to the task to distribute the effort among multiple users. This is the proposition made in (Hovy et al., 2014) by applying crowd sourced reannotation. In the same multiuser settings, (Lin et al., 2016) propose an approach to reannotate labels by integrating another human oracle in order to improve the quality of the annotations.

4 Methods

In order to improve noise detection, we focus on the seeding and querying steps of the active learning process. As the goal is to facilitate annotation of incorrect annotation, and not to create a prediction model per se, we did not explore the stopping phase. The following sections detail the baselines as well as the new methods used for the experiments in Section 6.

4.1 Seeding Methods

The seeding method's main goal from a reannotation perspective is to provide the highest error ratio for the budgeted seed size. This contrasts with a usual active learning task which is to find the most informational instances to annotate in order to improve the model's performances.

To capitalize on the silver classification information, we used outlier detection methods separately on each unique silver value. Our hypothesis for this is that most of the annotations should be of good quality, although noisy, meaning that clustering the instances for a single silver value should produce one or many clusters of correctly classified instances. A large enough dataset should then provide a cluster for each valid manifestation (depending on the features used) of a silver value. As the clusters represent valid cases, the outliers should represent either rare cases or, ideally, noisy labels which should be reannotated by the expert.

For each silver value of a corpus, a random instance was chosen in the detected outliers, to test the hypothesis that they should mostly be incorrectly annotated cases.

Four other outlier detection methods were tested as baselines to assess their performances and compare them to the above method. These are not normally used in the seeding phase and they do not consider the presence of a silver value in the feature set. The first is the one-class SVM (Schölkopf et al., 2001) which trains a support vector machine with a radial basis kernel function and returns the lowest supported instances in a distribution.

The local outlier factor (Breunig et al., 2000) computes the local deviation of density for an instance with its closest neighbors using a k-nearest approach. If the local density of the close-by instances is significantly higher, the instance is considered an outlier and is selected for human annotation.

The isolation forest algorithm (Liu et al., 2008) detects outliers by selecting the most isolated instance of a randomly selected feature set and split point in the value range. These splits are then projected into a tree structure and the average path length to an instance gives its degree of isolation, choosing the highest degree of isolation as the best potential instance to annotate.

Finally, the covariance detector (Rousseeuw and Driessen, 1999) uses a Gaussian distribution around the density cluster to assess the degree to which an instance might be part of that cluster.

4.2 Querying Methods

The proposed *double centroid* approach is based on the hypothesis that an annotator, either human or tool, tend to produce similar types of errors through the annotation process. The source of these types could be the inability to differentiate between two classes, unknown terminology, etc.

The method first use density-based clustering to group together newly annotated instances from the seeding or previous querying steps. It is applied once on erroneous instances, where the silver and new gold values did not match, and once on non-erroneous instances, where the silver and new gold values matched. This gives multiple clusters containing either noisy (C_n) or matching (C_m) annotations.

$$rank(l) = \left| \sum_{i=1}^{C_n} \frac{|C_i|}{dist(l, C_i)^2} - \sum_{j=1}^{C_m} \frac{|C_j|}{dist(l, C_j)^2} \right| \quad (1)$$

As shown in equation 1, for each silver instance l that was not yet picked for relabeling, a weighted squared distance $dist(l, C)^2$ is calculated separately against each cluster centroid. The weight used is the cluster cardinality $|C|$, so that nearer and larger clusters have more influence on an instance. These weighted distances are then summed up with opposing values, in this case C_n clusters having a positive influence and C_m having a negative one. The algorithm then selects those that have the highest value, following the hypothesis that they would be similar to known errors.

Other methods often applied in a standard active learning process have also been used as a basis of comparison with the proposed method, namely *distance to centroid*, *cosine similarity*, *hierarchical clustering* and *margin query*.

Distance to centroid calculates a density center point (centroid) from each instance of a specific gold value asked from the expert annotator. Each instance is then checked against each centroid and the ones which are furthest from all points are selected.

The cosine similarity works in a similar way as the previous method but uses the cosine between the instance and the centroids to assess the proximity.

Hierarchical clustering also uses the distance to clusters as a degree of uncertainty to choose ambiguous instances, but goes one step further by splitting these instances to choose only those which are the most different from one another. This usually helps to provide a better sample to annotate and avoid ambiguous but very similar instances.

Margin query chooses the instances with the smallest difference between the most probable predicted class and the second most probable.

5 Datasets

For this experiment, we used a total of four datasets, two targeting a document classification task and two for a text sequence classification task on part-of-speech tags.

As there were no publicly available manually corrected datasets, and correcting an existing one would have been too time consuming for the scope of this project, we simulated the noise level by applying a classification tool to each manually annotated corpus. They are each presented in the following sections and Table 1 shows the size and error rate for each of them. The error rate is calculated by dividing the number of errors in the silver standard (compared to the gold standard) by the total number of elements. Sections 5.5 and 5.6 describe how the noise was generated for each corpus to calculate the error rate.

Corpus	Size	Error rate
Reuters	9,149 docs	0.0941
WoS	46,985 docs	0.3916
GSD-French	402,120 words	0.1015
Sequoia	70,572 words	0.1080

Table 1: Corpus size (documents or words) and error rate.

5.1 WOS-46895

The WOS corpus (Kowsari, Kamran et al., 2018) contains 46,985 documents in English taken from the Web of Science website. Each document contains the abstract from a published scientific paper. These documents were picked from the fields of psychology, medical sciences, biochemistry, computer science and three specialization of engineering. While each document is only classified in a single topic, terminology coverage of some topics overlaps with others, such as between biochemistry and medical sciences.

Each topic is further broken down into 134 specialized areas, varying from 9 to 53 areas for each topic. While this dataset was primarily created for hierarchical classification, we only used the 134 areas (the second layer of classification) for the purpose of this research.

5.2 Reuters

The Reuters-21578 corpus (Lewis, 2004) is composed of 10,788 English documents consolidated for the text classification challenge of document understanding conference (DUC). There are 90 categories in the corpus. However, that dataset was originally used for multi-class classification. To use it for our research, we extracted only the articles having a single class and kept only the classes that appeared more than once in the dataset. The final dataset is made of 9,149 instances across 56 categories like *acq* and *earn*. The labels' distribution is heavily skewed as two of these classes make up 75% of the dataset instances.

5.3 French-GSD

We used the French portion of the GSD corpus (McDonald et al., 2013), version 2.2 at the time of writing. We merged the three parts (dev, train, test) into a single corpus consisting of 402,426 words (16,448 sentences). While it is fully tagged with dependence trees, we only retained the 17 universal dependency-based part-of-speech tags from the open class (*ADJ*, *ADV*, *INTJ*, etc.), the closed class (*ADP*, *AUX*, *CCONJ*, *DET*, etc.) and the other class (*PUNCT*, *SYM*, *X*). The feature set for each instance included the token's position in the sentence, surface form, lemma, length, presence of space after the token and case type (capitalized, all capital letters or mixed case).

5.4 Sequoia

The Sequoia corpus (Candito and Seddah, 2012) contains 3,099 French sentences (70,572 tokens) taken from different corpora such as Europarl, *L'Est Republicain* newspaper, French Wikipedia and European Medicine Agency. It has initially been annotated with constituency trees and then converted to surface syntactic dependency trees. In our experiment, we only classified the part-of-speech (POS) information. The feature set was the same as the French-GSD corpus.

5.5 POS Processing and Vectorisation

The GSD and Sequoia corpora were initially tagged manually with universal part-of-speech tags[1] which we used as the gold standard. While dependency information was widely available in the original corpus, we removed them as they would not be available in a raw text corpus without applying a high quality dependency analyzer. We kept morphological features as listed in Section 5.3.

In order to create a silver version of the dataset, each token was automatically reannotated with TreeTagger (Schmid, 1997) to provide a new set of tags. These tags were then converted to the universal part-of-speech tagset to make them comparable with the original corpus tags.

These corpora were then vectorized to be processable by the classification algorithms, projecting the information of each instance in a feature space. The feature set was also enriched for each instance with the information from the last five tokens and the next five tokens. The sentence boundaries were respected, so that the first token in a sentence would only get the next five tokens, the second token would get the previous token and the next five, and so on. As the algorithms used cannot deal with nominal data, each feature was one-hot encoded. Once encoded, each dataset contained 1,156 features including the silver annotation.

[1] https://universaldependencies.org/u/pos/

5.6 Classification Preprocessing and Vectorisation

The text documents in the WOS and Reuters corpora were stemmed using Porter stemmer (Porter, 1997) and had their stop words removed. In order to simulate silver level annotations, the corpora were vectorized and annotated using a five-parts iteration. In other words, the corpus was split into five batches containing 20% of the corpus, a model was trained on the gold values of 80% of the corpus, recreating a new model to annotate the remaining 20%, and a different batch was swapped to be annotated each time.

The two corpora were then vectorized with two word embeddings methods, fastText (Bojanowski et al., 2016), Word2vec (Mikolov et al., 2013), and the tf-idf (Salton and McGill, 1986) statistical method. The feature set size of the output vector from each method, either produced with neural networks or statistical measure, was set to 1,000 features.

Corpus	Method	F1
Reuters	fastText	0.87
	Word2vec	0.88
	tf-idf$_3$	0.91
WOS-46895	fasttext	0.59
	Word2vec	0.57
	tf-idf$_3$	0.61

Table 2: Performances of vectorization methods.

To see which one provided the best expressivity with its features, the vectors for each corpus were evaluated with ten-fold validation using logistic regression as the passive learning algorithm. The results in Table 2 show the performances for each vectorization method on each corpus. While the results are quite close to one another, the statistical tf-idf method using unigrams to trigrams (Tf-idf$_3$) provides higher results on all datasets. For this reason, we use the feature set provided by this method for the experiments.

6 Experiment

6.1 Seeding Experiment

We use the error rate of the corpus (as previously reported in Table 1) as a random baseline for seeding, which is the equivalent of making a random selection. As studied in (Hu et al., 2010), most of the papers either use this type of seeding method for active learning, choose a fixed number of instances from each target class value (knowledge that would not be accessible in a real world setting) or simply fail to mention the method.

The methods described in the previous section were run 100 times to smooth out the randomness effect of some outlier selection methods. It also helped to show if most of the outliers were in fact valid errors to be detected. Each chosen outlier was then compared to the gold value to verify if it was an error or a valid annotation. The number of unique silver values were not scaled down to a specific seed size so as to provide an overall measure of performance.

As some clustering methods do not scale well with large datasets, we applied a feature reduction algorithm on each dataset. We used a random forest (Breiman, 2001) estimator using 100 trees, building each one with a random subset of features from the dataset, evaluating the relevance of each feature and discarding the less meaningful ones. The Sequoia and French-GSD corpora were respectively reduced to 261 and 223 features, while 250 and 334 features were retained from the Reuters and WOS-46895 corpora. The silver annotation was used as the relevance indicator for this process instead of the gold which would not be available in a real setting.

The results presented in Table 3 show an improvement across all methods compared to baseline performances. The error rate column is averaged from all the results in the experimental runs. The gain is the ratio between the average error rate and the baseline performance from Table 1 (error rate column) using a random selection or a standard seeding method.

While all seeding methods provide an improvement compared to the baseline, the local outlier factor method seems the most promising when assessed from the average gain global score over all corpora. Performances on the WOS corpus were not favored by any of the four methods and were just marginally better with the local outlier factor.

6.2 Querying Experiment

The four baseline methods detailed in Section 4 were applied to each vectorized corpus for the document classification and POS tagging tasks. To avoid boosting the performance of the error seeking query method, we did not use the previous approaches for seeding. Instead we randomly se-

Method	Measure	Reuters	WoS	Sequoia	French-GSD	Average gain
SVM	EDP	0.3576	0.4346	0.1429	0.2938	
	Gain	3.80	1.11	1.32	2.89	2.28
Covariance	EDP	0.3606	0.4331	0.1786	0.2971	
	Gain	3.83	1.11	1.65	2.93	2.38
Isolation	EDP	0.3424	0.4336	0.1929	0.2267	
forest	Gain	3.64	1.11	1.79	2.23	2.19
Local	EDP	0.4030	0.4369	0.2000	0.3729	
outlier	Gain	4.28	1.12	1.85	3.67	**2.73**

Table 3: Seeding phase error detection precision and gain.

lected a subset of 20 instances for the seeding step. This ensured that the initial training set would have about the same standard level of errors as in the rest of the corpus. The query phase selected 20 instances per iteration before retraining the random forest model. These queries were answered by an artificial annotator who had access to the corresponding gold values. Each method was run on the corpora 20 times to smooth out the variability of the random aspects of some methods.

Table 4 shows the average error detection precision (EDP) for the complete set of experiments. It is the ratio between the number of errors detected (where silver and gold labels does not match) and the total number of queried instances at a specific point. A score of 1 would mean that the algorithm only submitted errors to be reannotated by the oracle. As the active learning process can be stopped at any point, the recall score is not used as it would imply that all errors should be submitted before the end of the process.

The gain ratio is the error detection precision divided by the corresponding corpus error rate. Gain values lower than one mean a lower performance than random selection. The EDP shown was calculated after 200 annotations, meaning one seed and 19 query iterations. This number was choosen from real-world experiences, where annotators usually consider that some errors should be encountered at that point, if any are to be found.

We can see that the proposed double centroid method outperformed all the other approaches on every corpus. Most methods did not perform well on the WoS corpus, which incidentally had four times the error rate of other corpus. The best gain ratio of our method was when detecting POS tagging errors on the FR-GSD and classification errors on the Reuters corpora. Aside from double centroid, most other methods performed at the same level as the others, except for distance to centroid on the Reuters corpus.

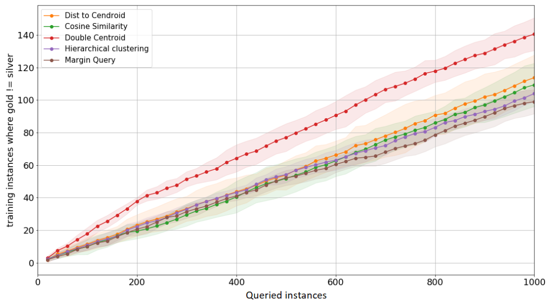


Figure 1: Performances of querying methods on the Reuters corpus.

As seen on Figure 1, which shows all methods applied to the classification task on the Reuters corpus for a total of 1,000 instances annotated, all methods seem to have a relatively stable slope. The double centroid method loses some velocity after hitting 220 instances. The other methods diverge near the end, but not very distinctively.

7 Analysis and Discussion

Looking at the slopes in Figure 1 at the start of the process, we can see that they start at about the same level of performance before settling into their tendencies. This is mainly due to the fact that the seed does not contain many instances about different gold values to provide significant clusters.

Why the performance drops on the WoS corpus compared to Reuters or other corpora can be attributed in part to the low expression power of the vectorized feature set shown in Table 2. They pro-

Method	Measure	Reuters	WoS	Sequoia	French-GSD	Average gain
Distance to	EDP	0.1194	0.3954	0.1049	0.1196	
centroid	Gain	1.27	1.01	1.03	1.11	1.10
Cosine	EDP	0.0974	0.3796	0.1198	0.1003	
similarity	Gain	1.04	0.97	1.18	0.93	1.03
Hierarchical	EDP	0.1110	0.3943	0.1196	0.1079	
clustering	Gain	1.18	1.01	1.18	1.00	1.09
Margin	EDP	0.1032	0.3937	0.1074	0.1057	
query	Gain	1.10	1.01	1.06	0.98	1.03
Double	EDP	0.1982	0.4283	0.2499	0.1565	
centroid	Gain	2.11	1.09	2.46	1.45	**1.78**

Table 4: Query phase error detection precision and gain after 200 instances.

vided only two thirds of the performance on the passive learning task compared to those produced from the Reuters corpus. While pretrained word embeddings could have been used, we wanted to avoid the issue of unknown tokens when dealing with specialized corpora.

In order to lower even more the effort of annotators, the next logical step would be to validate if the final prediction model was either as good or better at predicting any type of instances on the remaining corpus when compared to a standard, non-error detecting active learning process. This would entail that an error detection model could simply be used to annotate the remaining instances, correcting more errors and minimizing the creation of additional noise.

8 Conclusion and Future Work

We experimented with active learning methods on noisy corpora to lower the noise level, thus improving the overall quality of the datasets. Our proposed seeding method targeting error detection provided a gain factor of 2.73 when compared to the most used random seeding in active learning, while our double centroid method provided a 61.82% increase for finding noisy annotations when compared to baseline approaches used for active learning.

While there is room for improvement, these results show the potential application of active learning for corpus annotation. The applicability on two NLP tasks on different units (words or documents) shows a good adaptability of the tested methods, as the use of two languages in the corpora to avoid relying on language-specific techniques.

One untested hypothesis is if these methods provide the same level of performance when applied to corpora with human-generated noise instead of tool-generated noise. We expect that noise level might be lower in published and broadly used datasets if reannotated with the original protocol. This might influence the effectiveness of the tested methods.

These approaches should be tested on a broader set of natural language processing tasks, such as sentiment analysis, information quality, relevance identification, named entity classification, etc. This would either help to further advance the demonstration about the effectiveness of these methods, or to develop new approaches to facilitate the task of reannotation.

Some influential aspects of active learning have been left aside in this experiment as they did not directly implicate human annotators, like the cognitive charge of annotation correction for a human agent. This requires not only to assess the context of the existing annotation to deduce the correct annotation, but also to ponder, when the existing annotations differ, if they are not adding more noise.

Acknowledgments

This work has been supported by the Ministère de l'Économie et de l'Innovation du Québec (MEI). We would like to thank the anonymous reviewers for their relevant comments.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Leo Breiman. 2001. Random forests. In *Machine Learning*. pages 5–32.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *ACM SIGMOD 2000 Int. Conf. On Management of Data*. page 12.

Marie Candito and Djamé Seddah. 2012. Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical (the sequoia corpus : Syntactic annotation and use for a parser lexical domain adaptation method) [in french]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN*. ATALA/AFCP, pages 321–334. http://aclweb.org/anthology/F12-2024.

Wen-Chi Chou, Richard Tzong-Han Tsai, Ying-Shan Su, Wei Ku, Ting-Yi Sung, and Wen-Lian Hsu. 2006. A semi-automatic method for annotating a biomedical proposition bank. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*. Association for Computational Linguistics, Sydney, Australia, pages 5–12. https://www.aclweb.org/anthology/W06-0602.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *J. Artif. Int. Res.* 4(1):129–145.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. Experiments with crowdsourced re-annotation of a POS tagging data set. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. pages 377–382. http://aclweb.org/anthology/P/P14/P14-2062.pdf.

Rong Hu, Brian Mac Namee, and Sarah Jane Delany. 2010. Off to a good start: Using clustering to select the initial training set in active learning. In *FLAIRS Conference*.

Kowsari, Kamran, Brown, Donald, Heidarysafa, Mojtaba, Jafari Meimandi, Kiana, Gerber, Matthew, and Barnes, Laura. 2018. Web of Science Dataset. https://doi.org/10.17632/9rw3vkcfy4.6.

David Lewis. 2004. Reuters 21578 data set version 1.0 http://www.daviddlewis.com/resources/ testcollections/reuters21578/readme.txt.

Christopher H Lin, M Mausam, and Daniel S. Weld. 2016. Re-Active Learning: Active Learning with Relabeling. In *Thirtieth AAAI Conference on Artificial Intelligence*. page 8.

F. T. Liu, K. M. Ting, and Z. Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. pages 413–422. https://doi.org/10.1109/ICDM.2008.17.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 92–97. http://aclweb.org/anthology/P13-2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

M. F. Porter. 1997. Readings in information retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, chapter An Algorithm for Suffix Stripping, pages 313–316. http://dl.acm.org/citation.cfm?id=275537.275705.

Dietrich Rebholz-Schuhmann, Antonio José Jimeno-Yepes, Erik M. van Mulligen, Ning Kang, Jan Kors, David Milward, Peter Corbett, Ekaterina Buyko, Katrin Tomanek, Elena Beisswanger, and Udo Hahn. 2010. The CALBC silver standard corpus for biomedical named entities — a study in harmonizing the contributions from four independent named entity taggers. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA), Valletta, Malta. http://www.lrec-conf.org/proceedings/lrec2010/pdf/888_Paper.pdf.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1160–1170. https://doi.org/10.18653/v1/P17-1107.

Peter J. Rousseeuw and Katrien Van Driessen. 1999. A fast algorithm for the minimum covariance determinant estimator. *Technometrics* 41(3):212–223. https://doi.org/10.1080/00401706.1999.10485670.

Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval .

Helmut Schmid. 1997. Probabilistic part-of-speech tagging using decision trees. In Daniel Jones and

Harold Somers, editors, *New Methods in Language Processing*, UCL Press, London, GB, Studies in Computational Linguistics, pages 154–164.

Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.* 13(7):1443–1471. https://doi.org/10.1162/089976601750264965.

Maria Skeppstedt. 2013. Annotating named entities in clinical text by combining pre-annotation and active learning. In *Proceedings of the ACL Student Research Workshop*. page 74–80.

NLP Community Perspectives on Replicability

Margot Mieskes
University of Applied Sciences, Darmstadt
Germany
margot.mieskes@h-da.de

Karën Fort
Sorbonne Université, EA STIH
Paris, France
karen.fort@sorbonne-universite.fr

Aurélie Névéol
LIMSI, CNRS,
Université Paris-Saclay
France
neveol@limsi.fr

Cyril Grouin
LIMSI, CNRS,
Université Paris-Saclay
France
cyril.grouin@limsi.fr

Kevin Cohen
Computational Bioscience Program
University of Colorado, USA
kevin.cohen@gmail.com

Abstract

With recent efforts in drawing attention to the task of replicating and/or reproducing[1] results, for example in the context of COLING 2018 and various LREC workshops, the question arises how the NLP community views the topic of replicability in general. Using a survey, in which we involve members of the NLP community, we investigate how our community perceives this topic, its relevance and options for improvement. Based on over two hundred participants, the survey results confirm earlier observations, that successful reproducibility requires more than having access to code and data. Additionally, the results show that the topic has to be tackled from the authors', reviewers' and community's side.

1 Introduction

"As a community, we need to know where our approaches fail, as much – if not more so – as where they succeed." Despite this statement by Fokkens et al. (2013), we are still aiming at higher, faster, better results with little outside verification. And although it has become good practise to share code, data and parameters, previous work and experience indicate that sharing is still not as common as one would hope for. Call for Papers in major conferences encourage to submit or reference data and to submit code, i.e., in supplemental material[2]. But recent work indicates that this is not done thoroughly enough (Mieskes, 2017; Cohen et al., 2017; Wieling et al., 2018). Other factors mentioned by Fokkens et al. (2013), such as preprocessing methods, experimental setup, system variation, etc., are rarely reported. Pedersen (2008) urges to not fear any dispossession of a tool, and highlights that sharing code will result in its use in new systems and citations of the work describing this code. Moreover, we should consider sharing software as a way to improve it more efficiently. In recent years, we have seen a rise in attention targeted towards the task of replication for example in the COLING 2018 selection criteria[3], the LREC 4REAL Workshops (Branco et al., 2016, 2018) and the recent LREC initiative for replication.[4] But so far the task of replicating previous results has little merit in itself, but is rather only a (baseline) part in a paper. Additionally, normally it only gets reported in successful or mainly successful cases. Following the argument by Fokkens et al. (2013), the cases where it fails, hardly ever get reported, despite Calls for Papers encouraging negative results[5] and although they might be equally or even more important than the successful replication.

Our contribution therefore is to identify how the community views the topic of replication and what role each individual plays as an author, as a reviewer and as part of the NLP community. In conducting a survey, which drew answers from over two hundred respondents, we get a better picture of the factors that support or hinder making repli-

[1] We use *replication* to describe related efforts, regardless of the exact aim (see (Cohen et al., 2018))

[2] see for example http://www.acl2019.org/EN/call-for-papers.xhtml "ACL (...) encourages the submission of supplementary material to report (...) details necessary for the replication of the experiments."

[3] See: https://coling2018.org/paper-types/.

[4] This call occurred 5 weeks after we posted our call and are unrelated, but the latter might be inspired by our survey, see http://wordpress.let.vupr.nl/lrec-reproduction/.

[5] "A negative result" http://www.acl2019.org/EN/call-for-papers.xhtm (Short Papers)

cation more visible and how the three factors described above influence this. Our results indicate that the participants in general regard replication as an important issue and that the NLP community could do more to support replication, which would strengthen the field as a whole.[6]

2 Related Work

One of the earliest reports of a replication effort addresses manual word-sense disambiguation based on four words, representing different degrees of difficulty (Kilgarriff, 1999). The author reports that humans agree in this task on average in 95% of the cases. Following Fokkens et al. (2013), others look into parameters that influence the replicability of results. Dakota and Kübler (2017); Marrese-Taylor and Matsuo (2017) and Horsmann and Zesch (2017) report various parameters and problems with replication experiments for morphology and syntax.

In the field of biomedical NLP, Olorisade et al. (2017) assess the reproducibility of findings published in 33 papers. They notice that data sets were missing, making it impossible to reproduce results for 80% of the papers. These figures are in line with results reported by Mieskes (2017). They consider that a permanent link to the resources (data set, software, etc.) must exist along with published papers. As part of a NLP challenge, Névéol et al. (2016) report results on replicating experiments from three systems submitted to the CLEF eHealth track. They show that replication is feasible although "ease of replicating results varied". They suggest the allowance of extra pages for papers, where information required to replicate an experiment could be reported.

Moore and Rayson (2018) illustrate how to publish relevant details to reduce efforts in repeatability and generalisability. Suggestions include using only open data, open source code and providing extensive documentation in the code.

Wieling et al. (2018) describe one example where exact replication was possible and the authors list the parameters that allowed them to do so: a virtual image, containing all code and all data or providing CodaLab worksheets. Their study, which compared the situation in sharing research artefacts between 2011 and 2016 indicates that, while the situation has improved and the availability of data is high, access to code is less so and requesting code is unsuccessful in most of the cases. Based on results of their actual replication experiments, *at most 60%* of the studies are replicable, but only if the need for exact replication was relaxed.

Fares et al. (2017) present a repository and infrastructure containing texts, tools and embeddings for English and Norwegian. Their aim is to facilitate replicability and testing of previous results. Dror et al. (2017) propose a "replicability analysis framework" and demonstrate its use on various tasks such as part-of-speech tagging or cross-domain sentiment classification. They specifically target cases where algorithms are compared across multiple data sets. The results indicate that testing on a range of data sets is only beneficial if the data sets are heterogeneous.

3 Survey Design

Our survey has 18 questions, of which many were conditional and show only if they apply to the respondent. Thus, not all questions have been answered by all participants, while most multiple-choice questions allow for several answers, resulting in more answers than participants for these questions. Questions are grouped into three categories: (i) replication work in general, (ii) replicating one's own work and (iii) replicating others' work.

General questions quiz participants on their perception of replication work. We also inquire about their current position to investigate potential correlation with other aspects of the survey. Questions addressing participants' replication experience specifically enquired about research artefacts availability (data, code, parameters, etc.) and about the timeline of the replication experience in order to assess attrition.

The survey was advertised on professional mailing lists (BioNLP, Corpora, LN and GLCL[7]) and social network (LinkedIn). The appendix gives details on the progression of responses. With respect to sensitive data, only e-mail addresses were provided, on a voluntary basis, and we follow the ACM Code of Ethics and Professional Conduct[8],

[6] The complete results of the survey are available at https://github.com/replicateNLP/Survey-RANLP2019. Please note, that due to privacy regulations, we had to remove some free text answers that contain personal information, such as E-Mail addresses, which were given on a voluntary basis.

[7] Biomedical NLP, French and German NLP.

[8] https://www.acm.org/code-of-ethics

specifically sections 1.6 and 1.7.

Figure 1 illustrates the flow of the questionnaire. If a person did neither replicate their own or someone elses work, they only had to answer the blue marked questions. If a participant tried to replicate his/her own work, but not someone elses work, they only had to answer the blue and the orange questions. Only persons who have experience in replicating their own and someone elses work had to go through the whole questionnaire, including the green questions.[9] This flow in addition to the possibility to give more than one answer in some questions results in different numbers of answers for each question.

4 Results

We received 225 responses and the two biggest groups of participants in our study identify themselves as graduate students and postdocs.

With respect to when work on replication has been done, 36 participants (16%) gave more than one anwer, indicating that they did work on replication at various stages of their career. Most answers (50.3%) state that replication was done on MSc or PhD level, less on PostDoc level (20.7%) and slightly more as Faculty members (24.3%). However, we did not find strong correlations between the respondents' position and opinion on the importance (or lack thereof) of reproducibility. Figure 2 shows the absolute numbers for this question, while Figure 3 shows the numbers for the participants current position.

4.1 General Stance towards Replicability

The answers show that in 56.4% of the cases, work on replication is considered "Important" and another 7.5% state that it is "Somewhat Important". Only 2 answers indicate that this work is "Unimportant". 20% of the answers regard work on replication as publishable, while 11.8% deem it unpublishable. 87 participants gave more than one answer. The majority (49) consider work on replication as important *and* publishable, while 26 of them consider it important *but not* publishable (see Figure 4 for the absolute numbers).

4.2 Replicating one's Own Work

Roughly 70% (156) of the participants declare they have tried to replicate their own work while about 30% have not tried (total 225).

With respect to how often replication of the same experiment was tried, 15 participants gave more than one answer, giving us 172 answers (see Figure 5 for the detailed figures). Of these, 28.5% indicate that replication was tried only once, while 38.4% tried 3 times or more.

When looking at the last attempt (total answers 234), nearly half (47.0%) report that they reached the same general conclusions, while 23.1% state that they reached the same figures. A little less than 10% report that they managed to reimplement the system, but got significantly different results. Another 14.9% could not find either the code or the data or the parameters used for the experiment (see Figure fig:resultsOwn for the absolute numbers). 52 participants gave more than one answer of which 25 report that they reached the same general conclusions *and* the same figures. Overall, the results indicate that even in the case when researchers try to replicate their own work, they fully succeed in only 23.1% of the cases.

4.3 Replicating Others' Work

About 60% (total 130) of the participants report that they tried to replicate someone else's work (see Figure 7 for detailed, absolute numbers). 51 respondents gave more than one answer with respect to the results achieved when replicating somebody else's work, resulting in 211 answers. Approximately 40% of the answers state that they reached the same general conclusions or figures, while another 33.6% of the answers state that they managed to re-implement or re-run the system, but with significantly different results. Nearly 23.1% of the responses state that re-implementation or re-runnning of experiments was not achieved (see Figure 8 for the absolute responses for this question). This means that over half of the replication experiments failed, either early on or at the level of results achieved.

4.4 Accessibililty of Research Artefacts

For finding research artefacts such as code, data and parameters, respondents gave several answers, resulting in 250 answers for where the code can be found, 260 answers for finding data and resources and 233 answers for finding the experimental parameters. GitHub is by far the most popular (36.4% of the answers) for accessing *code*, but more than 23.6% of the answers state that code is found on the authors' personal webpage,

[9] Please note that only the most important questions are illustrated here and some questions have been left out.

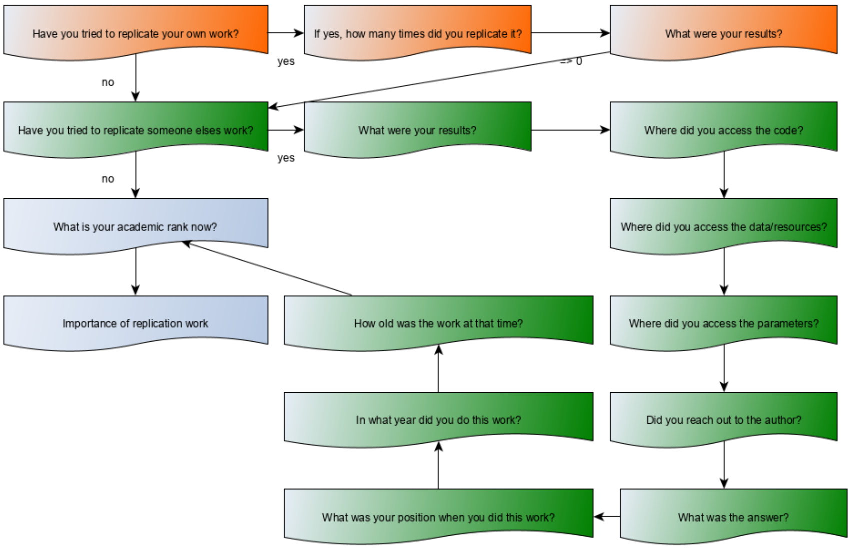

Have you tried to replicate your own work?
If yes, how many times did you replicate it?
What were your results?
yes
=> 0
no
Have you tried to replicate someone elses work?
What were your results?
Where did you access the code?
yes
no
What is your academic rank now?
Where did you access the data/resources?
Importance of replication work
How old was the work at that time?
Where did you access the parameters?
In what year did you do this work?
Did you reach out to the author?
What was your position when you did this work?
What was the answer?

Figure 1: Illustration of the questionnaire flow.

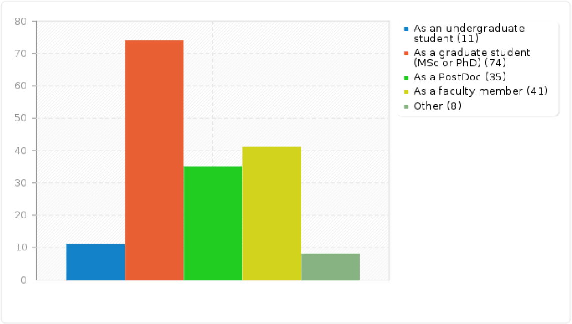


Figure 2: Position of the participants at the time they were doing replication experiments.

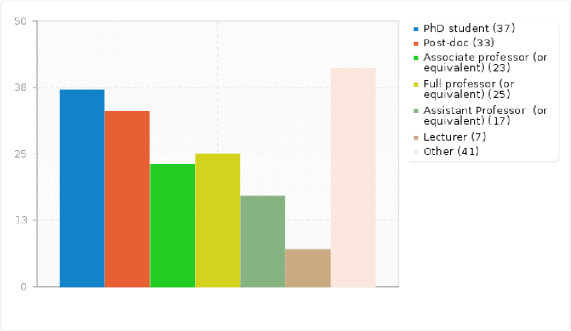


Figure 3: Position of the participants at the time of the survey.

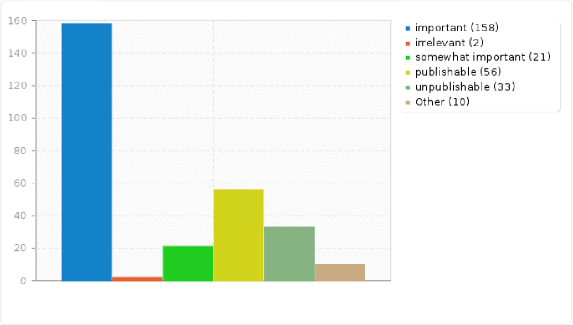


Figure 4: Importance of Replication in General.

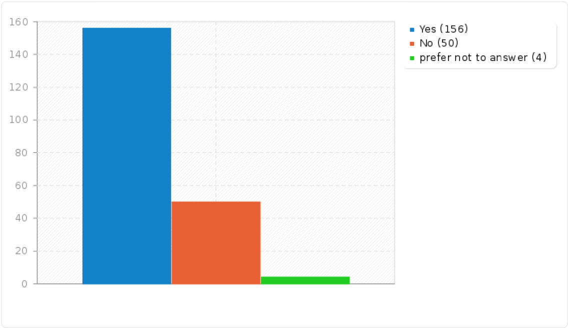


Figure 5: Participants who tried to replicate their own work.

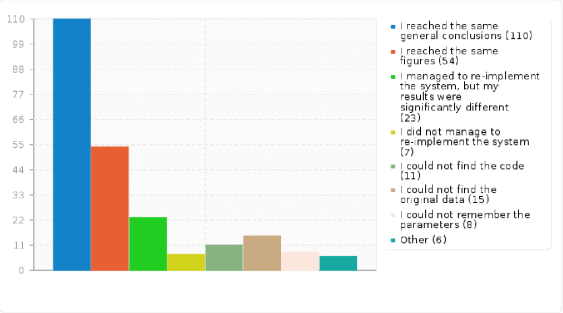


Figure 6: Results achieved when trying to replicate own work.

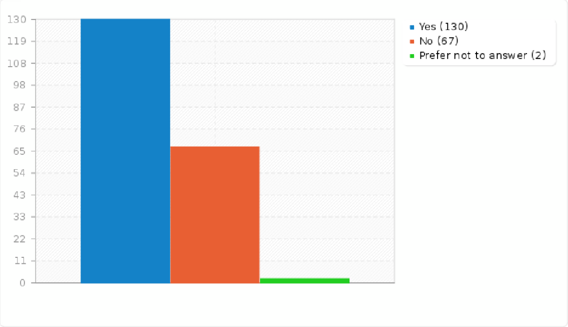


Figure 7: Participants who tried to replicate someone elses work.

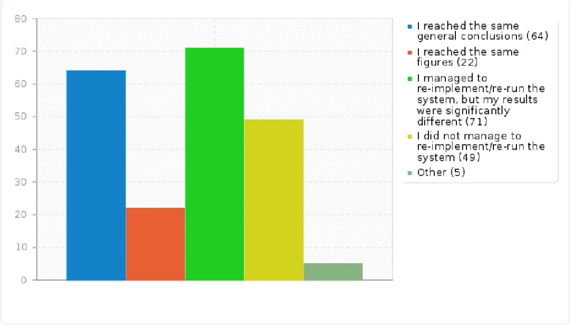


Figure 8: Results achieved when replicating someone elses work.

which does not guarantee availability beyond that person maintaining his/her webpage. More than 14% of the answers report that the code could not be found. *Data* is also primarily published via a GitHub or personal webpages (25% and 25.7% of the answers respectively). 11.1% report that the material used for the experiments could not be found. *Parameters* for experiments are primarily found in the respective publications (40.3% of the answers), while 21.9% of the answers state they could be found on GitHub as well. 13.7% report that they could not find parameters at all. Figure 9 illustrates the absolute numbers concerning the availability of code, while Figure 10 and Figure 11 illustrate them for other resources and parameters respectively.

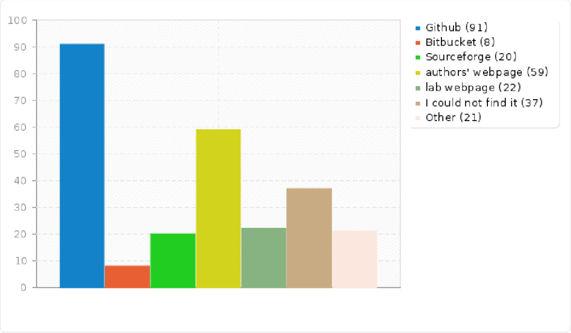


Figure 9: Sources for Accessing the Code for replication.

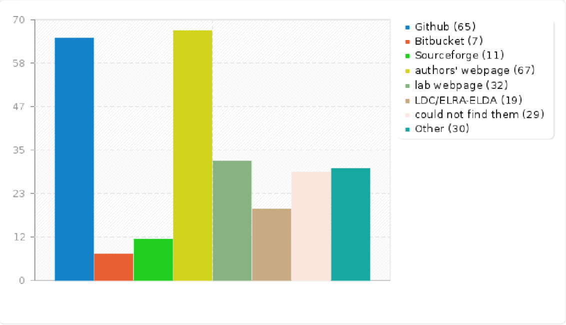


Figure 10: Sources for Accessing Data and other Resources.

Concerning these three elements, the text box associated with “Other” very frequently mentions “personal communication” or “e-mail” as a way of obtaining necessary information. This gives rise to a range of further issues, legal, ethical and in terms of transparency. Besides, it is only possible if authors actually answer such e-mails.

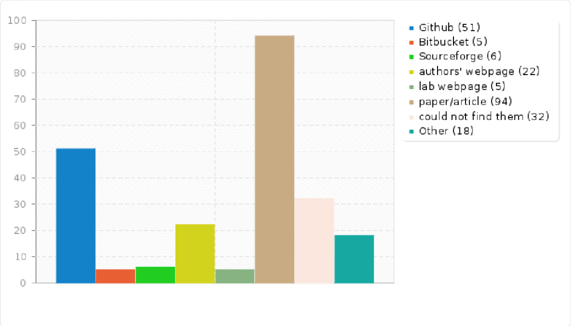


Figure 11: Sources for Accessing Parameters for Replication.

In our survey, 40% of the participants report that they tried to get in touch with the authors (see Figure 12), but only in about 30% of the reported cases received a helpful answer (164 answers received, as 41 participants gave more than one answer). Approximately 20.1% mention that unhelpful answers were received and almost 23.8% never received any answer. Due to evolution of careers, 13.4% found out that the person had left the lab or the e-mail bounced, resulting in almost 40% of examples where authors were unreachable (see Figure 13).

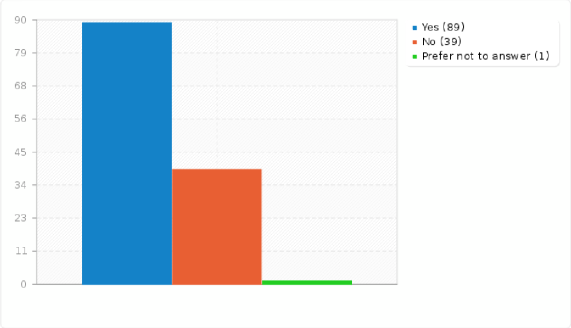


Figure 12: Participants who state that they reached out to original authors.

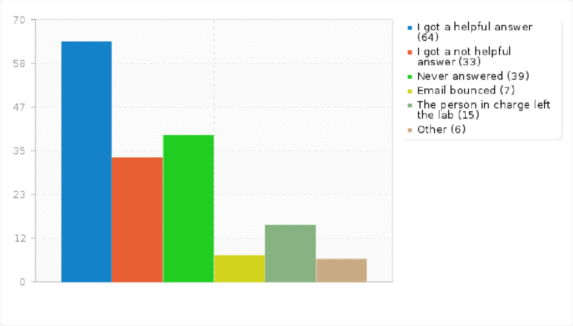


Figure 13: Quality of Answers by original authors.

5 Discussion

The survey results suggest that **replicability is perceived as an important issue** by a majority of responses ($\geq 60\%$). It is difficult to compute an accurate response rate for the survey, because we do not know the extent of the population comprised by subscribers to the mailing lists and professional networks that we reached out to. However, if we approximate the target population using the average participation in a *ACL conference (N=1,000), we can estimate the response rate at about 20%. This has previously been described as an "acceptable" response rate for online surveys.[10] While the responding population includes researchers with a wide range of seniority as well as members of academia and industry, it could be biased by their interest in replication. The results, and especially the comments, indicate that this is most likely not the case as some participants do not see the value in replicating previous results. They state that replicating previous work is only an "exercise" and actually an "overload" on the already busy researchers' schedule. One comment states that *"10 year old systems are irrelevant"* and *"ML is moving so fast"*, which renders replicability studies essentially worthless. This could explain why the two largest groups performing replicability studies are on PhD or PostDoc level.

What does this mean for replicability? Responses indicate a variety of views on how replicability should be facilitated. One comment states that there is already a culture of sharing and publishing data and code and this should be enough. Another participant states that in industrial research publishing data or code is difficult, but suggests that the validity of an approach can be proven by applying the method to other data and in reaching the same general conclusions. Some participants support the idea of giving more visibility to replicability by making it a prominent topic at major conferences and by enforcing reporting guidelines towards reproducibility.

Some participants mention that replicability is crucial to be taken seriously as a scientific field, even stating that the field is "suspect" if replication fails. One participant suggests that *"every paper*

[10] http://socialnorms.org/what-is-an-acceptable-survey-response-rate/

cited enough times should be replicated".

6 Conclusions and Future Work

Based on a survey we gave insight into the NLP community's view on replicability. We targeted three different facets of this topic: Authors publishing their work, Researchers building on top of other researchers' work and the Community, supporting such efforts. Our results show that on the authors' side more information has to be shared openly, rather than via personal communication. The use of reporting guidelines formalized into a protocol has been suggested recently for clinical NLP (Velupillai et al., 2018). Earlier studies from the clinical domain suggest that adherence to such guidelines is suboptimal (Samaan et al., 2013) and methods to improve adherence are being investigated (Blanco et al., 2017). The task of creating guidelines falls to the community and the adherence of such guidelines could become part of the reviewing process.

Experiments in replication fail more often than not. If we document and store all relevant information so that results could be reproduced by ourselves (e.g., before the final paper submission), the package could be published completely. Results by Wieling et al. (2018) indicate that images containing all the material or technical lab books published on CodaLab might be a way to proceed. Additionally, failure to replicate previous work (i.e. not achieving the same results and/or not being able to draw the same conclusions as previously reported), should be publishable in a way that gives us scientific merit and could be encouraged more.

Based on the comments, each of us, in all of our individual roles can improve the situation: As authors, we can be more diligent when reporting our experiments and experimental setup—even testing the replicability of our experiments ourselves. As reviewers, we can be more careful to check the supplementary material for relevant information, pointing out missing elements. As a community, we can appreciate replication more and develop guidelines both for authors and for reviewers.

Future Work The next steps include, but are not limited to, analyzing whether the supplementary material and appendices actually do improve replicability, as stated by Névéol et al. (2016). Furthermore, evaluating the repository offered by Fares et al. (2017), whether other researchers actually build on top of it and with what results. NAACL recently initiated a "test of time" award. This could be extended to consider experimental work that has been cited often and gained influence on a shorter time-scale. This work could be verified for a follow-up conference. Replicability could also become a factor in the best paper awards.

References

David Blanco, Jamie J Kirkham, Douglas G Altman, David Moher, Isabelle Boutron, and Erik Cobo. 2017. Interventions to improve adherence to reporting guidelines in health research: a scoping review protocol. *BMJ Open*, 7(11).

António Branco, Nicoletta Calzolari, and Khalid Choukri, editors. 2016. *Proceedings of the Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*.

António Branco, Nicoletta Calzolary, and Khalid Choukri, editors. 2018. *Proceedings of the 4REAL 2018 - Workshop on Replicability and Reproducibility of Research Results in Science and Technology of Language*. European Language Resources Association, Paris.

K. Bretonnel Cohen, Aurélie Névéol, Jingbo Xia, Negacy Hailu, Larry Hunter, and Pierre Zweigenbaum. 2017. Reproducibility in Biomedical Natural Language Processing. In *AMIA annual symposium proceedings*, page 1994. American Medical Informatics Association.

K. Bretonnel Cohen, Jingbo Xia, Pierre Zweigenbaum, Tiffany Callahan, Orin Hargraves, Foster Goss, Nancy Ide, Aurélie Névéol, Cyril Grouin, and Lawrence E. Hunter. 2018. Three Dimensions of Reproducibility in Natural Language Processing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* Miyazaki, Japan, May 7–12, 2018. European Language Resources Association (ELRA).

Daniel Dakota and Sandra Kübler. 2017. Towards Replicability in Parsing. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017* Warna, Bulgaria, 2–8 September 2017, pages 185–194. INCOMA Ltd.

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability Analysis for Natural Language Processing: Testing Significance with Multiple Datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486.

Murhaf Fares, Andrey Kutuzov, Stephan Oepen, and Erik Velldal. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic*

Conference on Computational Linguistics Gothenburg, Sweden, 22–24 May, 2017, pages 271–276. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from Reproduction Problems: What Replication Failure Teaches Us. In *Proceedings of the 51st Conference of the Association for Computational Linguistics* Sofia, Bulgaria 4–9 August 2013, pages 1691–1701.

Tobias Horsmann and Torsten Zesch. 2017. Do LSTMs really work so well for PoS tagging? – A replication study. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* Copenhagen, Denmark, September 9–11, 2017, pages 727–736. Association for Computational Linguistics.

Adam Kilgarriff. 1999. 95% Replicability for Manual Word Sense Tagging. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL-1999)* Bergen, Norway, 8–12 June, 1999. Association for Computational Linguistics.

Edison Marrese-Taylor and Yutaka Matsuo. 2017. Replication issues in syntax-based aspect extraction for opinion mining. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics* Valencia, Spain, April 3–7, 2017, pages 23–32. Association for Computational Linguistics.

Margot Mieskes. 2017. A Quantitative Study of Data in the NLP community. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 1–7, Valencia, Spain. Association for Computational Linguistics.

Andrew Moore and Paul Rayson. 2018. Bringing replication and reproduction together with generalisability in NLP: Three reproduction studies for Target Dependent Sentiment Analysis. In *Proceedings of the 27th International Conference on Computational Linguistics* Santa Fe, New Mexico, USA, August 20–26, 2018, pages 1132–1144. Association for Computational Linguistics.

Aurélie Névéol, K. Bretonnel Cohen, Cyril Grouin, and Aude Robert. 2016. Replicability of Research in Biomedical Natural Language Processing: a pilot evaluation for a coding task. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis* Austin, Texas, November 5, 2016, pages 78–84. Association for Computational Linguistics.

Babatunde Kazeem Olorisade, Pearl Brereton, and Peter Andras. 2017. Reproducibility of studies on text mining for citation screening in systematic reviews: Evaluation and checklist. *Journal of Biomedical Informatics*, 73:1 – 13.

Ted Pedersen. 2008. Empiricism Is Not a Matter of Faith. *Computational Linguistics*, 34(3):465–470.

Zainab Samaan, Lawrence Mbuagbaw, Daisy Kosa, Victoria Borg Debono, Rejane Dillenburg, Shiyuan Zhang, Vincent Fruci, Brittany Dennis, Monica Bawor, and Lehana Thabane. 2013. A systematic scoping review of adherence to reporting guidelines in health care literature. *Journal of Multidisciplinary Healthcare*, 6:169–88.

Sumithra Velupillai, Hanna Suominen, Maria Liakata, Angus Roberts, Anoop D. Shah, Katherine Morley, David Osborn, Joseph Hayes, Robert Stewart, Johnny Downs, Wendy Chapman, and Rina Dutta. 2018. Using clinical Natural Language Processing for health outcomes research: Overview and actionable suggestions for future advances. *Journal of Biomedical Informatics*, 88:11 – 19.

Martijn Wieling, Josine Rawee, and Gertjan van Noord. 2018. Squib: Reproducibility in Computational Linguistics: Are We Willing to Share? *Computational Linguistics*, 44(4):641–649.

Unsupervised Data Augmentation for Less-Resourced Languages with no Standardized Spelling

Alice Millour and **Karën Fort**
Sorbonne Université / STIH - EA 4509
28, rue Serpente, 75006 Paris, France
alice.millour@etu.sorbonne-universite.fr,
karen.fort@sorbonne-universite.fr

Abstract

Non-standardized languages are a challenge to the construction of representative linguistic resources and to the development of efficient natural language processing tools: when spelling is not determined by a consensual norm, a multiplicity of alternative written forms can be encountered for a given word, inducing a large proportion of out-of-vocabulary words.

To embrace this diversity, we propose a methodology based on crowdsourcing alternative spellings from which variation rules are automatically extracted. The rules are further used to match out-of-vocabulary words with one of their spelling variants. This virtuous process enables the unsupervised augmentation of multi-variant lexicons without requiring manual rule definition by experts. We apply this multilingual methodology on Alsatian, a French regional language and provide (i) an intrinsic evaluation of the correctness of the obtained variants pairs, (ii) an extrinsic evaluation on a downstream task: part-of-speech tagging.

We show that in a low-resource scenario, collecting spelling variants for only 145 words can lead to (i) the generation of 876 additional variant pairs, (ii) a diminution of out-of-vocabulary words improving the tagging performance by 1 to 4%.

1 Natural Language Processing and Non-Standardized Languages

Non-standardized languages present a great productivity of spelling variants for a given word. The absence of standardized spelling points up the geographical and demographic variations that might exist and are otherwise smoothed down. This variability results in the coexistence of alternative written forms, hence in a large proportion of out-of-vocabulary words in the context of supervised machine learning.

In what follows, we first present our approach to generate spelling variant pairs based on an initial set of crowdsourced spelling variant pairs. This method is language independent and relies on resources that do not require expert knowledge, hence can easily be crowdsourced.

Second, we exemplify the use of such a method to reduce the proportion of unknown words that undermines supervised algorithms in the context of non-standardized languages.

1.1 Working with Multi-Variant Linguistic Resources

The question of variation in non-standardized languages naturally arises starting when one begins the process of corpus building (or collection). When dialectal and spelling variants overlap, inter- and intra-dialectal variations can be hard, not to say impossible, to untangle. In the following, we will design as "spelling variant" any variant due to either dialectal variation, spelling convention variation, or an accumulation of both.

Although one might chose to work on corpora produced in a controlled environment, in which the spelling conventions and writers are carefully chosen, this setup is unlikely to produce satisfying results on real-life data.

Producing linguistic resources, be it lexica, raw or annotated corpora, represents a cost that cannot be afforded for languages missing resources in the broad sense, including funding and experts. Crowdsourcing has proven to be a viable option to produce quality resources at a reduced cost (Chamberlain et al., 2013). Applying crowdsourcing to less-resourced non-standardized lan-

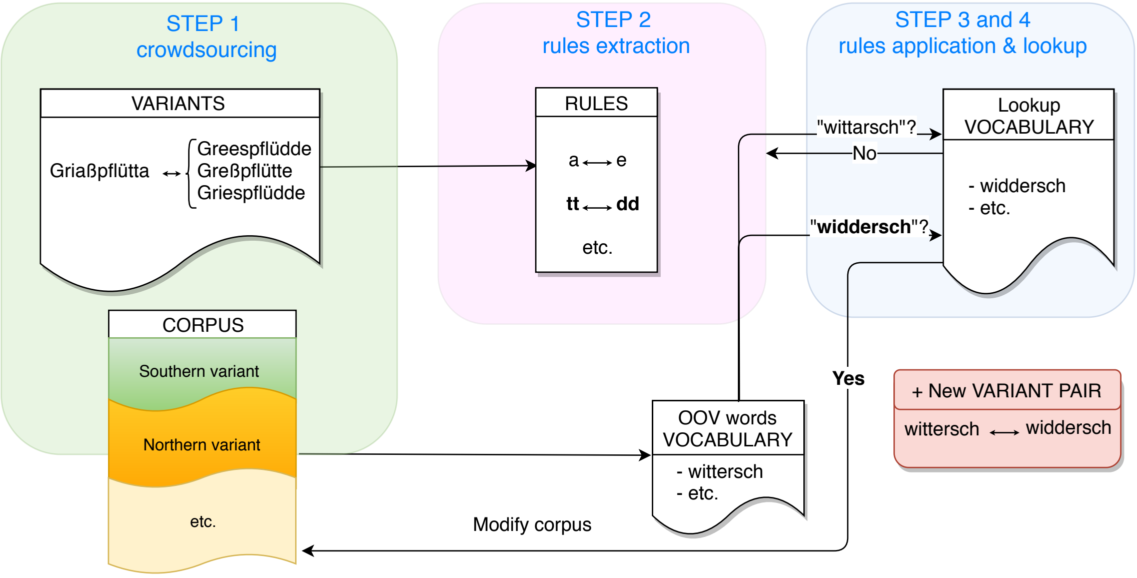


Figure 1: Data augmentation process.

guages presents additional difficulties such as accessibility to the speakers, or representativity of contents (Millour and Fort, 2018b).

Yet, when a community of speakers can be found on-line, it seems necessary to empower them to produce raw corpora and to document variability. In fact, the speakers appear to be, collectively, the only experts of the mechanisms at stake.

To meet this goal, we developed a crowdsourcing platform that collects two types of resources: (1) raw texts and (2) spelling variants on these texts. These resources are used to seed the unsupervised augmentation of the multi-variant lexicon following a process that we detail in Figure 1.

1.2 Process Overview

Given an existing linguistic resource (corpus, lexicon, or both) R_{Lookup} and a set of out-of-vocabulary words Voc_{OOV}, the process consists of four steps:

1. crowdsourcing spelling variant pairs,
2. automatic rules extraction,
3. application of the rules on elements of Voc_{OOV},
4. lookup of the resulting transformed spelling in R_{Lookup}.

These steps are detailed in sections 2 and 3 and illustrated with their application on Alsatian.

In the context of OOV words reduction in a given corpus, step 4 is followed by a transposition of those for which a variant has been identified in R_{Lookup}. Especially, in the context of supervised machine learning, one cannot expect to find all existing variants in a training corpus. By replacing an OOV word by one of its already known spelling variants, we make the most of the annotations we have at our disposal (see Section 4).

1.3 The Case of Alsatian

Alsatian is a French regional language counting 550,000 speakers in 2004 (Barre and Vanderschelden, 2004). This continuum of Alemannic dialects is an example of language in which the dialectal variants are not erased in the written form by any spelling system.

Initiatives such as the Orthal guidelines (Crévenat-Werner and Zeidler, 2008) have been developed to unify the Alsatian spelling while being respectful of its variations. Yet, these keep the variability (*Kìrisch* and *Kìch* are the Orthal version of *Kerisch* and *Kîch*, Northern and Southern possible versions for the word "church"), and are still unknown by a majority of Alsatian Internet users as shown by a recent survey (Millour, 2019).

For this reason, the dialectal variations (6 to 8 variants emerge from the continuum) combine with the variety of spelling habits, which might depend, for instance, on the linguistic backgrounds of the speakers.

Also, since there exist an active community of on-line speakers, Alsatian is a good candidate for crowdsourcing experiments.

2 Crowdsourcing Spelling Variants

We developed a slightly gamified crowdsourcing platform, Recettes de Grammaire[1] which allows us to collect (i) raw corpora in the shape of cooking recipes, (ii) part-of-speech annotations on the recipes, and (iii) alternative spellings. The platform is language independent and its source code is freely available on GitHub[2] under the CeCILL v2.1 license.[3]

We do not differentiate variants due to a variation in dialects, in spelling or in an accumulation of these two factors during collection.

The addition of a new spelling variant can be performed: (i) by adding a variant to any word that is present on the platform by clicking on a word cloud on the main page (see Figure 2), (ii) by dynamically editing the written contents on the website thanks to a feature called "Personally, I would have said it like that!", illustrated on Figure 3.

These features enable the participants to modify the content they read and further annotate in a manner that suits their writing habits. In fact, feedback we received on previous experiments led on crowdsourcing part-of-speech annotations for Alsatian (Millour and Fort, 2018a) highlighted the fact that some participants felt unrepresented by the texts on the platform, and that annotating dialectal or spelling variants they are not familiar with was an obstacle hard to overcome.

The interface allows the participants to provide an alternative spelling for either a single word or a sequence of words. Although the latter facilitates the task for the participants, it sometimes leads to alternative spellings which number of words did not match the original version, hence could not be immediately aligned. In such cases and when possible, the alternative spellings were manually aligned with the original version.

So far, the collected resource contains 367 variants provided by 10 participants for 145 words (with two to six variants per word), e.g. {*bìtsi*, *bessel*, *béssel*}, "a bit of".

The only information we possess about these participants is the languages they speak and their place of origin, when they fill it in in their profile. Based on the information provided by 8 of them, we can assume that 3 to 4 dialectal areas are covered by the towns of origin of the participants. No assumption can be made regarding their proficiency in Alsatian.

The size of this resource does not allow us to perform direct lookup for any out-of-vocabulary word me might encounter. However, we can use the aligned variants to identify substitution patterns, and extract sets of rules we apply to any OOV word as described in the following section.

3 Unsupervised Data Augmentation

3.1 Rules Extraction

In the manner of (Prokić et al., 2009), we used ALPHAMALIG[4], a multi sequence alignment tool, to perform the alignment of our variants necessary to the extraction of substitution patterns. The tool requires to be provided with an alphabet of symbols, weighted with the match, mismatch, insertion and deletion scores of given characters. Since we have no *a priori* knowledge of these scores, the only assumption we made is that vowels are more likely to match vowels than consonants and vice versa. Insertion and deletion are given the same scores for all characters. An example of the alignment obtained for four crowdsourced variants is given in table 1.

ˆ	G	A	L	-	R	**Ì**	**E**	**W**	L	**E**	K	**Ü**	**E**	C	H	**E**	$	(1)
ˆ	G	A	L	**E**	R	**I**	**E**	**B**	L	**E**	K	**Ü**	**E**	C	H	**A**	$	(2)
ˆ	G	A	L	**E**	R	-	**E**	**W**	L	**E**	K	**Ù**	-	C	H	**E**	$	(3)
ˆ	G	A	L	-	R	**Ì**	**A**	**W**	L	**A**	K	**Ü**	**A**	C	H	**A**	$	(4)

Table 1: Alignment of four variants of the Alsatian (compound) word for "carrot cake".

From the produced alignments we can identify substitution patterns of different degrees of rigidity, depending on the size of the context. We extract three sets of rules which either force the matching of the left (L), right (R) or both contexts (L+R).

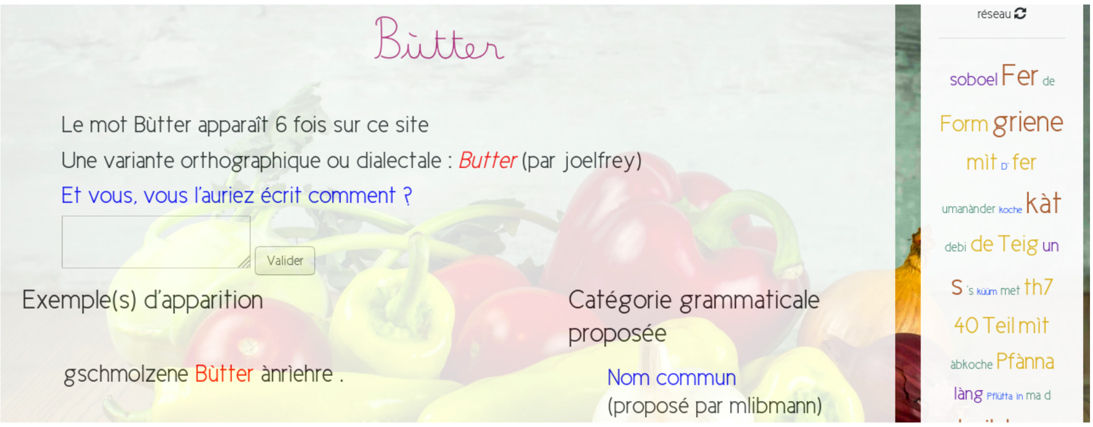
The ˆ and $ characters, respectively representing the beginning and the end of a word, are in-

[1] "Grammar's Recipes", see https://bisame.paris-sorbonne.fr/recettes.

[2] See https://github.com/alicemillour/Bisame/tree/recipes.

[3] See http://www.cecill.info/index.

[4] Source code: http://alggen.lsi.upc.es/recerca/align/alphamalig/intro-alphamalig.html.



Bùtter

Le mot Bùtter apparaît 6 fois sur ce site

Une variante orthographique ou dialectale : Butter (par joelfrey)

Et vous, vous l'auriez écrit comment ?

Exemple(s) d'apparition

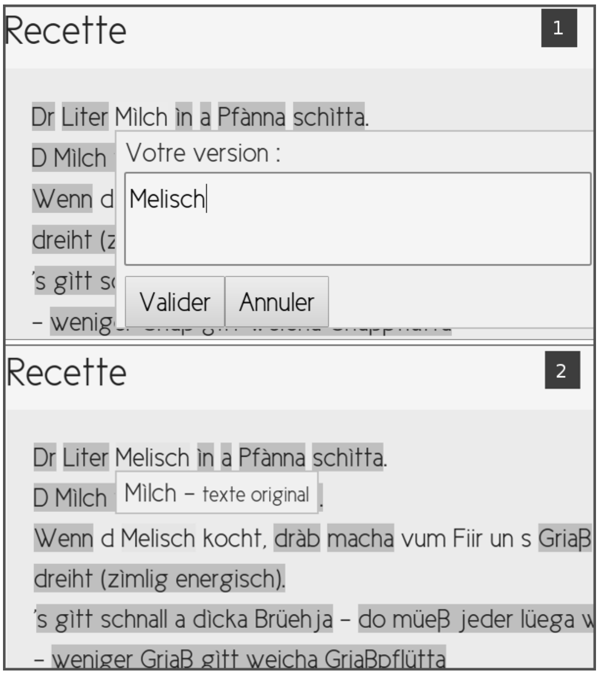
gschmolzene Bùtter ànrìehre .

Catégorie grammaticale proposée

Nom commun

(proposé par mlibmann)

Figure 2: Spelling addition using the wordcloud. The word is shown in its context, with the proposed part-of-speech (if available).



Recette

Dr Liter Milch in a Pfànna schìtta.

D Milch

Wenn d

dreiht (z

's gitt s

– wenig

Votre version :

Melisch

Recette

Dr Liter Melisch ìn a Pfànna schìtta.

D Milch Mìlch – texte original .

Wenn d Melisch kocht, dràb macha vum Fiir un s Griaß

dreiht (zimlig energisch).

's gìtt schnall a dìcka Brüehja – do müeß jeder lüega w

– weniger Griaß gìtt weicha Griaßpflütta

Figure 3: Spelling addition (1) and visualization (2) (highlighted words present at least one additional variant).

terpreted as elements of context. The rules are extracted from each pair of the combination of the aligned variants. From the aligned variants (1) and (2) showcased in Figure 1, four L+R rules are extracted: LR ↔ LER ; RÌE ↔ RIE ; EWL ↔ EBL ; HE$ ↔ HA$. The eight left-and-right-context-only corresponding rules are deduced from the L+R rules.

Since the result we seek is not to normalize the spelling, each rule can be used in both directions which are considered equally frequent.

From the 367 variant pairs collected for Alsatian, we extracted 213 unique rules using the left and right contexts, 227 rules using the left context only, 186 rules using the right context only.

3.2 Variant Identification and Filtering

Given a vocabulary of known words V_{lookup}, the identification of potential variants of an OOV word includes (i) optional preliminary filtering, (ii) application of rules, and (iii) lookup:

1. preliminary filtering (optional): if the unknown word is identified as a known proper noun in the lexicon, it is ignored.

2. application of the rules: for each set of rules, L+R, L, R, used in this order, the subset of rules applying to the original OOV word $R_{original\_word}$ is identified, and ordered by rule frequency. From this subset, we apply on the OOV word each possible *combination* of rules, meaning that if three rules A, B, C apply, the sequences of rules {A},{B},{C},{A;B},{A;C},{B;C} and {A;B;C} are applied.

3. lookup: the sequence of rules apply until the produced form is matched with a word present in V_{lookup}.

Although this “brute-force” method generates a great quantity of noise, the filtering operated by V_{lookup} leads to the matching of OOV word with existing variant candidates.

Since part of the dialectal and spelling variation mechanisms may be similar to some of the language morphological rules (such as gender, number, conjugation or declension), the generated variant pairs should be manually checked in context.

This phenomenon is illustrated by the analysis of the pairs generated for Alsatian in Section 5.

4 Evaluation on a Downstream Task

To illustrate the benefits of the identification of variant pairs, we evaluate its impact on a downstream task: part-of-speech (POS) tagging.

Previous experiments on Alsatian from (Millour and Fort, 2018b) have shown that using multiple variants for training can lead to a drop of accuracy on the sections of the evaluation corpus which do not match the variants represented in the training corpus (-1.4% accuracy when a corpus of Strasbourg specific variant is added in the training of a Southern variant only).

In this context, we use our methodology to match OOV words from the evaluation corpus with their potential spelling variant appearing in the training corpus.

It is important to understand that this process is independent from the tagger, and occurs after it has been trained. The extraction of pairs is performed at the time of annotation on a previously **unseen** corpus.

4.1 Language Resources and Tools Used for Evaluation

Experiments in POS tagging Alsatian include our previous work (Millour and Fort, 2018b), which uses `MElt` (Denis and Sagot, 2012), a freely available sequence labeller achieving at best 84% accuracy when the variants in the training and the evaluation corpus are carefully controlled. Experiments using word embeddings have been also been carried on Alsatian by (Magistry et al., 2018), using a raw corpus of 200 000 tokens and reaching 91% accuracy.

In the following experiments, we chose to train `MElt`, which enables us to take advantage of available lexicons existing for Alsatian. The differential in performance is more interesting to us than the performance *per se*, which is why we chose not to focus on testing our methodology on other taggers.

Two POS-tagged corpora are available for Alsatian. Both are made of texts produced in an uncontrolled environment (such as `Wikipedia`[5]) and contain multiple variants of the language:

- The `Crowdsourced Corpus` (Millour and Fort, 2018b), $C_{rowd}C$, annotated by benevolent participants on a dedicated crowdsourcing platform `Bisame`[6] with the universal POS tagset (Petrov et al., 2012) extended with two categories: APPART (preposition-determiner contraction), and FM (foreign words). The corpus contains 9,282 tokens (439 sentences), and is available under CC BY-NC-SA license. The accuracy of the annotations provided by the benevolent participants has been evaluated to 93% (Millour and Fort, 2018b).

- The `Annotated Corpus for the Alsatian Dialects` (Bernhard et al., 2018a), $T_{rad}C$, annotated with the tagset described above, extended with the categories EPE (epenthesis) and MOD (modal verb) (Bernhard et al., 2018b). The corpus contains 12,570 tokens (533 sentences) and is available under CC BY-SA license. It was annotated manually by expert linguists.

We manually corrected $T_{rad}C$ to match the tagset used in $C_{rowd}C$.

The corpus resulting from the concatenation of the two corpora, $C_{oncat}C$, was used for the following experiments. We performed a cross validation on 4 subdivisions (80% used for training, $C_{oncat}C80$, 20% for the evaluation, $C_{oncat}C20$).

We also have at our disposal two lexica:

- a multi-variant lexicon $M_{ultiVar}L$ of 54,355 entries annotated with their POS, containing grammatical words (Bernhard and Ligozat, 2013), verbs from (Steiblé and Bernhard, 2016), and various entries from (i) the Office for Alsatian Language and Culture (OLCA)

[5] See `https://als.wikipedia.org`

[6] See `https://bisame.paris-sorbonne.fr`.

bilingual lexicons, (ii) the dictionary compiled by the Culture and Heritage of Alsace Association (ACPA), and (iii) a multilingual French-German-Alsatian dictionary (Adolf, 2006).

- the `Lexicon of Place Names in the Alsatian Dialects` which contains 1,346 entries (Bernhard, 2018), used during training only.

4.2 Application of the Methodology

Since the identification of potential variant pairs depends on the initial conditions of the experiment, *i.e.* the corpus, and optionally, the lexica used to train the model beforehand, we present two experiments in which these parameters vary.

For each experiment, we extract from the training corpus the vocabulary VT_lookup and from the external lexicon, the vocabulary VL_lookup. We use the set of rules presented in Section 3.1.

We prioritize the lookup in VT_lookup to further ease the evaluation of the generated pairs relying on the context.

If the length of the OOV word is less or equal to four characters (^ and $ excluded), only the L+R rules are applied: it has been observed in preliminary tests that shorter words were more likely to lead to erroneous matching such as *das* (determiner) /*dass* (subordinating conjunction) or *dien* (auxiliary) /*dene* (determiner). Additionally, we force the variant candidates to have the same letter case as the OOV word.

After variant pairs have been generated, the OOV words are replaced by their variant candidate, and the pre-trained model is applied on the transposed evaluation corpus. After the corpus has been tagged, the transposed words are replaced by they original form.

4.3 Experiment 1: Uncontrolled Setup

By "uncontrolled", we mean that training and evaluation corpora are both extracted from a shuffled corpus that contains multiple variants.

Our first model is trained with $C_{oncat}C80$ (17,136 words) and evaluated on $C_{oncat}C20$ (4,374 words) before and after its transposition. After the application of the three sets of rules, using both the vocabularies extracted from $C_{oncat}C80$ and $M_{ultiVar}L$ for the lookup, 56 variant pairs were discovered and the same number of words were transposed.

	Before transp.	After transp.
Overall	0.859	0.864
OOV words	24%	22%

Table 2: Accuracy of the model trained on multi-variant corpora, before and after the corpus transposition.

The proportion of OOV words was diminished by around 2% resulting in an improvement of the tagging performance of 0.5 points (see table 2). This minimal impact is expected since the performance on "known words" is around 10 points higher than on OOV words in this setup. In fact, considering the sizes of our corpora, lowering the number of OOV words of 100 is expected to improve the overall results of 0.2 points.

4.4 Experiment 2: Controlled Setup

By "controlled", we mean that training and evaluation each contain a specific variant of Alsatian selected in a multi-variant corpus. In the following, we compare homogeneous and heterogeneous setups, in which the training and evaluation corpora either contain the same or distinct variants of Alsatian.

To highlight the effect of our methodology in an heterogeneous context, met when no corpus of each possible variant is available, we manually split $C_{oncat}C$ in two sub-corpora $N_{orth}C$ (4,880 words) and $S_{outh}C$ (7,690 words) based on the frequencies of the -e and -a noun endings, which are specific of the Northern and Southern variants respectively.

The results of these experiments are presented in table 3.

Unsurprisingly, the best results are obtained when training and evaluation corpora are of the same variant. Yet, we can observe that in this setup, the effect of transposition to identified variants has a higher impact on the proportion of OOV words and the tagging performances.

The efficiency of the methodology largely depends on: (i) the respective and relative sizes of the training and evaluation corpora, (ii) the variation in variants existing between them.

This experiment shows that the performance of a tool trained on a given corpus can be improved by modifying the corpus it is applied on to match the vocabulary it was trained with.

	$N_{orth}C20$		$S_{outh}C20$	
$N_{orth}C80$			Before transp.	After transp.
Overall	0.853		0.714	**0.752**
OOV words	40%		54%	52%
$S_{outh}C80$	Before transp.	After transp.		
Overall	0.788	**0.809**	0.864	
OOV words	51%	48%	29%	

Table 3: Accuracy of the model trained on mono-variant corpora, before and after the corpus transposition.

5 Obtained Results

The newly created resource contains 876 pairs of variants, from which 400 were identified in the training corpus, and 476 in the lookup lexicon. The size of the created resource depends on the size of the lookup corpora and lexicon, and on the number of rules. The application of the method to any unpreviously seen text may increase the number of variant pairs.

A subset of 60 pairs of these automatically generated variant pairs were submitted to an Alsatian teacher, familiar with both the dialectal and spelling variants. The pairs were presented in the context of their sentence. The expertise of the teacher was used to measure the precision of the pairs, not the recall.

Among the 60 pairs:

- 30 were actual dialectal or spelling variants.
- 13 were pairs of different forms of identical words, *e.g.*: *ìhm* (dative pronoun) / *ìrhem* (genitive pronoun), *kált* (feminine adjective) / *kálte* (masculine adjective), *wùrd* (future auxiliary) / *wärd* (conditionnal auxiliary) etc.
- 10 were caused by erroneous matching we managed to correct by making the adjustments described in section 4.2, i.e. (i) forcing the case of potential variants to match the case of the original OOV word, (ii) limiting the application of rules considering only left or right context to words which size is over four characters.
- 7 were caused by erroneous matching we were not yet able to correct *e.g.* *kräfti* ("strongly", adverb) / *kräftiger* ("stronger", adjective), *mine* ("mine", determiner) / *meine* ("believe", verb) etc.

These results show that the generated variant pairs should be hand-checked, a task that can itself be crowdsourced, provided that we have access to the context of appearance of both elements.

By construction, the newly generated pairs will not provide additional substitution rules. Yet, they provide information on the frequency of the substitution patterns.

Additionally, the erroneous variant pairs manually filtered out can be used as counter examples of variants, and further used to train variant classifiers (see, for instance, (Barteld, 2017)).

6 Related Work

Dealing with non-standardized, less-resourced languages, takes us to the limits of NLP: first, we have no standard to rely on and not enough expert linguists to help us, and second, very few language resources are available for us to work with, even raw corpora. These two constraints are rarely met in the literature and, to our knowledge, the solution we propose has never been used before.

However, it closely relates to other experiments that involve at least a standard spelling and sometimes an expert linguist supervision. One such example is VARD 2, a tool that allows to manually and automatically standardize Early Modern English (Baron and Rayson, 2008, 2009). Another one concerns the Basque language (Etxeberria Uztarroz et al., 2014) and proposes a solution to map the variations of the language to the standard form using an existing morphological analyzer and a parallel corpus. Obviously, a lot of more or less recent publications concern the design and use of morphophonological rules, in particular in various flavors of FSTs, but most of them require the intervention of a highly-skilled linguist.

Among such publications, the work by Kimmo Koskenniemi on modeling regular correspondences between Finnish and Estonian is particularly inspiring (Koskenniemi, 2013), but inapplicable in our case. The same goes for the type for work described in (Theron and Cloete, 1997), in which the rules are automatically extracted but with a known (morphological) goal.

The closest work to ours is that described in (Barteld, 2017), as it focuses on detecting spelling variants in Middle Low German unrelated to a standard. Yet, the described method requires the training of a classifier to filter the generated pairs. This classifier is based on a resource that contains 1,834 pairs of spelling variants, a resource that is unavailable for most non-standardized languages.

Regarding Alsatian more specifically, Bernhard (2014) aligns spelling variants relying on a multi-variant bilingual French-Alsatian lexicon annotated with part-of-speech and a phonetization of Alsatian. This high dependency on existing resources make this method challenging to adapt to other languages for which the only available experts are the very speakers of the language.

7 Conclusion

We have presented a method to automatically generate pairs of spelling variants based on a small subset of crowdsourced pairs.

The method does not require manual rules definition by experts and is language independent. The resources needed to perform variant pair detection can be easily produced by the speakers, who hold the knowledge of the of the variation mechanisms. The crowdsourcing of variants, unlike that of POS tags, requires no prior training.

In fact, even the expertise necessary for the validation of the variant pairs is about to be transferred to the participants of the crowdsourcing platform.

The originality of this methodology is that once the rules have been extracted, the process feeds from previously unseen texts. This is particularly useful in a less-resourced scenario where a raw corpus is being collected from various sources.

The code of both the gamified crowdsourcing platform and the variants generation is freely available on GitHub[7]. The created multi-variant lexicon is also available under a CC license.

We plan to extend this work to other non-standardized languages. We have started working on adapting the platform to Mauritian, a French-based Creole, the morphology of which is very different from that of Alsatian.

[7] See `https://github.com/alicemillour/Bisame/tree/recipes`.

Acknowledgments

We wish to thank the participants of `Bisame` and `Recettes de Grammaire` for their motivation and comments, as well as J-N. S. Kempf for his time and expertise on the Alsatian variants. We also thank B. Sagot who inspired us with the title of this paper.

References

Paul Adolf. 2006. *Dictionnaire comparatif multilingue: français-allemand-alsacien-anglais*. Midgard, Strasbourg, France.

Alistair Baron and Paul Rayson. 2008. Vard 2: A tool for dealing with spelling variation in historical corpora. In Aston University, editor, *Proceedings of the Postgraduate Conference in Corpus Linguistics,*. Birmingham, UK.

Alistair Baron and Paul Rayson. 2009. Automatic standardisation of texts containing spelling variation: How much training data do you need? In University of Liverpool, editor, *Proceedings of the Corpus Linguistics Conference*. Liverpool, UK.

Corinne Barre and Mélanie Vanderschelden. 2004. *L'enquête "étude de l'histoire familiale" de 1999 - Résultats détaillés*. INSEE, Paris.

Fabian Barteld. 2017. Detecting spelling variants in non-standard texts. In *Proceedings of Student Research Workshop (EACL 2017)*. Valencia, Spain.

Delphine Bernhard. 2014. Adding dialectal lexicalisations to linked open data resources: the example of alsatian. In *Proceedings of the Workshop on Collaboration and Computing for Under Resourced Languages in the Linked Open Data Era (CCURL 2014)*. Reykjavik, Iceland, pages 23–29. https://hal.archives-ouvertes.fr/hal-00966820.

Delphine Bernhard. 2018. Lexicon of place names in the alsatian dialects. https://doi.org/10.5281/zenodo.1404873.

Delphine Bernhard, Pascale Erhart, Dominique Huck, and Lucie Steiblé. 2018a. Annotated corpus for the alsatian dialects. Guide d'annotation, LiLPa, Université de Strasbourg.

Delphine Bernhard and Anne-Laure Ligozat. 2013. Es esch fàscht wie Ditsch, oder net? étiquetage morphosyntaxique de l'alsacien en passant par l'allemand. In *Proceedings of TALARE (Traitement Automatique des Langues Régionales de France et d'Europe) (TALN'13)*. Les Sables d'Olonne, France, pages 209–220.

Delphine Bernhard, Anne-Laure Ligozat, Fanny Martin, Myriam Bras, Pierre Magistry, Marianne Vergez-Couret, Lucie Steible, Pascale Erhart, Nabil Hathout, Dominique Huck, Christophe Rey, Philippe Reynés, Sophie Rosset, Jean Sibille, and Thomas Lavergne. 2018b. Corpora with Part-of-Speech Annotations for Three Regional Languages of France: Alsatian, Occitan and Picard. In *Proceedings of 11th edition of the Language Resources and Evaluation Conference (LREC 2018)*. Miyazaki, Japan. https://hal.archives-ouvertes.fr/hal-01704806.

Jon Chamberlain, Karën Fort, Udo Kruschwitz, Mathieu Lafourcade, and Massimo Poesio. 2013. Using games to create language resources: Successes and limitations of the approach. In Iryna Gurevych and Jungi Kim, editors, *The People's Web Meets NLP*, Springer Berlin Heidelberg, Theory and Applications of Natural Language Processing, pages 3–44. https://doi.org/10.1007/978-3-642-35085-6_1.

Danielle Crévenat-Werner and Edgar Zeidler. 2008. *Orthographe alsacienne - Bien écrire l'alsacien de Wissembourg à Ferrette*. Jérôme Do Bentzinger.

Pascal Denis and Benoît Sagot. 2012. Coupling an annotated corpus and a lexicon for state-of-the-art pos tagging. *Lang. Resour. Eval.* 46(4):721–736. https://doi.org/10.1007/s10579-012-9193-0.

Izaskun Etxeberria Uztarroz, Iñaki Alegría Loinaz, Mans Hulden, and Larraitz Uria Garin. 2014. Learning to map variation-standard forms in basque using a limited parallel corpus and the standard morphology. *Procesamiento del Lenguaje Natural* 52:13–20.

Kimmo Koskenniemi. 2013. Finite-state relations between two historically closely related languages. In Northern European Association for Language Technology, editor, *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013*. Oslo , Norway, volume 18, pages 43–53.

Pierre Magistry, Anne-Laure Ligozat, and Sophie Rosset. 2018. Étiquetage en parties du discours de langues peu dotées par spécialisation des plongements lexicaux. In *Proceedings of Conférence sur le Traitement Automatique des Langues Naturelles*. Rennes, France. https://hal.archives-ouvertes.fr/hal-01793092.

Alice Millour. 2019. Getting to Know the Speakers: a Survey of a Non-Standardized Language Digital Use. In *Proceedings of 9th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Poznań, Poland. https://hal.archives-ouvertes.fr/hal-02137280.

Alice Millour and Karën Fort. 2018a. À l'écoute des locuteurs : production participative de ressources langagières pour des langues non standardisées. In *Revue TAL : numéro spécial sur les langues peu dotées (59-3)*, Association pour le Traitement Automatique des Langues.

Alice Millour and Karën Fort. 2018b. Toward a Lightweight Solution for Less-resourced Languages: Creating a POS Tagger for Alsatian Using Voluntary Crowdsourcing. In *Proceedings of 11th International Conference on Language Resources and Evaluation (LREC'18)*. Miyazaki, Japan. https://hal.archives-ouvertes.fr/hal-01790615.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of 8th International Conference on Language Resources and Evaluation (LREC 2012)*. Istanbul, Turkey.

Jelena Prokić, Martijn Wieling, and John Nerbonne. 2009. Multiple sequence alignments in linguistics. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 18–25. http://dl.acm.org/citation.cfm?id=1642049.1642052.

Lucie Steiblé and Delphine Bernhard. 2016. Towards an Open Lexicon of Inflected Word Forms for Alsatian: Generation of Verbal Inflection. In *JEP-TALN-RECITAL 2016*. Paris, France, volume 2 of *Proceedings of la conférence conjointe JEP-TALN-RECITAL 2016, volume 2 : TALN*, pages 547–554. https://hal.archives-ouvertes.fr/hal-01338411.

Pieter Theron and Ian Cloete. 1997. Automatic acquisition of two-level morphological rules. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, ANLC '97, pages 103–110. https://doi.org/10.3115/974557.974573.

Neural Feature Extraction for Contextual Emotion Detection

Elham Mohammadi, Hessam Amini and Leila Kosseim
Computational Linguistics at Concordia (CLaC) Lab
Department of Computer Science and Software Engineering
Concordia University, Montréal, Québec, Canada
first.last@concordia.ca

Abstract

This paper describes a new approach for the task of contextual emotion detection. The approach is based on a neural feature extractor, composed of a recurrent neural network with an attention mechanism, followed by a classifier, that can be neural or SVM-based. We evaluated the model with the dataset of the task 3 of SemEval 2019 (EmoContext), which includes short 3-turn conversations, tagged with 4 emotion classes. The best performing setup was achieved using ELMo word embeddings and POS tags as input, bidirectional GRU as hidden units, and an SVM as the final classifier. This configuration reached 69.93% in terms of micro-average F1 score on the main 3 emotion classes, a score that outperformed the baseline system by 11.25%.

1 Introduction

Emotions are an intricate part of human communication. Being able to interpret and react to the emotions of others allows one to better communicate. Emotion information can be extracted from a variety of physiological sources, such as electroencephalography (EEG) signals (Zhang et al., 2016), skin temperature (Li and Chen, 2006), speech signals (Trigeorgis et al., 2016) and facial expressions (Mao et al., 2015), as well as text (Yassine and Hajj, 2010).

The rise of social media and the availability of human-written online diaries, blog posts, and comments has lead to an increase in research on the automatic detection of sentiment and emotion from textual data.

Sentiment analysis and opinion mining can range from coarse-grained binary classification of texts into positive or negative classes to finer-grained classification into a variety of emotion categories, such as happy, sad, angry, and scared. Such a classification is useful in business and marketing (Medhat et al., 2014), and a variety of downstream NLP applications, such as text-to-speech, to maintain the emotion present in text, and human-computer interaction in order to take into account the emotional state of users and make responses more human-like (Hirat and Mittal, 2015).

Fine-grained emotion detection based solely on text is a challenging task. As only linguistic cues are available, facial expressions and voice features, which are known to be discriminating (Poria et al., 2016), cannot be used. In addition, several emotions can be expressed textually by the same linguistic cues, such as emotion keywords, interjections, and emojis (Liew and Turtle, 2016). Finally, many pieces of text, especially online comments, posts, and tweets are too short to allow for correct classification. These challenges highlight the importance of using contextual information in order to detect the emotion conveyed in a piece of text.

The goal of this work is to investigate the effectiveness of different models using neural networks and Support Vector Machines (SVM) for contextual emotion detection in textual conversations.

2 Related Work

Textual emotion detection has typically been addressed as a multi-class classification task, where a text is classified into different emotional categories, ranging from basic emotions to finer-grained emotional classes. Studies focusing on emotion detection have made use of different corpora and different evaluation metrics.

Dini and Bittar (2016) broke down the task of emotion detection from tweets into a cascade of decisions: classifying tweets into emotional and non-emotional categories, and then tagging the emotional tweets with the appropriate emotion label. For the latter, they compared a symbolic system using gazetteers, regular expressions, and graph transformation, with a machine learning system using a linear classifier with words, lemmas, noun phrases, and dependencies as features. Using their collected corpus of emotional tweets, the rule-based approach achieved an F1 score of 0.41, while the machine learning approach yielded an F1 score of 0.58 on 6 emotion classes.

Mohammad and Bravo-Marquez (2017) made use of an SVM regression model to determine the intensity of 4 emotions: anger, fear, joy, and sadness in a dataset of tweets that they have previously collected and annotated. As features, they used word and character n-grams, word embeddings trained using the word2vec skip-gram model (Mikolov et al., 2013), and affect-related lexical features. Using the Pearson correlation coefficient as evaluation metric, they demonstrated that word embeddings yield better results than n-gram features. They achieved their best average result of 0.66, using a combination of word embeddings and lexical features.

Abdul-Mageed and Ungar (2017) also collected their own dataset of emotional tweets using emotion hashtags. They trained word embeddings on the training data, employed a gated recurrent neural network (Cho et al., 2014) as a classifier and achieved an average F1 score of 0.87 over 3 emotion datasets, labelled with 8 emotions.

Abdullah and Shaikh (2018) proposed an approach to detect the intensity of affect in tweets. Their features include feature vectors extracted using the *AffectiveTweets* package of *Weka* (Holmes et al., 1994), as well as word2vec and doc2vec (Le and Mikolov, 2014) embeddings. They developed three models using different subsets of the feature set as input to either a dense feed-forward network or a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network. Using the dataset of SemEval 2018 task 1 (Affect in Tweets) (Mohammad et al., 2018), they achieved their best Spearman correlation score of 0.69 over 4 emotions by averaging over the outputs of the three models.

More recently, Khanpour and Caragea (2018) focused on domain-specific emotion detection. They created a dataset of 2107 sentences taken from online forums on the Cancer Survivors Network website[1]. In order to combine the strengths of lexicon-based and machine learning approaches, they proposed a model that uses word2vec embeddings as input to a Convolutional Neural Network (CNN) (LeCun et al., 1999). The CNN generates feature vectors which are then augmented with domain-specific lexical features. The combined features are then used as input to an LSTM network which classifies the texts into 6 different emotion categories.

While most of the literature has focused on the detection and assessment of emotions in online textual data, few researchers have investigated emotion detection in textual conversations. We argue that the detection of emotions from dialogues poses new challenges compared to emotion detection from monologues, as the utterances made by different interlocutors can influence differently the emotional state of a speaker.

In this work, we investigate the effectiveness of neural feature extraction for the task of emotion detection in short dialogues.

3 Dataset and Task

The dataset used in this work is taken from Chatterjee et al. (2019b). It consists of short 3-turn dialogues between two speakers (turn 1 uttered by speaker 1, turn 2 uttered by speaker 2, and turn 3 uttered by speaker 1 again). Table 1 shows two samples of the dataset[2].

The goal is to detect the emotion of speaker 1 in turn 3, taking into account the previous turns. The data is annotated with 4 emotions: *happy*, *angry*, *sad*, and *others*. In order to simulate a real-life task, the distribution of the labels in the dataset is highly imbalanced: 50% of the training data belongs to the *others* class, while 14%, 18%, and 18% of the training data is dedicated to classes *happy*, *angry*, and *sad*, respectively. The test and development sets are even more imbalanced, with 85% of the samples labelled as *others*. Table 2 summarizes some statistics of the dataset.

[1] https://csn.cancer.org/

[2] Samples are taken from https://competitions.codalab.org/competitions/19790.

ID	**Turn1 (Speaker1)**	**Turn2 (Speaker2)**	**Turn3 (Speaker1)**	**Label (of Turn3)**
156	You are funny	LOL I know that. :p	☺	happy
187	Yeah exactly	Like you said, like brother like sister ;)	Not in the least	others

Table 1: Two sample dialogues from the EmoContext 2019 dataset.

Dataset	**Label**	**# of Samples**	**Percentage**	**Average # of Tokens**		
				Turn 1	**Turn 2**	**Turn 3**
Train	*happy*	4243	14%	4.873	7.195	3.825
	angry	5506	18%	5.107	6.859	5.457
	sad	5463	18%	4.608	6.450	4.829
	others	14948	50%	4.232	6.493	4.153
	All	30160	100%	4.550	6.650	4.467
Development	*happy*	142	5%	4.761	7.444	3.690
	angry	150	5%	4.647	7.347	4.867
	sad	125	5%	4.624	6.200	5.192
	others	2338	85%	4.245	6.546	4.143
	All	2755	100%	4.311	6.620	4.207
Test	*happy*	284	5%	5.063	6.845	3.493
	angry	298	5%	4.470	6.456	4.856
	sad	250	5%	5.000	6.632	4.936
	others	4677	85%	4.279	6.601	4.143
	All	5509	100%	4.362	6.607	4.184
All	*happy*	4669	12%	4.881	7.181	3.801
	angry	5954	16%	5.063	6.851	5.412
	sad	5838	15%	4.626	6.452	4.842
	others	21963	57%	4.243	6.521	4.150
	All	38424	100%	4.506	6.642	4.408

Table 2: Statistics of the EmoContext 2019 dataset.

4 The Model

Figure 1 shows the overall architecture of our model for the task of contextual emotion detection. The model is composed of two main components: 1) the neural feature extractor and 2) the classifier.

4.1 The Neural Feature Extractor

As shown in Figure 1, the neural feature extractor is a recurrent neural network with an attention mechanism. The feature extractor is responsible for creating dense vector representations for each dialogue turn. As a result, the model uses 3 feature extractors, one for each dialogue turn.

Each neural feature extractor is composed of an input layer, a recurrent layer, and an attention layer, explained below.

The Input Layer takes as input the vector representations of each word in the corresponding dialogue turn. Each dialogue turn is a sequence of tokens, represented as a vector $[x_{i,1}, x_{i,2}, \ldots, x_{i,t}, \ldots, x_{i,n}]$, where $x_{i,t}$ is the corresponding vector for the t-th word in the i-th dialogue turn, and n is the length of the i-th turn. The vector representation for each token ($x_{i,t}$) is composed of the word embedding corresponding to the token, concatenated with a one-hot representation of the token's part-of-speech (POS) tag.

The Recurrent Layer takes as input the token vectors ($[x_{i,1}, x_{i,2}, \ldots, x_{i,t}, \ldots, x_{i,n}]$), and processes them in a forward and a backward passes. In the forward pass, the content value of the hidden layer at a specific time-step is calculated using the value of the input at the current time-step, and the content value of the hidden layer in the previous time-step.

Equation 1 shows how the content value of the hidden layer is calculated at a specific time-step t, where x_t represents the input value in the current time-step, and h_t and $h_{t\pm1}$ represent the content value of the hidden node in the current and previous/next time-steps (in the forward or backward pass), respectively, and f_h is the function that calculates the value of h_t using x_t and $h_{t\pm1}$. Subsequently, the output of the hidden layer is calculated using Equation 2, where y_t is the output of the hidden layer at time-step t, and f_y is the function that calculates the output value based on h_t.

$$h_t = f_h(x_t, h_{t\pm1}) \tag{1}$$

$$y_t = f_y(h_t) \tag{2}$$

The Attention Layer is a function that automatically assigns weights to the output of the recurrent layer at each time-step, and calculates the weighted sum of the outputs using their corresponding weights (Vaswani et al., 2017). Following several works that have shown significant improvement in text classification with the use of attention (e.g. Yang et al., 2016; Zhou et al., 2016; Wang et al., 2016; Cianflone et al., 2018), we incorporated an attention mechanism in our contextual emotion detection framework. Equation 3 shows the overall mechanism of our attention layer, where $\omega_{t'}$ represents the corresponding weight for the output of the recurrent layer at time-step t' in, and n is the number of time-steps (i.e. the length of the dialogue turn).

$$Attention = \sum_{t'=1}^{n} y_{t'}\omega_{t'} \tag{3}$$

In our model, the weights are calculated by applying a single N-to-1 feed-forward layer on

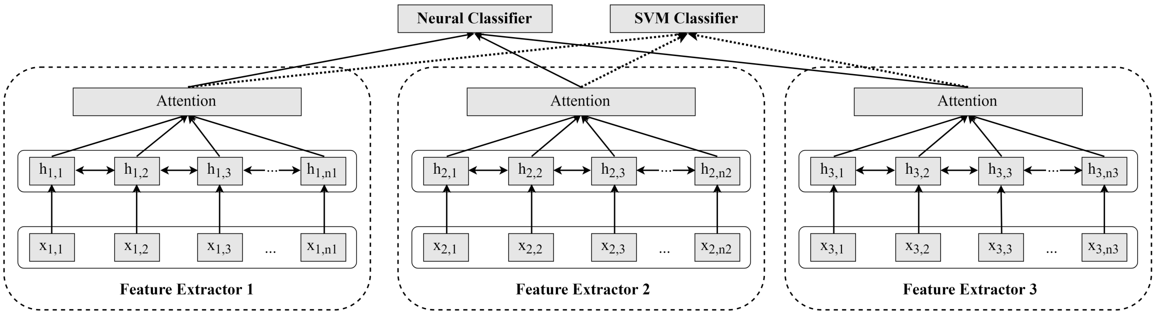

Neural Classifier
SVM Classifier
Attention
Attention
Attention
$h_{1,1}$ $h_{1,2}$ $h_{1,3}$ ... $h_{1,n1}$
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ ... $x_{1,n1}$
Feature Extractor 1
$h_{2,1}$ $h_{2,2}$ $h_{2,3}$ ... $h_{2,n2}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ ... $x_{2,n2}$
Feature Extractor 2
$h_{3,1}$ $h_{3,2}$ $h_{3,3}$ ... $h_{3,n3}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ ... $x_{3,n3}$
Feature Extractor 3

Figure 1: Architecture of the model.

the output of the recurrent layer at each time-step (where N is the size of the output of the recurrent layer), concatenating the results, and applying a softmax over them. Equations 4 and 5 show the mechanisms used to calculate the weights, where w corresponds to the weights in the single-layer neural network, and ν_t is the single value, which is the result of feeding y_t to the fully-connected layer.

$$\nu_t = y_t \times w \tag{4}$$

$$\omega = Softmax([\nu_1, \nu_2, \nu_3, \ldots, \nu_n]) \tag{5}$$

4.2 The Classifier

As shown in Figure 1, we experimented with two types of classifiers at the output layer: A fully-connected neural network, followed by a softmax activation function, and an SVM, which takes as input the neural representations generated by the 3 latent feature extractors for each dialogue turn.

The neural classifier is trained jointly with the neural feature extractors, while the SVM classifier is completely trained after each training epoch of the neural network, using the features extracted by the 3 neural feature extractors.

5 Experimental Setup

The neural network components of our model were developed using PyTorch (Paszke et al., 2017) and the SVM was developed using the Scikit-learn library (Pedregosa et al., 2011). In this section, we will explain the different setups that we experimented for the task of contextual emotion detection.

5.1 Word Embeddings

In order to test our model, we experimented with two different pretrained word embeddings. As the first word embedder, we chose GloVe (Pennington et al., 2014), which is pretrained on 840B tokens of web data from Common Crawl, and provides 300d vectors as word embeddings. As our second word embedder, we experimented with ELMo (Peters et al., 2018), which produces word embeddings of size 1024, and is pretrained on the 1 Billion Word Language Model Benchmark[3] (Chelba et al., 2014).

The main reason for choosing these two word embedders was to evaluate the effect of their embedding mechanisms for our task. As opposed to GloVe which assigns a word embedding to each token, the ELMo word embedder calculates the embedding for each token from its constituent characters by also taking into account its textual context. We suspected that this approach would lead to better results in our task (see Section 6).

5.2 POS Tags

The spaCy library[4] was used for tokenization and POS tagging, and the Penn Treebank tagset standard (Marcus et al., 1993) was followed for assigning POS tags to tokens. This lead to one-hot vectors for POS information of size 51.

5.3 Recurrent Units

Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014) were both experimented with as the building blocks of the recurrent layer. For both LSTM and GRU, 2 layers of 25 bidirectional recurrent units were stacked.

5.4 Neural Network Optimization

The Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-4} was used to train the neu-

[3] The selected versions of GloVe and ELMo lead to the best results in our task. We also experimented with other versions of the two, but their performances were inferior.

[4] `https://spacy.io/`

ral network. Cross-entropy with class weights was used as the loss function. In order to handle the imbalanced class distribution in the dataset (see Table 2), the corresponding weight for each class was calculated proportional to the inverse of the number of samples for that class in the training data. This way, more penalty was applied to the network when an error was made on a sample from a minority class rather than a more frequent one.

Minibatches of size 32 were used during training and testing, and zero-padding was applied in order to handle different input sequence lengths. In order to minimize padding, samples with similar average lengths of tokens over the three turns were put in the same batch.

Finally, in order to avoid the exploding gradient problem (Pascanu et al., 2012), gradient clipping with a norm of 0.5 was applied.

5.5 SVM Hyperparameters

The SVM utilizes a polynomial kernel with degree of 4. To set the parameter γ, the `svm.SVC` model in Scikit-learn was initiated with its parameter `gamma` set to `auto`, which automatically sets γ to the inverse of the number of features extracted by the neural feature extractor. In our model, this value was set to 1/150, since each of the three neural feature extractors extracts 50 features from the dialogue turn that it handles.

5.6 Overall Training Process

As indicated in Section 4.2, the neural classifier was trained jointly with the neural feature extractors, while the SVM was trained separately after each epoch, using the extracted features on the training data.

The models with either neural or SVM classifier were trained for 50 epochs, and the model's parameters were saved after each training epoch. The optimal parameters were then picked as the ones that led to the highest micro-average F1 score on the three main emotion classes (all except class *other*) on the development dataset. This final model with the optimal trained parameters was then evaluated on the test set.

6 Results

The official evaluation metric used at the EmoContext shared task is the micro-average F1 score over the three main emotion classes, i.e. *happy*, *angry*, and *sad* (ignoring the 4th class, *others*).

Figure 2 shows the performance of each model on the development dataset, throughout the training process. As Figure 2 shows, using both the neural classifier and the SVM classifier, the models with GRU as the recurrent units and ELMo embeddings as input features were generally superior to the others.

The notation *<type-of-recurrent-unit>+<type-of-classifier> with <type-of-input-features>* is used in the rest of the paper, to refer to each model; for example, *LSTM+NN with GloVe* refers to the model that uses LSTM units in the recurrent layer, fully-connected neural layer as classifier, and GloVe embeddings as input; whereas *GRU+SVM with ELMo+POS* denotes the version of our model with GRU units in the recurrent layer, SVM as the classifier, and ELMo embeddings and one-hot encoded POS tags as input.

As indicated in Section 5.6, the final versions of the models were chosen based on their performance on the development dataset, i.e. for each model, the final set of trained parameters were the one that yielded the maximum micro-average F1 score on the three emotion classes on the development dataset.

The results achieved from our models are also compared with the baseline system, provided by the EmoContext 2019 shared task (Chatterjee et al., 2019a). The baseline system is composed of a neural network with 128 LSTM units in the hidden layer, and as input features, uses the GloVe word embeddings, pretrained on 6 billion tokens from Wikipedia 2014 and the Gigaword 5 corpus[5].

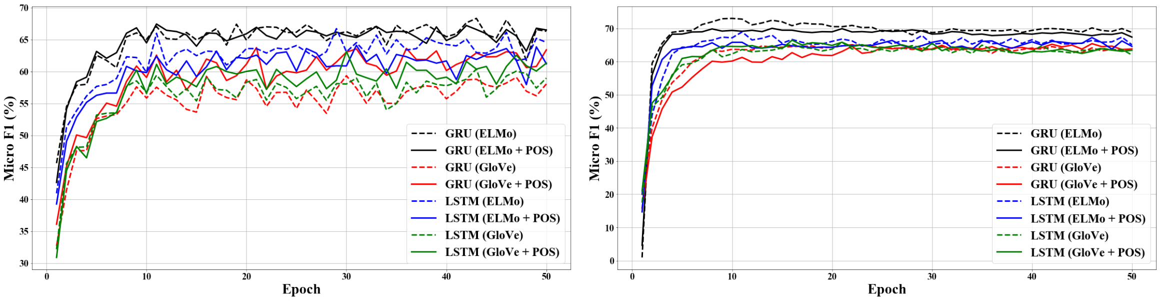
Table 3 shows the performance of each model[6], where the best micro-average F1 scores are highlighted in bold. The results show that the model *GRU+SVM with ELMo* yields the best performance of 73.03% on the development data, while the model *GRU+SVM with ELMo+POS* outperforms all the other models on the test dataset with a micro-average F1 score of 69.93%, by being marginally better than *GRU+SVM with ELMo*[7].

The results also show that, with the exception of the two models *LSTM+NN with GloVe* and *GRU+NN with GloVe* which have inferior perfor-

[5] `https://catalog.ldc.upenn.edu/LDC2011T07`

[6] At the time of writing this paper, the F1 score of the baseline model had not been released for each emotion class.

[7] The result that we submitted to the EmoContext shared task was slightly higher than the current result (70.72%), which was achieved by also using the features from the development dataset to train the SVM classifier.



(a) With the neural classifier.

(b) With the SVM classifier.

Figure 2: Performance of the model (in micro-average F1 score over the three main emotion classes) on the development dataset, as a function of the number of training epochs

mance than the baseline system on the test dataset, all the other models significantly outperform the baseline model on both development and test data.

7 Discussion

To better understand the results, we analyzed the effect of different components of the model.

7.1 Effect of Input Features

The results in Table 3 demonstrate that the models that use ELMo as word embeddings have a significantly higher performance than the ones with GloVe. We believe that this is due to two main reasons: 1) The ELMo word embedder is character-based, which allows it to better handle out-of-vocabulary words, and 2) ELMo takes into account the textual context of the token when extracting the word embedding.

Table 3 also shows that the use of POS tags leads to a significant improvement with the models that utilize GloVe word embeddings. On the other hand, for the models that use ELMo embeddings as input, this is not the case. As Table 3 shows, in several occasions, the use of POS tags has even reduced the performance of ELMo-based models. We believe that, in the case of GloVe, where the word embeddings are context-independent, POS tags can improve token representations to also take into account the textual context in which the tokens have occurred. However, since ELMo already takes into account the textual context when extracting the token representations, POS tags do not help much and can even be redundant in some cases.

7.2 Effect of the Recurrent Units

The results in Table 3 show that for the models that incorporate ELMo embeddings, the ones that use GRU in their recurrent layer significantly outperform the ones with LSTM; however, this behavior cannot be observed in GloVe-based models, as we can see several cases, where the LSTM-based models are slightly better.

It could be concluded that, for the current task, since GloVe word embeddings are context-independent, a stronger recurrent unit is required to capture the context, while in the case of ELMo, where context is already taken into account, a simpler recurrent unit such as GRU is enough while being less prone to overfitting.

7.3 Effect of the Classifiers

Table 3 shows that, in almost all cases, the models with the SVM classifier significantly outperform the ones with the neural classifier. We believe that, although the neural network may have been able to reach similar results as the SVM, the latter reached this performance using less fine-tuning due to its explicit design to optimize the margin size between classes.

On another note, Figure 2 shows that the models with the SVM classifier were significantly more robust and demonstrated much less performance fluctuation during training than the ones with the neural classifier. We believe that this is due to the more deterministic nature of SVM in comparison to the neural networks.

However, the most important drawback of the SVM was the training time: since the SVM classifier was trained separately from feature extractors, training it entailed additional training time to the model. All being said, we believe that, if training

Model	Input Feature	F1 Score on Development Data				F1 Score on Test Data			
		happy	*angry*	*sad*	**Micro**	*happy*	*angry*	*sad*	**Micro**
BASELINE		–	–	–	58.61	–	–	–	58.68
LSTM+NN	GloVe	53.45	67.37	59.39	60.40	52.05	67.02	52.89	57.60
	Glove+POS	58.33	68.28	62.54	63.15	58.03	68.68	59.77	62.35
	ELMo	63.88	67.61	69.14	66.74	62.07	65.14	67.37	64.74
	ELMo+POS	59.03	65.92	68.42	64.13	63.95	62.06	68.56	64.54
GRU+NN	GloVe	47.79	65.67	66.19	59.36	51.34	65.80	58.04	58.26
	Glove+POS	57.94	68.39	64.74	63.77	61.24	69.27	57.82	62.99
	ELMo	64.26	67.58	74.71	68.34	63.32	66.21	69.73	66.33
	ELMo+POS	65.48	65.15	73.12	67.46	62.20	66.40	71.64	66.53
LSTM+SVM	GloVe	54.85	67.06	66.91	65.83	53.00	68.05	57.30	62.84
	Glove+POS	61.30	68.95	66.20	66.46	60.30	67.32	60.84	63.26
	ELMo	65.64	65.91	73.44	68.94	63.78	65.25	70.10	66.45
	ELMo+POS	62.31	68.44	68.73	67.25	63.37	64.09	68.86	67.28
GRU+SVM	GloVe	50.21	68.31	71.00	65.08	50.22	70.26	60.13	62.02
	Glove+POS	52.77	68.73	66.67	65.42	56.00	69.68	57.96	63.11
	ELMo	67.33	67.83	75.40	**73.03**	65.08	68.83	73.07	69.39
	ELMo+POS	68.21	66.26	74.46	70.03	64.71	69.13	71.05	**69.93**
AVERAGE		59.55	67.34	68.82	65.96	59.42	67.07	64.07	64.22

Table 3: The performance of each model on the development and test datasets, in terms of F1 score on each emotion class, and micro-average F1 over the three main emotion classes. The AVERAGE is computed over the proposed models and does not include the baseline.

time is not a concern, the SVM classifier is a better option than the neural one.

An interesting finding regarding the SVM classifier is that, in contrast to the neural network where applying class-weights to the loss function helped improve the performance of the models, applying class-weights to the SVM decreased its performance.

7.4 A Closer Look at Emotion Classes

The row labelled *AVERAGE* in Table 3 provide information regarding the difficulty of detecting each class. Table 3 shows that, among the three main classes, *happy*, *angry*, and *sad*, the class *happy* was the most difficult to detect.

Table 2 shows that the low average F1 score for class *happy* is probably due to the significantly smaller number of samples with this class in the training data (14%) in comparison to the samples from the other two emotion classes (18% and 18%). Although the weighted loss functions (see Section 5.4) somehow managed to handle the imbalanced class distribution in the data, the optimal weights are not necessarily proportional to the inverse of the frequency of classes.

7.5 Quality of the Extracted Neural Features

To better understand the contribution of the extracted neural features from the feature extractors, we calculated the mutual information between the values of each neural feature and the classes.

Figure 3 shows the average and the standard deviation of the mutual information between the features extracted from each neural feature extractor in each model and the classes in the training data. Since both the neural and the SVM classifiers use the same set of neural features, we did not differentiate between models with similar neural feature extractors and different classifiers.

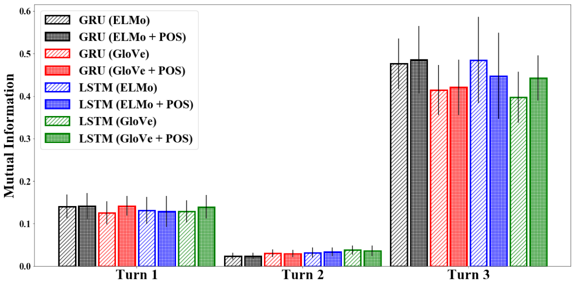


Figure 3: The average mutual information between the features of each dialogue turn, extracted by each neural feature extractor, and the classes.

As Figure 3 shows, in all cases, the features extracted from the third turn of the conversation have the highest mutual information with the classes, and the ones from the second turn have the lowest. This agrees with the nature of the dataset, where the label is assigned to the emotion of speaker 1 (who uttered dialogue turns 1 and 3) after the third turn is uttered. This also indicates that the nature of emotion detection in the context of dialogues is different from that in monologues in two ways: not only do the utterances by different speakers contribute differently to the emotional state of a speaker, but also the timing of the utterances by the same speaker has an impact on the contribution of that utterance to the emotion classification.

Figure 3 shows that the difference in average mutual information between the extracted features and the emotional classes are higher for features from the third (i.e. the most recent) dialogue turn. As expected, the features from the ELMo-based models have significantly higher mutual information with the classes than the ones from the GloVe-based models. The features from the third dialogue turn in models with GRU have slightly higher mutual information than the ones from the models with LSTM. However, the standard deviation between the features extracted by the GRU-based models are significantly smaller than the ones with LSTM, showing that between the models with LSTM and the ones with GRU, the features extracted from the models with GRU had more similar amount of contribution to the classification task than the ones from the LSTM. This has led to the GRU-and-ELMo-based models outperforming the others.

A surprising finding is that the neural features extracted by the GRU-based models from the second dialogue turn have the least mutual information with the classes in comparison to the ones from the other models. Observing this, we experimented with our classifiers by disregarding the neural features from the second turn; however, this lead to a slight performance drop. We hypothesize that, although the neural features from the second dialogue turns bring only a small contribution to the classification, the GRU-based models tend to focus more on the features from the other two turns. This leads to the second feature extractor being less focused upon, and as a result, being less trained than the ones from other models.

8 Conclusion and Future Work

In this paper, we proposed a model for the task of contextual emotion detection. We evaluated our model with the EmoContext 2019 shared task dataset (Chatterjee et al., 2019b), which consists of 3 turn conversations tagged with one of the labels: *happy*, *sad*, *angry*, and *others*, based on the emotion present in the last dialogue turn.

The proposed model utilizes an attention-based recurrent neural network. We experimented with GloVe and ELMo embeddings, alongside POS tags as input, LSTM and GRU as recurrent units, and a neural or an SVM classifier. The best result on the test dataset was achieved with ELMo and POS tags as input, GRU as recurrent units, and SVM as the final classifier. Using this setup, we reached a performance of 69.93% in terms of micro-average F1 score which is a significant improvement over the baseline of 58.68%.

Three future directions can be proposed. The first is to investigate a more effective way of handling the imbalanced distribution of labels in the dataset. As a example, methods for finding the optimal class-weights for training the models can be investigated.

Secondly, the use of different number of features for each dialogue turn can be studied. As shown in Figure 3, features extracted from different dialogue turns had different levels of contribution to the final classification. In that case, more features could be extracted from turns 3 and 1 (uttered by the same speaker) in comparison to turn 2, which has the least contribution to the classification.

Lastly, knowing that the SVM classifier is capable of outperforming the neural one, studies can be performed in order to make the extracted features more suitable for the SVM classifier.

Acknowledgments

We would like to express our gratitude to Parsa Bagherzadeh for insightful discussions throughout the course of this work. We would also like to thank the anonymous reviewers for their comments on an earlier version of this paper.

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*. Vancouver, Canada, pages 718–728.

Malak Abdullah and Samira Shaikh. 2018. Teamuncc at SemEval-2018 task 1: Emotion detection in English and Arabic tweets using deep learning. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018)*. New Orleans, Louisiana, USA, pages 350–357.

Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. 2019a. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior* 93:309–317.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019b. SemEval-2019 task 3: EmoContext: Contextual emotion detection in text. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Minneapolis, Minnesota, USA.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*. Singapore.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar, pages 1724–1734.

Andre Cianflone, Yulan Feng, Jad Kabbara, and Jackie Chi Kit Cheung. 2018. Let's do it "again": A first computational approach to detecting adverbial presupposition triggers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*. Melbourne, Australia, pages 2747–2755.

Luca Dini and André Bittar. 2016. Emotion analysis on twitter: The hidden challenge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia.

Ruchi Hirat and Namita Mittal. 2015. A survey on emotion detection techniques using text in blogposts. *International Bulletin of Mathematical Research* 2(1):180–187.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Geoffrey Holmes, Andrew Donkin, and Ian H. Witten. 1994. Weka: a machine learning workbench. In *Proceedings of ANZIIS 1994 - Australian New Zealnd Intelligent Information Systems Conference*. Brisbane, Australia, pages 357–361.

Hamed Khanpour and Cornelia Caragea. 2018. Fine-grained emotion detection in health-related online posts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. Brussels, Belgium, pages 1160–1166.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computing Research Repository* arXiv:1412.6980.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*. Bejing, China, pages 1188–1196.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, Springer, pages 319–345.

Lan Li and Ji-hua Chen. 2006. Emotion recognition using physiological signals. In *Proceedings of 16th International Conference on Artificial Reality and Telexistence (ICAT 2006)*. Hangzhou, China, pages 437–446.

Jasy Suet Yan Liew and Howard R. Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*. San Diego, California, USA, pages 73–80.

Qi-rong Mao, Xin-yu Pan, Yong-zhao Zhan, and Xiang-jun Shen. 2015. Using kinect for real-time emotion recognition via facial expressions. *Frontiers of Information Technology & Electronic Engineering* 16(4):272–282.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5(4):1093–1113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop Proceedings of the International Conference on Learning Representations (ICLR 2013)*. Scottsdale, Arizona, USA.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*. Vancouver, Canada, pages 65–77.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval 2018)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 1–17.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *Computing Research Repository* arXiv:1211.5063v1.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS 2017 Autodiff Workshop*. Long Beach, California, USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018)*. New Orleans, Louisiana, USA, pages 2227–2237.

Soujanya Poria, Erik Cambria, Newton Howard, Guang-Bin Huang, and Amir Hussain. 2016. Fusing audio, visual and textual clues for sentiment analysis from multimodal content. *Neurocomputing* 174(PA):50–59.

George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? End-to-end speech emotion recognition using a deep convolutional recurrent network. In *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*. Shanghai, China, pages 5200–5204.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Curran Associates, Inc., Long Beach, California, USA, pages 5998–6008.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. Austin, Texas, USA, pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*. San Diego, California, USA, pages 1480–1489.

Mohamed Yassine and Hazem Hajj. 2010. A framework for emotion mining from text in online social networks. In *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*. Sydney, Australia, pages 1136–1142.

Yong Zhang, Xiaomin Ji, and Suhua Zhang. 2016. An approach to EEG-based emotion recognition using combined feature extraction method. *Neuroscience Letters* 633:152–157.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. Berlin, Germany, pages 207–212.

Empirical Study of Diachronic Word Embeddings for Scarce Data

Syrielle Montariol
LIMSI - CNRS
Univ. Paris-Sud, Univ. Paris-Saclay
Société Générale
syrielle.montariol@limsi.fr

Alexandre Allauzen
LIMSI - CNRS
Univ. Paris-Sud, Univ. Paris-Saclay
alexandre.allauzen@limsi.fr

Abstract

Word meaning change can be inferred from drifts of time-varying word embeddings. However, temporal data may be too sparse to build robust word embeddings and to discriminate significant drifts from noise. In this paper, we compare three models to learn diachronic word embeddings on scarce data: incremental updating of a Skip-Gram from Kim et al. (2014), dynamic filtering from Bamler and Mandt (2017), and dynamic Bernoulli embeddings from Rudolph and Blei (2018). In particular, we study the performance of different initialisation schemes and emphasise what characteristics of each model are more suitable to data scarcity, relying on the distribution of detected drifts. Finally, we regularise the loss of these models to better adapt to scarce data.

1 Introduction

In all languages, word usage can evolve over time, mirroring cultural or technological evolution of society (Aitchison, 2001).

For example, the word "Katrina" used to be exclusively a first name until year 2005 when hurricane Katrina devastated the United States coasts. After that tragedy, this word started to be associated with the vocabulary of natural disasters.

In linguistics, *diachrony* refers to the study of temporal variations in the use and meaning of a word. Detecting and understanding these changes can be useful for linguistic research, but also for many tasks of Natural Language Processing (NLP). Nowadays, a growing number of historical textual data is digitised and made publicly available. It can be analysed in parallel with contemporary documents, for tasks ranging from text classification to information retrieval. However, the use of conventional word embeddings methods have the drawback to average in one vector the different word's usages observed across the whole corpus. This *static* representation hypothesis turns out to be limited in the case of temporal datasets.

Assuming that a change in the context of a word mirrors a change in its meaning or usage, a solution is to explore diachronic word embeddings: word vectors varying through time, following the changes in the global context of the word. While many authors proposed diachronic embedding models these last years, these methods usually need large amounts of data to ensure robustness.

However, temporal datasets often face the problem of scarcity; beyond the usual scarcity problem of domain-specific corpora or low-resource languages, a temporal dataset can have too short compared to the volume of the full dataset.[1] Moreover the amount of digital historical texts is limited for many languages, particularly for oldest time periods.

This paper addresses the following question: In case of scarce data, how to efficiently learn time-varying word embeddings? For this purpose, we compare three diachronic methods on several sizes of datasets. The first method is incremental updating (Kim et al., 2014), where word vectors of one time step are initialised using the vectors of the previous time step. The second one is the dynamic filtering algorithm (Bamler and Mandt, 2017) where the evolution of the embeddings from one time step to another is controlled using a Gaussian diffusion process. Finally, we experiment dynamic Bernoulli embeddings (Rudolph and Blei, 2018) where the vectors are jointly trained on all time slices.

These three models are briefly described in section 3. The hyper-parameters are specifically tuned towards efficiency on small datasets. Then, we explore the impact of different initialisation

[1] A short period can be one month or less, depending on the domain.

scheme and compare the behaviour of word drifts exhibited by the models. Finally, we experiment regularising the models in order to tackle the faults detected in the previous analysis. The experiments in section 4 are made on the *New York Times Annotated Corpus* (NYT) [2] (Sandhaus, 2008) covering two decades.

2 Related Work

The first methods to measure semantic evolution rely on detecting changes in word co-occurrences, and approaches bases on distributional similarity (Gulordava and Baroni, 2011) . The use of automated learning methods, based on word embeddings (Mikolov et al., 2013), is recent and has undergone a increase in interest these last two years with the successive publication of three articles dedicated to a literature review of the domain (Kutuzov et al., 2018; Tahmasebi et al., 2018; Tang, 2018). In this section, we mainly consider this second line of work, along with the peculiarities of scarce data.

Kim et al. (2014) developed one of the first method to learn time-varying word sparse representations. It consists in learning an embedding matrix for the first time slice t_0, then updating it at each time step t using the matrix at $t-1$ as initialisation. This method is called incremental updating . Another broadly used method it to learn an embedding matrix for each time slice independently; due to the stochastic aspect of word embeddings, the vectorial space for each time slice is different, making them not directly comparable. Thus, authors perform an alignment of the embeddings spaces by optimising a geometric transformation (Hamilton et al., 2016; Dubossarsky et al., 2017; Szymanski, 2017; Kulkarni et al., 2015)).

In the case of sparse data, in addition to the approximative aspect of the alignment that harms the robustness of the embeddings, these methods are sensitive to random noise, which is difficult to disambiguate from semantic drifts. Moreover, the second one require large amounts of data for each time step to prevent overfitting. Tahmasebi (2018) shows that low-frequency words have a much lower temporal stability than high-frequency ones. In (Tahmasebi et al., 2018), the authors explain that usual methods for diachronic embeddings training such as the two previously presented are inefficient in the case of low-frequency words and hypothesise that a new set of methods, pooled under the name of *dynamic* models, may be more adapted. These models use probabilistic models to learn time-varying word embeddings while controlling the drift of the word vectors using a Gaussian diffusion process. Bamler and Mandt (2017) uses Bayesian word embeddings, which makes the algorithm more robust to noise in the case of sparse data; while Rudolph and Blei (2018) relies on a Bernoulli distribution to learn the dynamic embeddings jointly across all time slices, making the most of the full dataset.

Outside of the framework of diachrony, several attempts aim at improving or adapting word embeddings to low-volume corpora in the literature. It can involve morphological information (Luong et al., 2013) derived from the character level (Santos and Zadrozny, 2014; Labeau et al., 2015), and often make use of external resources: semantic lexicon (Faruqui et al., 2015), and pre-trained embeddings from larger corpora (Komiya and Shinnou, 2018). However, to our knowledge, no work has attempted to apply similar solutions to the problem of sparse data in temporal corpora, even thought this situation has been faced by many authors, often in the case of short time steps for social media data (Stewart et al., 2017; Bamler and Mandt, 2017; Kulkarni et al., 2015).

3 Diachronic Models

This section briefly describes the three models under study: the Skip-Gram incremental updating algorithm from Kim et al. (2014), the dynamic filtering algorithm of Bamler and Mandt (2017), and the dynamic Bernoulli embeddings model from Rudolph and Blei (2018). We consider a corpus divided into T time slices indiced by t. For each time step t, every word i is associated with two vectors u_{it} (word vector) and v_{it} (context vector).

3.1 Incremental Skip-Gram (ISG)

This algorithm relies on the skip-gram model estimated with negative sampling (SGNS) method described in (Mikolov et al., 2013) and it can be summarised as follows. The probability of a word i to appear in the context of a word j is defined by $\sigma(u_{i,t}^T v_{j,t})$, with σ being the sigmoid function. Words i and j are represented by their embedding

[2] https://catalog.ldc.upenn.edu/LDC2008T19

vectors $u_{i,t}$ and $v_{j,t}$) at time t. The matrices U_t and V_t gathers all of them for the whole vocabulary. The context is made of a fixed number of surrounding words and each word in the context are considered as independent of each other.

The negative sampling strategy associates to each observed word-context pair (the positive examples) n^+_{ijt}, a set of negative examples n^-_{ijt}. The negative examples are sampled for a noise distribution following Mikolov et al. (2013).

Let n^{+-}_t denote for the time step t, the union of positive and negative examples. The objective function can be defined as the following log-likelihood:

$$\begin{aligned} log\, p(n^{+-}_t|U_t, V_t) &= \mathcal{L}_{pos}(U_t, V_t) + \mathcal{L}_{neg}(U_t, V_t) \\ &= \sum_{i,j=1}^{L} (n^+_{ijt} log\, \sigma(u^T_{i,t} v_{j,t}) + n^-_{ijt} log\, \sigma(-u^T_{i,t} v_{j,t})) \end{aligned} \tag{1}$$

For the first time slice, the matrices U_1 and V_1 are initialised using a Gaussian random noise $\mathcal{N}(0, 1)$ before being trained according to equation (1). Then, for each successive time slice, the embeddings are initialised with values of the previous time slice following the methodology of (Kim et al., 2014). This way, the word vectors of each time step are all in the same vectorial space and directly comparable.

3.2 Dynamic Filtering of Skip-Gram (DSG)

This second method relies on the Bayesian extension of the SGNS model described by Barkan (2015). The main idea is to share information from one time step to another, allowing the embeddings to drift under the control of a diffusion process. A full description of this approach, denoted as the filtering model, can be found in (Bamler and Mandt, 2017).

In this model, the vectors $u_{i,t}$ and $v_{i,t}$ are considered as latent vectors. Under a Gaussian assumption, they are represented by their means $(\mu_{u_{i,t}}, \mu_{v_{i,t}})$ and variances $(\Sigma_{u_{i,t}}, \Sigma_{v_{i,t}})$. They are initialised for the first time slice with respectively a zero mean vector and a identity variance matrix.

The temporal drift from one time step to another follows a Gaussian diffusion process with zero mean and variance D. This variance is called the *diffusion* constant and has to be tuned along with the other hyperparameters. Moreover, at each time step a second Gaussian prior with zero mean and variance D_0 is added, resulting in the following distributions over the embeddings matrices U_t:

$$\begin{aligned} p(U_1|U_0) &\sim \mathcal{N}(0, D_0) \\ p(U_t|U_{t-1}) &\sim \mathcal{N}(U_{t-1}, D)\, \mathcal{N}(0, D_0). \end{aligned} \tag{2}$$

The same equation stands for V_t. Training this model requires to estimate the posterior distributions over U_t and V_t given n^{+-}_t. This (Bayesian) inference step is unfortunately untractable. In (Bamler and Mandt, 2017), the authors propose to use variational inference (Jordan et al., 1999) in its online extension (Blei et al., 2017). The principle of variational inference is to approximate the posterior distribution with a simpler variational distribution $q_\lambda(U, V)$ (λ gathers all the parameters of q). This variational posterior will be iteratively updated at each time step. The final objective function can be written as follows:

$$\begin{aligned} \mathcal{L}_t(\lambda) = E_{q_\lambda}&[log\, p(n^{+-}_t|U_t, V_t)] \\ &+ E_{q_\lambda}[log\, p(U_t, V_t)|n^{+-}_{1:t-1}] \\ &- E_{q_\lambda}[log\, q_\lambda(U_t, V_t)]. \end{aligned} \tag{3}$$

This loss function is the sum of three terms: the log-likelihood (computed following equation (1)), the log-prior (which enforces the smooth drift of embedding vectors, sharing information with the previous time step), and the entropy term (approximated as the sum of the variances of the embedding vectors).

3.3 Dynamic Bernoulli Embeddings (DBE)

The DBE models extends the *Exponential Family Embeddings* (EFE)(Rudolph et al., 2016), a probabilistic generalisation of the *Continuous Bag-of-Words* (CBOW) model of Mikolov et al. (2013). The main idea is that the model predicts the central word vector conditionally to its context vector following a Bernoulli distribution. A detailed description of the model can be found in (Rudolph and Blei, 2018).

Each word i has T different embeddings vectors u_{it}, but this time, the context vectors v_i are assumed to be fixed across the whole corpus. The embedding vector u_{it} drifts throughout time following a Gaussian random walk, very similarly to equation (2):

$$\begin{aligned} U_0,\ V &\sim \mathcal{N}(0, \lambda_0^{-1} I), \\ U_t &\sim \mathcal{N}(U_{t-1}, \lambda^{-1} I). \end{aligned} \tag{4}$$

The *drift* hyper-parameter λ controls the temporal evolution of U_t, and is shared across all time steps. The training process, described more precisely by Rudolph and Blei (2018), relies on a variant of the negative sampling strategy described by Mikolov et al. (2013). The goal is to optimise the model across all time steps jointly, by summing over t the following loss function:

$$\mathcal{L}_t = \mathcal{L}_{pos}(U_t, V) + \mathcal{L}_{neg}(U_t, V) + \mathcal{L}_{prior}(U_t, V). \quad (5)$$

The two first terms are computed as in equation (1). The third term is defined as :

$$\mathcal{L}_{prior}(U_t, V) = -\frac{\lambda_0}{2}\sum_{i=1}^{L}\|v_i\|^2 - \frac{\lambda_0}{2}\sum_{i=1}^{L}\|u_{i,0}\|^2 - \frac{\lambda}{2}\sum_{i,t}\|u_{i,t} - u_{i,t-1}\|^2. \quad (6)$$

The role of $\mathcal{L}_{prior}$ is twofold: it acts as a regularisation term on V and U_t, and as a constraint on the drift of U_t, preventing it from going to far apart from U_{t-1}

4 Experimental Results

The goal of this study is to compare the behaviour of the three algorithms described in section 3 in case of low-volume corpora. We evaluate their predictive power on different volumes of data to compare the impact of two initialisation methods, and analyse the behaviour of the drift of the embeddings.

4.1 Experimental Setup

We use the *New York Times Annotated Corpus* (NYT) (Sandhaus, 2008) [3] containing around 1 855 000 articles ranging from January 1^{st} 1987 to June 19th 2007. We divide the corpus into $T = 20$ yearly time steps (the incomplete last year is not used in the analysis) and held out 10 % of each time step for validation and testing. Then, we sample several subsets of the corpus : 50 %, 10%, 5% and 1% of the training set. This way, we can compare the models on each subset to evaluate their ability to train a model in the case of low-volume corpora.

We remove stopwords and choose a vocabulary of $V = 10k$ most frequent words. Indeed, a small vocabulary is more adequate for sparse data in a temporal analysis in order to avoid having time steps were some word does not appear at all. The total number of words in the corpus after preprocessing is around 38.5 million. It amounts to around 200k words per time step in the 10 % subset of the corpus, thus only 20k in the 1 % subset.

To tune the hyperparameters, we use the log-likelihood of positive examples $\mathcal{L}_{pos}$ measured on the validation set. We train each model for 100 epochs, with a learning rate of 0.1, using the Adam optimiser. For the DSG model, we use a diffusion constant $D = 1$ and a prior variance $D_0 = 0.1$ for both corpora. For the DBE model, we use $\lambda = 1$ and $\lambda_0 = 0.01$.

We choose an embedding dimension $d = 100$, as the experiments show that a small embedding dimension, as in (Stewart et al., 2017), leads to smoother word drifts and makes the model less sensitive to noise when the data is scarce.

We use a context window of 4 words and a negative ratio of 1; we observed that having a higher number of negative samples artificially increased the held-out likelihood, but equalised the drifts of all the words in the corpus. Thus, in an extreme scarcity situation, each negative sample has a high weight during training: the number of negative samples has to be very carefully selected depending on the amount of data.

4.2 Impact of Initialisation on Sparse Data

The embedding vectors of the ISG and DBE models are initialised using a Gaussian white noise, while the means and variances of DSG are initialised with null vectors and identity matrices respectively. However, a good initialisation can greatly improve the quality of embeddings, particularly in the case of scarce data.

We experiment the impact of two types of initialisation on the log-likelihood of positive examples on the test set.

Internal initialisation:

We train each model in a static way on the full dataset. Then, we use the resulting vectors as initialisation for the first time step of the diachronic models. This methods is especially suitable for domain-specific corpora where no external comparable data is available.

[3] released by the Linguistic Data Consortium

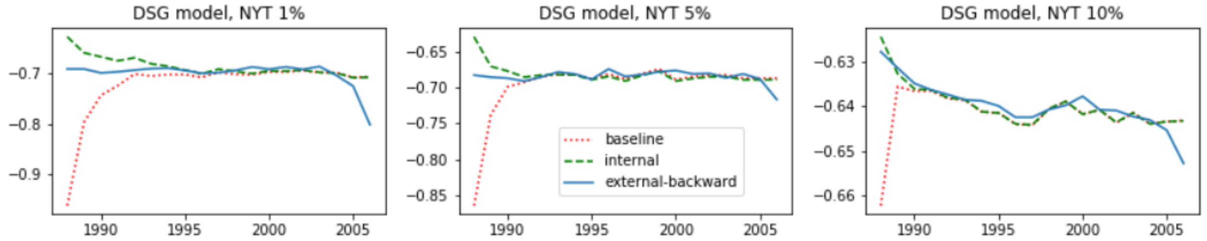


Figure 1: Log-likelihoods for the DSG model on three subsets of the corpus, comparing the baseline (random initialisation) with the two initialisation methods: *internal* is the initialisation from the full dataset while *external-backward* is the initialisation with the Wikipedia vectors, with training from most recent to oldest time step.

Backward external initialisation:
We use a set of embeddings pre-trained on a much larger corpus for initialisation : The Wikipedia corpus (dump of August 2013) (Li et al., 2017) with vectors of size 100. These embeddings are representative of the use of words in 2013; and in general, large corpora exist almost exclusively for recent periods. Thus, we choose to use the pre-trained embeddings as initialisation for the *last* time step (the most recent). Then, we update the embeddings incrementally from new to old (*reverse incremental updating*).
This method would be particularly suitable for corpora with low volume in older time slices, as it is the case for most of the historical dataset in languages other than English.
For the DSG model, the pre-trained vectors are used as the mean parameter for each word. The variance parameter is fixed at 0.1. Experiments with a prior variance of 0.01 and 1 had a lowest log-likelihood on the validation set.

The log-likelihood curves in figure 1 show that the internal initialisation has a better impact on the likelihood at the beginning of the period, as it is closer to the data than the external initialisation. The positive impact of the backward external initialisation increases with the volume of data.

Overall, the mean log-likelihoods across all time steps (Table 1) are higher using the internal initialisation. We conjecture that internal initialisation is more profitable to the model when the period is short (here, two decades) with low variance. The backward external initialisation has very close scores to the internal one, and is more suitable for higher volume datasets with a longer period, as it gives higher benefit to the likelihood for bigger subsets.

Initialisation / Model	**Random**	**Internal**	**Backward external**
ISG	-3.17	-2.589	-2.686
DSG	-0.749	-0.686	-0.695
DBE	-2.935	-2.236	-2.459

Table 1: Log-likelihood on the 5% subset of the NYT corpus for each model, with the three initialisation schemes.

4.3 Visualising Word Drifts

A high log-likelihood performance does not necessarily imply that the drifts detected by the models are meaningful. In this section, we examine the distribution of word drifts outputted by each model with the internal initialisation. The computed drift is the L2-norm of the difference between the embeddings at t_0 and the embeddings at each t:

$$drift(U_t) = \left[\sum_{i=1}^{L} (u_{i,t} - u_{i,t_0})^2 \right]^{1/2} \quad (7)$$

In the case of the DSG model were the words are represented as distributions, we compute the difference of the mean vectors.

We plot the superimposed histograms of *successive* drifts (Figure 2) from $t_0 = 1987$ to each successive time step, for all studied models. For example, on the histograms, the lightest colour curve represents the drift between $t_0 = 1987$ and $t = 2006$ and the darkest one is the drift between $t_0 = 1987$ and $t = 1988$.

A first crucial property is the *directed* aspect of the drifts: when the words progressively drift away from their initial representation in a directed fashion. On 10 % of the dataset, the DBE model shows well this behaviour, with a very clear colour gradient. It is also the case for the other models on this subset. With 1 % of the dataset on the contrary, the ISG model is unable to display a directed be-

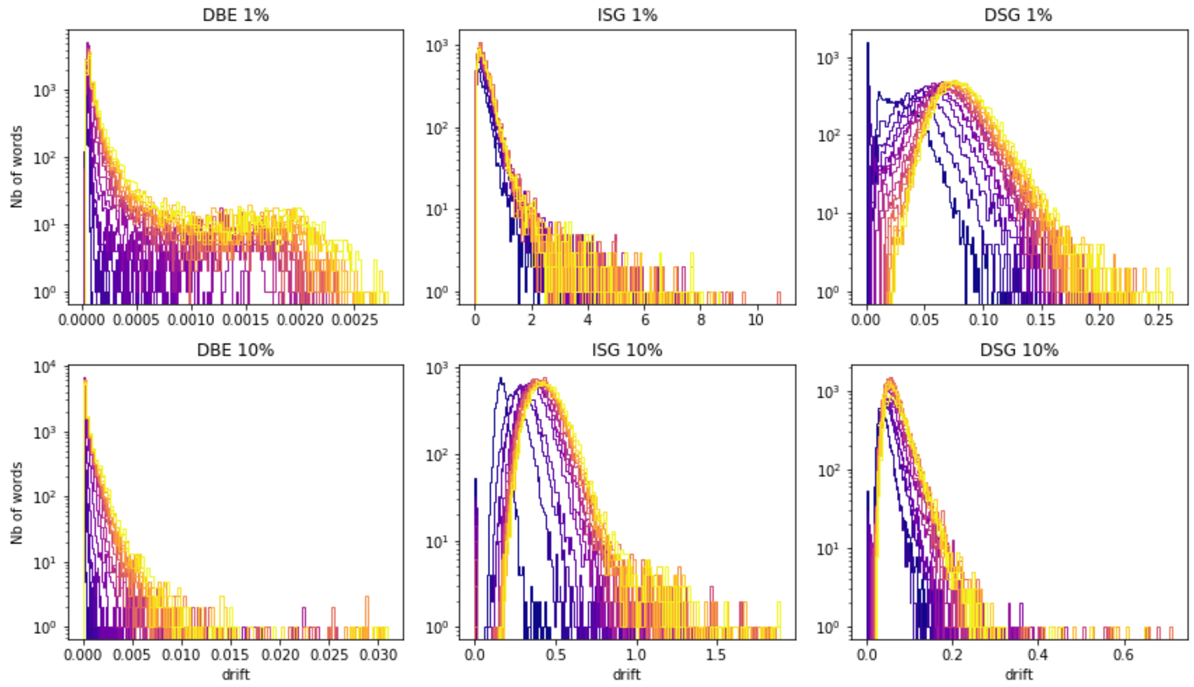


Figure 2: Histogram of word drift for each model on two subsets of the NYT corpus. The drifts are computed from $t_0 = 1987$ to each successive time step, and superposed on the histogram. The lightest colours indicate drifts calculated until the most recent time steps. The number of words are on logarithmic scale.

haviour (no colour gradient), while the two other models do. This is justified by the use of the diffusion process to link the time steps in equations 2 and 5: it allows the DSG and DBE models to emphasise the directed fashion of drifts even in the situation of scarce data.

The second property to highlight is the capacity of the models to discriminate words that drift from words that stay stable. From the human point of view, a majority of words has a stable meaning (Gulordava and Baroni, 2011); especially on a dataset covering only two decades like the NYT. The DBE model has a regularisation term (equation 6) to enforce this property, and a majority of words have a very low drift on the histogram. However, on 1 % of the dataset, this model can not discriminate very high drifts from the rest. The ISG and DSG models have a different distribution shape, with the peak having a drift superior to zero. The only words that do not drift on their histograms are the one that are absent from a time step.

To conclude, both the DBE and DSG model are able to detect directed drifts even in the 1 % subset of the NYT corpus, while the ISG can not. However, the drift distributions of the DBE and DSG models have a much shorter shorter tail on the 1 % subset than on the 10 % subset: they are not able to discriminate very high drifts from the rest of the words in extreme scarcity situation.

4.4 Regularisation Attempt

To tackle the weakness of the DBE and DSG models on the smallest subset, we attempt to regularise their loss in order to control the weight of the highest and lowest drifts. Our goal is to allow the model to:

- better discriminate very high drifts;
- be less sensitive to noise, giving lower weight to very low embedding drifts.

We test several possible regularisation terms to be added to the loss. The best result is obtained with the *Hardshrink* activation function, which is defined this way :

$$\begin{aligned} HardShrink(x) &= x, \text{ if } x > \beta \\ &= -x, \text{ if } x < -\beta \\ &= 0, \text{ otherwise} \end{aligned} \tag{8}$$

For the DSG and DBE models, we add to the loss the following regularisation term, amounting to a thresholding function applied to the drift:

$$\text{reg}_\beta = \alpha * HardShrink(\text{drift}(U_t), \beta) \tag{9}$$

Where α is the regularisation constant to be tuned, β is the threshold of the hardshrink function, and the drift is computed according to equation 7. The regularisation term is minimised. The activation function acts as a threshold to limit the amount of words having an important drift. We choose β as the mean drift for both models.

For both DSG and DBE, the right tail of the distribution of the drifts with regularisation (Figure 3) is much longer than in the original model (Figure 2). Moreover, in the case of the DSG model, more words have a drift very close to zero.

To conclude, the regularised DSG model considers more words as temporally stable. Furthermore, regularising the loss of the dynamic models allows to better discriminate extreme word embedding drifts for very small corpora.

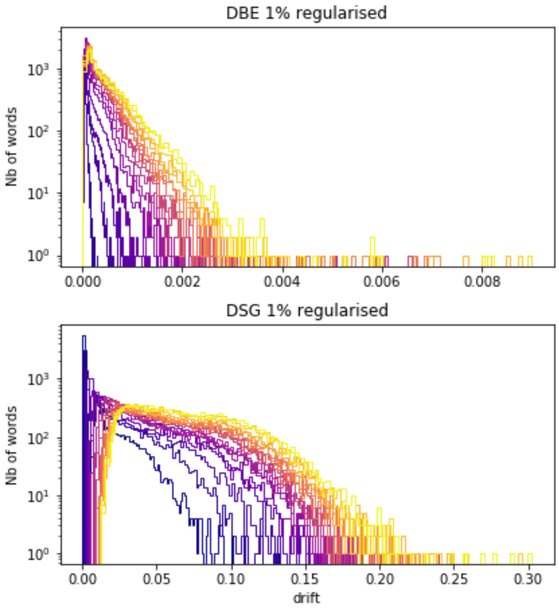


Figure 3: Histogram of word drift for the DBE and DSG regularised models on the 1 % subset.

5 Summary & Future Work

To summarise, we reviewed three algorithms for time-varying word embeddings: the incremental updating of a skip-gram with negative sampling (SGNS) from Kim et al. (2014) (ISG), the dynamic filtering applied to a Bayesian SGNS from Bamler and Mandt (2017) (DSG), and the dynamic Bernoulli embeddings model from Rudolph and Blei (2018) (DSG), a probabilistic version of the CBOW.

We proposed two initialisation schemes: the internal initialisation, more suited for low volume of data, and the backward external initialisation, more suited for higher volumes and long periods of temporal study. Then, we compared the distributions of the drifts of the models. We conclude that even in extreme scarcity situations, the DBE and DSG models can highlight directed drifts while the ISG model is too sensitive to noise. Moreover, the DBE model is the best at keeping a majority of the words stable. This property, as long as the ability to detect directed drift, are two important properties of a diachronic model. However, both have low ability to discriminate the highest drifts on a very small dataset. Thus, we added a regularisation term to their loss using the *Hardshrink* activation function, successfully getting longer distribution tails for the drifts.

An important future work is the multi-sense aspect of words. Polysemy is a crucial topic when dealing with diachronic word embeddings, as the change in usage of a word can reflect various changes in its meaning. However, the more different senses are taken into account, the more data is needed to tackle it. An evolution of the DSG model presented in this paper to adapt to this problem would be to represent words while taking into account the context of each occurrence of a word to disambiguate its meaning. Brazinskas et al. (2018) propose such model in a static fashion, where word vectors depends on the context and are drawn at token level from a word-specific prior distribution. The framework is similar to the Bayesian skip-gram model from Barkan (2015) used in the DSG model; but the goal is to predict a distribution of meanings given a context for each word occurrence. We are working on adapting this model to a dynamic framework.

References

Jean Aitchison. 2001. Language change: Progress or decay? In *Cambridge Approaches to Linguistics*. Cambridge University Press.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 380–389, International Convention Centre, Sydney, Australia. PMLR.

Oren Barkan. 2015. Bayesian neural word embedding.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2017. Variational inference: A review for statisticians. *CoRR*, abs/1601.00670.

Arthur Brazinskas, Serhii Havrylov, and Ivan Titov. 2018. Embedding words as distributions with a bayesian skip-gram model. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1775–1789. Association for Computational Linguistics.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1136–1145. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615. Association for Computational Linguistics.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the google books ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501. Association for Computational Linguistics.

Michael I. Jordan, Zoubin Ghahramani, and et al. 1999. An introduction to variational methods for graphical models. In *MACHINE LEARNING*, pages 183–233. MIT Press.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 61–65. Association for Computational Linguistics.

Kanako Komiya and Hiroyuki Shinnou. 2018. Investigating effective parameters for fine-tuning of word embeddings using only a small corpus. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 60–67, Melbourne. Association for Computational Linguistics.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 625–635, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397. Association for Computational Linguistics.

Matthieu Labeau, Kevin Löser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. pages 232–237, Lisbon, Portugal. Association for Computational Linguistics.

Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers, and Xiaoyong Du. 2017. Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2421.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1003–1011. International World Wide Web Conferences Steering Committee.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486.

Evan Sandhaus. 2008. The new york times annotated corpus. In *Philadelphia : Linguistic Data Consortium*. Vol. 6, No. 12.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings.

Ian Stewart, Dustin Arendt, Eric Bell, and Svitlana Volkova. 2017. Measuring, predicting and visualizing short-term change in word representation and usage in vkontakte social network. In *ICWSM*.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 448–453. Association for Computational Linguistics.

Nina Tahmasebi. 2018. A study on word2vec on a historical swedish newspaper corpus. In *DHN*.

Nina Tahmasebi, Lars Borin, and Adam Jatowt. 2018. Survey of computational approaches to diachronic conceptual change. *CoRR*, 1811.06278.

Xuri Tang. 2018. A state-of-the-art of semantic change computation. *Natural Language Engineering*, 24(5):649–676.

A Fast and Accurate Partially Deterministic Morphological Analysis

Hajime Morita **Tomoya Iwakura**
Fujitsu Laboratories Ltd., Kanagawa, Japan
{hmorita, iwakura.tomoya}@fujitsu.com

Abstract

This paper proposes a partially deterministic morphological analysis method for improved processing speed. Maximum matching is a fast deterministic method for morphological analysis. However, the method tends to decrease performance due to lack of consideration of contextual information. In order to use maximum matching safely, we propose the use of Context Independent Strings (CISs), which are strings that do not have ambiguity in terms of morphological analysis. Our method first identifies CISs in a sentence using maximum matching without contextual information, then analyzes the unprocessed part of the sentence using a bi-gram-based morphological analysis model. We evaluate the method on a Japanese morphological analysis task. The experimental results show a 30% reduction of running time while maintaining improved accuracy.

1 Introduction

Morphological analysis segments a text into a sequence of morphemes. The analysis is the first step of analyzing languages such as Chinese, Thai and Japanese. In Japanese, the analysis is a compound task of word segmentation, POS tagging, recognition of conjugation and lemmatization.

Japanese morphological analysis derives a word dictionary. The dictionary is a set of tuples of word surface (conjugated form), base form (lemma of words) and POS tags. Given an input string of a sentence, the morphological analyzer enumerates all substrings of the sentence that are listed in the dictionary. Then the analyzer builds a large lattice (DAG) of the words as a set of candidate word sequences. Because the lattice often becomes large, the lattice building tends to be a bottle neck of the morphological analysis. Finally, the analyzer selects a path on the lattice and gets the analysis result.

Morphological analysis is often used as a preprocessing step for shallow analyses on a large amount of texts collected from Web. These analyses include word counting, keyword extraction and named entity recognition. Morphological analysis tends to occupy a larger part of the processing time in the tasks. Therefore, we investigated a new morphological analysis method that improves efficiency and performance.

There are three kinds of Japanese morphological analysis methods in terms of the range of using contexts. The first group is maximum matching-based methods that do not use context (Sproat et al., 1996; Nagata, 1997; Sassano, 2014). Maximum matching is a significantly fast deterministic method for morphological analysis. However, this method tends to have a low performance, because it often fails an segmentation of phrases which have ambiguity depending on their context. The second group is neural network based methods that do not have explicit limit to the range of context (Morita et al., 2015; Chen et al., 2015; Wang et al., 2016; Kurita et al., 2017; Tolmachev et al., 2018). However, neural network based methods generally require heavy computational cost. The third group is commonly used local context based methods (Kudo et al., 2004; Neubig et al., 2011; Zheng et al., 2013; Kaji and Kitsuregawa, 2013; Kurohashi and Kawahara, 2014). The methods typically use bi-grams as their context. Among these methods, the bi-gram based methods are faster than neural network based methods and their performance is not far from that of neural network based methods.

This paper proposes a partially deterministic morphological analysis method for improved processing speed. We improve processing speed by incorporating maximum matching (the first group) to a bi-gram-based morphological engine (the third group).

In order to alleviate the performance loss of maximum matching, we propose a concept of Context Independent Strings (CISs), which are strings having no ambiguity in terms of morphological analysis. We also propose an algorithm for the building of the CISs dictionary from the large amount of automatically analysed texts. The dictionary maps CISs to the results of morphological analysis (sequence of words and POS tags).

Our method first applies maximum matching to a sentence using the CIS dictionary. CISs in a sentence are deterministically analyzed by the dictionary. Then the method analyze the rest part of the sentence using a bi-gram-based morphological analysis. We can use high-performance and computationally heavy methods such as neural network based methods for the building of the dictionary. That is, we can partially use their analysis results without recomputing through the CIS dictionary.

We obtain CISs from automatically parsed large texts with a state-of-the-art morphological analyzer JUMAN++ (Morita et al., 2015; Tolmachev et al., 2018). We evaluate this method on a Japanese morphological analysis task. The experimental results show a 30% reduction of running time while maintaining improved accuracy.

2 Context Independent Strings

The key idea of using maximum matching without performance loss is to separate character strings into two classes: context independent string (CIS) and context dependent string (CDS).

A CIS is a string that has a one-to-one correspondence with its grammatical analysis result, while a CDS has two or more grammatical analysis results that depend on its contexts. A CIS has to satisfy two conditions:

- The string does not have two or more grammatical analysis results.
- The analysis result is not affected by any strings adjacent to the beginning and end of the strings.

We describe CISs with the following three examples.

```
(A)  床の間
   (alcove, or gap of floor )
(B)  南京
   (Nanjing)
(C)  南       京都      病院
   (South) (Kyoto) (hopital)
```

(A) is a string that does not satisfy the first condition of CIS. “床の間” has two grammatical analysis result: “床の間 (alcove)” with one word or “床 (floor) + の (to) + 間 (gap)” with three words. The analysis depends on the context where the string occurred.

In (B), the string “南京” has only one grammatical analysis result “南京” (Nanjin). However, the string is a part of (C) “南京都病院”, and the string corresponts “南” (South) + the first character of “京都” (Kyoto). That violates the second condition of CIS.

When we segment the example (C) “南京都病院” (South Kyoto Hospital) using word-level maximum matching, the example is segmented into “南京+都+病院” (Nanjin metropolitan hospital) instead of correct segmentation “南+京都+病院” (South Kyoto hospital). This is because the longest word matched from the beginning of the example (C) is CDS (“南京”).

In fact, whole (C) is an example of a CIS. There is no “南京+都+病院” (Nanjin metropolitan hospital)“, and then the string has only one grammatical analysis result.

3 Proposed Method

This section describes the building method of a CIS dictionary, and deterministic morphological analysis using the CIS dictionary.

3.1 Building a CISs Dictionary

A CIS dictionary is a data structure where each CIS is associated to its morphological analysis result.

We built the CIS dictionary using the algorithm showed in **Algorithm 1**. In this method, the dictionary is built from a large automatically analyzed corpus C. We collected word N-grams n_j, their surfaces surface(n_j) and pointers j of the sentences in which the N-grams occurred, as a set of candidates (H, N, S).

In STEP (1) at the line 14 of Algorithm 1, when an identical surface was associated with two or

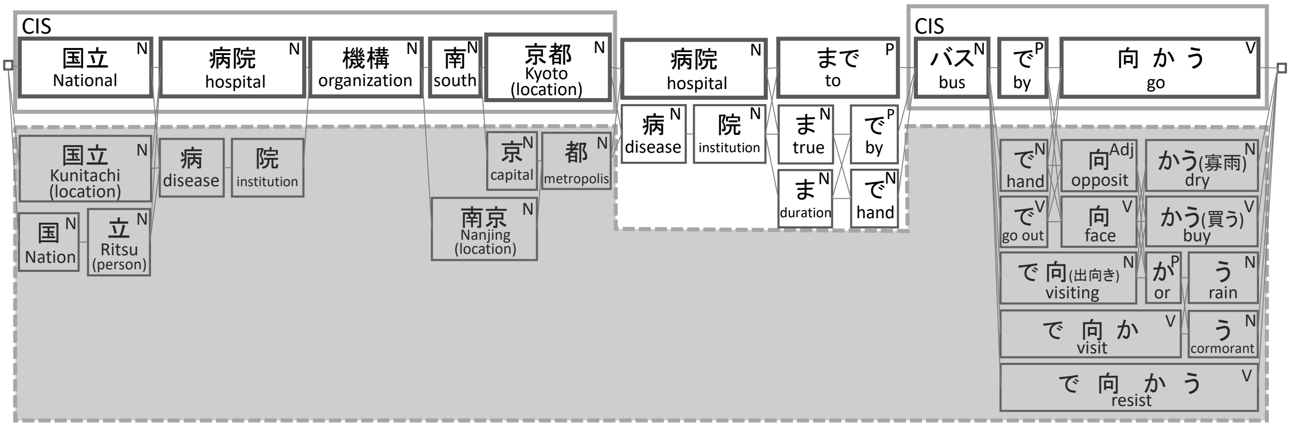


Figure 1: Example of a lattice that is built by our baseline method and our proposed model using a CIS dictionary. The CIS dictionary contains "国立病院機構南京都" and "バスで向かう", so the corresponding results are loaded from the dictionary. The word candidates surrounded by dotted line are not generated in our proposed model. Our implementation builds a CIS dictionary from word sequences of length 5 or less. Thus, even though "国立病院機構南京都病院" (6 words) is clearly a CIS, it is not in the dictionary.

more N-grams, or the N-grams occurred less frequently than a predetermined threshold T, the surfaces are discarded from the candidates. We aim to discard ambiguous strings similar to example (A).

In STEP (2) at the line 15, we remove CDSs that violate the second condition of CIS. As we can see from Example (B), the surface of CDS appears in a part of other word N-gram. The $\mathrm{findString}$ gives a set of sentences where the string p occurs in corpus C. The step checks there is no occurrence of the string p that appears in a part of another N-gram. We discarded any surface that occurred in a sentence in which the corresponding N-gram did not occur. This is because the occurrence indicates that the surface was a part of another different N-gram.

In STEP (3) at the line 15, the step is filtering using a heuristic. If the corpus is sufficiently large, the process considers any string before and after the N-gram. However, it is infeasible to cover all contexts of all expressions in a corpus of limited size. For the purpose of augmenting the corpus, we discarded the N-grams where words at the beginning or ending of N-gram are sensitive to preceding and succeeding strings. Specifically, if the beginning or end of the N-gram is a one-character word and is not in a predefined white list (punctuations, and a part of particles), the N-gram is discarded by this step. We assume that one-character words are often a suffix or prefix of other words and these words are sensitive to the surrounding contexts. We show the process as $\mathrm{isSensitive}$ function in Algorithm 1.

Finally, we build a CIS dictionary D using the collected CISs and their analysis result.

3.2 Analysis using a CIS Dictionary

There are three steps to analyse a sentence using a CIS dictionary: (1) Deterministic analysis by maximum matching with a CIS dictionary, (2) Building lattice by lookup dictionary and (3) Search for the least cost path on the lattice. We show our algorithm in Algorithm 2.

First, **LookUpLongestCIS** in Algorithm 2 finds CISs with the maximum match of a CIS dictionary from each character of the input sentence. If a CIS is found, the algorithm skips the range of matched string, and calls **LookUpLongestCIS** again. If it is not found, the step is restarted from the next character. The algorithm calls *LookUpDictionary* and builds a word lattice for each span where any analysis result is not obtained with the maximum matching.

Finally, whole sentence is connected as one lattice, and a word sequence with the lowest cost is obtained by Viterbi algorithm. On this lattice, the spans deterministically analyzed are expressed as nodes corresponding to the analysis result obtained from the dictionary, and their costs are 0. Other costs are calculated by a bi-gram model.

Figure 1 shows an example of a lattice that is built by our proposed method.

Algorithm 1 An algorithm for building a CIS dictionary

```
$T$ is a threshold for N-gram frequency,
$C \leftarrow \{s_0, \ldots, s_M\}$,
$S \leftarrow \{\}$
$H, R, D$ are associative arrays. $H$ maps surface to array of sentence ids, $R$ maps surface to array of analysis result, $D$ maps surface to an analysis result
for $i = 0$ to $N$ do
    for all $n_j \in s_i$ do
        $\text{insert}(H[\text{surface}(n_j)], i)$
        $\text{insert}(R[\text{surface}(n_j)], n_j)$
        $\text{insert}(S, \text{surface}(n_j))$
    end for
end for
for all $p \in S$ do
    # STEP (1)
    next if $|R[p]| > 1$ OR $|H[p]| < T$
    # STEP (2)
    next if $H[p] \neq \text{findString}(p, C)$
    # STEP (3)
    next if $\text{isSensitive}(R[p][0])$
    $D[p] \leftarrow (R[p][0])$
end for
return $D$
```

Algorithm 2 Morphological analysis using a CIS dictionary and Regular Expressions

```
S $\leftarrow \{c_0, \ldots, c_n\}$, $i \leftarrow 0$, $j \leftarrow 0$, $k \leftarrow 0$
$L$ = A lattice structure
while $i \leq N$ do
    result, length $\leftarrow$ LookUpLongestCIS(S, $i$)

    if result != $\phi$ then
        L[$i$] $\leftarrow$ result
        $i \leftarrow i$ + length
    else
        $i \leftarrow i$ + 1
    end if
end while
for $j = 0$ to $N$ –(2) do
    if $j$ is processed by LookUpLongestCIS then
        continue
    end if
    L[$j$] $\leftarrow$ LookUpDictionary(S,$j$)
end for
return Viterbi(L) –(3)
```

4 Experiments

4.1 Data and Models

We build a CIS dictionary using Mainichi News articles published from 1991 to 2010. The corpus contains approximately 2 million articles. We analyzed the corpus using JUMAN++ v1.02 (Morita et al., 2015). We set threshold T to 4, and limited the maximum N-gram length to 5.

A morphological dictionary used in our models is stemmed from JUMAN++ v1.02. We trained our feature parameters of CRF using L-BFGS (Kudo et al., 2004; Liu and Nocedal, 1989). We trained our models using Kyoto University Web Document Leads Corpus (Hangyo et al., 2012; Kawahara et al., 2014) that contains approximately 15,000 manually annotated sentences. The corpus contains manually annotated word segmentations, POS tags, dependencies, predicate-argument structures including zero anaphora, and so on. We used sentences with id (w201106-00000–w201106-00023) for training and sentences with id (w201106-00024) for development.

We evaluated the performance of our methods by KNB corpus (Hashimoto et al., 2011) that consists of 249 articles (4,186 sentences). The corpus contains manually annotatted word segmentation, POS tags and dependencies. Then, we evaluated the efficiency of our models by measuring running times of analysis on news articles of Yomiuri Shimbun in 2013. The text consists of approximately 300 thousand articles (approximately 4 million sentences).

We measured the performance of the methods by two performance measures Word and Word+POS. Word is the F-value of word segmentation, and Word+POS is the F-value of joint evaluation of word segmentation and POS tagging.

4.2 Performance Evaluation

We compared our proposed model with the baseline and JUMAN++. Partially deterministic analysis using a CIS dictionary did not lose to their performance but slightly improved in both Word and Word+POS. Our baseline is almost a reimplementation of MeCab (Kudo et al., 2004) and performs almost equally. JUMAN++ is an upper bound of our proposed model because a deterministically analyzed part of the analysis result is almost equivalent with the result of JUMAN++.

Table 1: Performance comparison of various systems.

Method	Word (F-val)	Word+POS (F-val)
Baseline	95.94	95.07
Partially deterministic (Proposed)	**95.99**	**95.17**
JUMAN++	96.84	96.15

Table 2: Comparing running times of various systems.

Method	Time (seconds)
Baseline	315.3
Partially deterministic (Proposed)	**223.4**
MeCab	124.5
JUMAN++	341705.7

4.3 Running Time Comparison

We compared our systems with publicly available implementations. MeCab is the fastest, but our implementation (Baseline) marks similar running time. Since our baseline model is not largely different from MeCab, the difference of running time is due to the implementation. Partially deterministic analysis reduces running time by 30%. If we implement our proposed method on MeCab or JUMAN++, we suspect it will improve their efficiency.

5 Conclusion

We presented a new fast partially deterministic morphological analysis method. We introduced the concept of Context Independent Strings and presented an algorithm for building a dictionary of CISs from a large corpus. We checked that the proposed method improved both efficiency and performance of morphological analysis. The method reduced the running time by 30% and improved the performance 0.1 pt in Word+POS.

References

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1197–1206.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*. Faculty of Computer Science, Universitas Indonesia, Bali,Indonesia, pages 535–544.

Chikara Hashimoto, Sadao Kurohashi, Keiji Shinzato, and Nagata Masaaki. 2011. Construction of a blog corpus with syntactic, anaphoric, and sentiment annotations (in japanese). *Journal of Natural Language Processing* 18(2):175–201.

Nobuhiro Kaji and Masaru Kitsuregawa. 2013. Efficient word lattice generation for joint word segmentation and pos tagging in japanese. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. pages 153–161.

Daisuke Kawahara, Yuichiro Machida, Tomohide Shibata, Sadao Kurohashi, Hayato Kobayashi, and Manabu Sassano. 2014. Rapid development of a corpus with discourse annotations using two-stage crowdsourcing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 269–278.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*. Association for Computational Linguistics, Barcelona, Spain, pages 230–237.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural joint model for transition-based chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL2017)*. Association for Computational Linguistics.

Sadao Kurohashi and Daisuke Kawahara. 2014. Juman ver.7.01. http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.* 45(3):503–528.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2292–2297.

Masaaki Nagata. 1997. A self-organizing japanese word segmenter using heuristic word identification and re-estimation. In *Proceedings of the 5th Workshop on Very Large Corpora*. pages 203–215.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. HLT '11, pages 529–533.

Manabu Sassano. 2014. Deterministic word segmentation using maximum matching with fully lexicalized rules. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. pages 79–83.

Richard Sproat, William Gale, Chilin Shih, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Comput. Linguist.* 22(3):377–404.

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, pages 54–59. https://doi.org/10.18653/v1/D18-2010.

Heng-Jun Wang, Nian-Wen Si, and Cheng Chen. 2016. An effective joint model for chinese word segmentation and pos tagging. In *Proceedings of the 2016 International Conference on Intelligent Information Processing*. ICIIP '16, pages 35:1–35:6.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 647–657.

`incom.py` – A Toolbox for Calculating Linguistic Distances and Asymmetries between Related Languages

Marius Mosbach[†], **Irina Stenger, Tania Avgustinova, Dietrich Klakow**[†]
Collaborative Research Center (SFB) 1102: Information Density and Linguistic Encoding
† Spoken Language Systems, Saarland Informatics Campus
Saarland University, Germany
ira.stenger@mx.uni-saarland.de
avgustinova@coli.uni-saarland.de
{mmosbach, dietrich.klakow}@lsv.uni-saarland.de

Abstract

Languages may be differently distant from each other and their mutual intelligibility may be asymmetric. In this paper we introduce `incom.py`, a toolbox for calculating linguistic distances and asymmetries between related languages. `incom.py` allows linguist experts to quickly and easily perform statistical analyses and compare those with experimental results. We demonstrate the efficacy of `incom.py` in an incomprehension experiment on two Slavic languages: Bulgarian and Russian. Using `incom.py` we were able to validate three methods to measure linguistic distances and asymmetries: Levenshtein distance, word adaptation surprisal, and conditional entropy as predictors of success in a reading intercomprehension experiment.

1 Introduction

1.1 Related Work

Linguistic phenomena may be language specific or shared between two or more languages. With regard to cross-lingual intelligibility, various constellations are possible. For example, speakers of language A may understand language B better than language C, i.e. $[A(B) > A(C)]$ while speakers of language B may understand language C better than language A, i.e. $[B(C) > B(A)]$. For instance, Ringbom (2007) distinguishes between *objective* (established as symmetrical) and *perceived* (not necessarily symmetrical) cross-linguistic similarities. Asymmetric intelligibility can be of linguistic nature. This may happen if language A has more complicated rules and/or irregular developments than language B, which results in structural asymmetry (Berruto, 2004).

Recently, in the INCOMSLAV framework (Fischer et al., 2015; Jágrová et al., 2016; Stenger et al., 2017a,b), measuring methods were developed that are of direct relevance for modelling cross-lingual asymmetric intelligibility. While it has been common to use (modifications of) the Levenshtein distance (Levenshtein, 1966) to predict phonetic and orthographic similarity (Beijering et al., 2008; Gooskens, 2007; Vanhove, 2016), this string-edit distance is completely symmetric. To account for asymmetric cross-lingual intelligibility Stenger et al. (2017b) employ additional measures of conditional entropy and surprisal (Shannon, 1948). Conditional character adaptation entropy and word adaptation surprisal, as proposed by Stenger et al. (2017b), quantify the difficulties humans encounter when mapping one orthographic system to another and reveal asymmetries depending on stimulus-decoder configurations in language pairs.

Similarly, the research of Jágrová et al. (2016) shows that Czech and Polish, both West Slavic, using the Latin script, are orthographically more distant from each other than Bulgarian and Russian, South and East Slavic respectively using the Cyrillic script. Both language pairs have similar lexical distances, however, the asymmetric conditional entropy based measures suggest that Czech readers should have more difficulties reading Polish text than vice versa. The asymmetry between Bulgarian and Russian is very small with a predicted minimal advantage for Russian readers (Stenger et al., 2017b). Additionally Stenger et al. (2017a) found that word-length normalized adaptation surprisal appears to be a better predictor than aggregated Levenshtein distance when the same stimuli sets in different language pairs are compared.

1.2 This paper

To calculate linguistic distances and asymmetries, perform statistical analyses and visualize the obtained results we developed the linguistic toolbox `incom.py`. The toolbox is validated on the Russian-Bulgarian language pair. We focus on word-based methods in which segments are compared at the orthographic level, since orthography is a linguistic determinant of mutual intelligibility which may facilitate or impede reading intercomprehension. We make the following contributions.

1. We provide implementations of various metrics for computing linguistic distances and asymmetries between languages.

2. We demonstrate the use of `incom.py` in an intercomprehension experiment for the Russian-Bulgarian language pair.

3. We show how `incom.py` can be used to validate word adaptation surprisal and conditional entropy as predictors for intercomprehension and discuss benefits over Levenshtein distance.

The remainder of this paper is structured as follows. The considered distance metrics implemented in the `incom.py` toolbox are introduced in Section 2. Section 3 describes the linguistic data used in the experiments. Section 4 presents the evaluation results of the statistical measures and compares them with the intelligibility scores obtained in a web-based cognates guessing. Finally, in Section 5, conclusions are drawn and future developments outlined.

2 Linguistic Distances and Asymmetries

2.1 Distance measures

We start with the introduction of basic notations and present the implemented distance measures.

2.1.1 Notation

Let L denote a language such as Russian or Bulgarian. Each language L has an associated alphabet – a set of characters – $\mathcal{A}(L)$ which includes the special symbol $\varnothing$[1]. We use $w \in L$ to denote a word in language L and $c_i \in w$ to denote the i-th character in word w. Note that while L is a set, w is not and may contain duplicates. Further, we assume the characters $c_i \in w$ are ordered with c_0 being the first and $c_{|w-1|}$ being the last character of word w, where the length $|w|$ of a word w is given by the number of characters it contains, including duplicates. Given two words w_i, w_j, the *alignment* of w_i and w_j results in two new words $\widetilde{w}_i, \widetilde{w}_j$ where $|\widetilde{w}_i| = |\widetilde{w}_j|$. We say character $s_k \in \widetilde{w}_i$ is aligned to character $t_l \in \widetilde{w}_j$ if $k = l$. That is, they occur at the same position.

2.1.2 Levenshtein distance

Levenshtein distance (LD) (Levenshtein, 1966) is, it its basic implementation, a symmetric similarity measure between two strings – in our case words – $w_i \in L_1$ and $w_j \in L_2$. Levenshtein distance quantifies the number of operations one has to perform in order to transform w_i into w_j. Levenshtein distance allows to measure the orthographic distance between two words and has been successfully used in previous works for measuring the linguistic distance between dialects (Heeringa et al., 2006) as well as the phonetic distance between Scandinavian language varieties (Gooskens, 2007). When computing Levenshtein distance between two words $\mathrm{LD}(w_i, w_j)$, three different character transformations are considered: character deletion, character insertion, and character substitution. In the following we use $\mathcal{T} = \{\text{insert}, \text{delete}, \text{substitute}\}$ to denote the set of possible transformations. A cost $c(t)$ is assigned to each transformation $t \in \mathcal{T}$ and setting $c(t) = 1\ \forall t \in \mathcal{T}$ results in the most simple implementation.

`incom.py` allows computing $\mathrm{LD}(w_i, w_j)$ based on a user-defined cost matrix $\mathbf{M}$, which contains the complete alphabets $\mathcal{A}(L_1), \mathcal{A}(L_2)$ of two languages L_1, L_2 as rows and columns, respectively, as well as the costs for every possible character substitution. That is, for two characters $s \in \mathcal{A}(L_1)$ and $t \in \mathcal{A}(L_2)$, $\mathbf{M}(s, t)$ is the cost of substituting s by t. This user defined cost matrix allows computing linguistically motivated alignments by incorporating a linguistic prior into the computation of the Levenshtein distance. For example, we assign a cost costs of 0 when mapping a character to itself. In case of $\mathbf{M}$ being symmetric, the Levenshtein distance remains symmetric. Along with the edit distance between the two words w_i and w_j our implementation of the Levenshtein distance returns the alignments $\widetilde{w}_i, \widetilde{w}_j$ of w_i and w_j, respectively. Given the length $K = |\widetilde{w}_i|$ of the alignment, we are fur-

[1] $\varnothing$ plays an important role when computing alignments. We will also refer to it as *nothing*

ther able to compute the normalized Levenshtein distance $\mathrm{nLD}(w_i, w_j) = \frac{\mathrm{LD}(w_i,w_j)}{K}$. For computing both the alignment and the resulting edit distance `incom.py` uses the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) following a dynamic-programming approach.

2.1.3 Word adaptation surprisal

Given two aligned words $\widetilde{w}_i$ and $\widetilde{w}_j$, we can compute the *Word Adaptation Surprisal* (WAS) between $\widetilde{w}_i$ and $\widetilde{w}_j$. Intuitively, word adaptation surprisal measures how confused a reader is when trying to translate w_i to w_j character by character. In order to define WAS formally, we introduce the notation of *Character Adaptation Surprisal* (CAS). Given a character $s \in \mathcal{A}(L_1)$ and another character $t \in \mathcal{A}(L_2)$, the character adaptation surprisal between s and t is defined as follows:

$$\mathrm{CAS}(s,t) = -\log_2(P(t \mid s)) \tag{1}$$

Now, the word adaptation surprisal between $\widetilde{w}_i \in L_1$ and $\widetilde{w}_j \in L_2$ can be computed straightforwardly by summing over all characters of the aligned word pair, i.e.

$$\mathrm{WAS}(\widetilde{w}_i, \widetilde{w}_j) = \sum_{k=0}^{K-1} \mathrm{CAS}(s_k, t_k) \tag{2}$$

where $K = |\widetilde{w}_i| = |\widetilde{w}_j|$. Similarly, the normalized word adaptation surprisal is computed as

$$\begin{aligned}\mathrm{nWAS}(\widetilde{w}_i, \widetilde{w}_j) &= \frac{1}{K}\sum_{k=0}^{K-1} \mathrm{CAS}(s_k, t_k) && (3)\\ &= \frac{1}{K}\mathrm{WAS}(\widetilde{w}_i, \widetilde{w}_j) && (4)\end{aligned}$$

Note that in contrast to Levenshtein distance, word adaptation surprisal is not symmetric.

Computing CAS (and hence also WAS) depends on the conditional probability $P(t|s)$, which is usually unknown. `incom.py` estimates $P(t|s)$ by $\hat{P}(t|s)$ which is based on corpus statistics. Given the alignments of a corpus $\mathcal{C}$ of word pairs produced by the Levenshtein algorithm, we compute $P(t|s)$ by counting the number of times t is aligned with s and divide over the total number of occurrences of character s, i.e.

$$\begin{aligned}\hat{P}(L_2 = t | L_1 = s) &= \frac{count(L_1 = s \wedge L_2 = t)}{count(L_1 = s)} && (5)\\ &\approx P(L_2 = t | L_1 = s) && (6)\end{aligned}$$

Certainly the quality of the estimate $\hat{P}(L_2 = t|L_1 = s)$ depends on the size of the corpus $\mathcal{C}$. In addition to the corpus based estimated character surprisals, `incom.py` provides functionality to modify the computed CAS values in a manual post-processing step. Based on this, the modified word adaptation surprisal can be computed as:

$$\mathrm{mWAS}(\widetilde{w}_i, \widetilde{w}_j) = \sum_{k=0}^{K-1} \mathrm{mCAS}(s_k, t_k) \tag{7}$$

where mCAS denotes the modified character adaptation surprisal. Similar to using a user defined cost matrix $\mathbf{M}$ when computing the Levenshtein distance, using modified character surprisal allows to incorporate linguistic priors into the computation of word adaptation surprisal.

2.1.4 Conditional entropy

Another asymmetric measure that is supported by our `incom.py` toolbox is *Conditional Entropy* (CE) (Shannon, 1948). Formally, the entropy of a discrete random variable X is defined as the weighted average of the surprisal values of this distribution. As discussed above, we can obtain the character surprisals based on the alignments obtained when computing the Levenshtein distance. Using these surprisal values we can compute the entropy of a language L as

$$H(L) = -\sum_{c \in L} P(L = c)\log_2 P(L = c) \tag{8}$$

In this work we are interested the entropy of a language L_1, e.g. Russian, that we compare to the entropy of another language L_2, e.g. Bulgarian. Thus we compute the conditional entropy between two languages L_1 and L_2.

$$\mathrm{CE}(L_1|L_2) = -\sum_{c_2 \in L_2} P(c_2)H(L_1|L_2 = c_2) \tag{9}$$

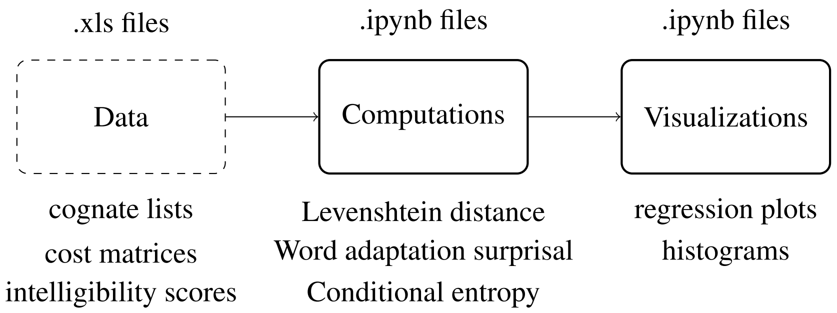


Figure 1: High-level overview of the `incom.py` toolbox.

Intuitively, CE$(L_1|L_2)$ measures the difficulty for a L_1 reader reading language L_2. Note that similar to the word adaptation surprisal, both entropy and conditional entropy highly depend on the number of available word pairs and only serve as an approximation to the true (unknown) entropy and conditional entropy, respectively.

2.2 `incom.py` toolbox

A high-level overview of the `imcom.py` toolbox is shown in Figure 1. The toolbox is a collection of jupyter notebooks based on the pandas and NumPy libraries[2]. To foster reproducibility and provide a resource for other researchers to easily compute linguistic distances and asymmetries we make `imcom.py` available online `https://github.com/uds-lsv/incompy`. In addition to computing distances and asymmetries based on a corpus of word pairs, `incom.py` readily supports visualizing the obtained results.

3 Data Sources

3.1 Language material

The Bulgarian (BG) and Russian (RU) data used in this work comes from a collection of parallel word lists consisting of internationalisms, Pan-Slavic vocabulary, and cognates from Swadesh lists. The words belong to different parts of speech, mainly nouns, adjectives, and verbs. We chose to use vocabulary lists instead of parallel sentences or texts in order to exclude the influence of other linguistic factors. The lists, each containing 120 words, were manually adjusted by removing non-cognates by possibly substituting them with etymologically related items, if such could be found, and adding further cognates[3]. Thus, for example, BG–RU ние–мы (nie–my) ‘we’ was removed and the BG звяр (zvjar) ‘beast’ instead of животно (životno) ‘animal’ was added to its RU cognate зверь (zver’) ‘animal, beast’. In a second step, a cross-linguistic rule set was designed taking into account diachronically motivated orthographic correspondences, e.g. BG–RU: б:бл, ж:жд, я:е, ла:оло etc. Following (Fischer et al., 2015) we apply the rule set to the parallel word lists in a computational transformation experiment and categorized all cognates in the respective pairs as either (i) identical, or (ii) successfully transformed, or (iii) non-transformable by this rule set. The stimuli selection for the online experiments (Section 3.2) is based on the successfully transformed ones: 128 items of a total of 935 (from all lists, excluding doublets). In this way we could exclude possible different derivational morphemes between related languages (e.g. BG–RU хладен–холодный (chladen–cholodnyj) ‘cold’) in order to focus on the impact of mismatched orthographic correspondences for cognate intelligibility. Even though it may seem artificial to test isolated words, the underlying assumption here is that correct cognate recognition is a precondition of success in reading intercomprehension. If the reader correctly recognizes a minimal proportion of words, he or she will be able to piece the written message together.

3.2 Web-based experiments

The orthographic intelligibility between BG and RU was tested in web-based experiments (`http://intercomprehension.coli.uni-saarland.de/en`) in which 71 native speakers of BG and 94 native speakers of RU took

[2] `https://jupyter.org`, `https://pandas.pydata.org`, `https://www.numpy.org`

[3] Shared inherited words from Proto-Slavic, shared loans, for example, internationalisms. Cognates are included in the definition; partial cognates are pairs of words which have the same meaning in both languages only in some contexts, for example, BG мъж (măž) ‘man, husband’ and RU муж (muž) ‘husband’.

		Stimuli	
		Bulgarian	Russian
Native	Bulgarian	–	74.67%
	Russian	71.33%	–

Table 1: Intercomprehension scores from free translation tasks performed by humans.

part. The participants started with registration and then completed a questionnaire in their native language. The challenges were presented: 2 with each 60 different BG stimuli in each group for RU speakers and 2 with 60 different RU stimuli in each group for BG speakers. The order of the stimuli were randomized. The participants saw the stimuli on their screen, one by one, and were given 10 seconds[4] to translate each word into RU or into BG. It was also possible to finish before the 10 seconds were over by either clicking on the 'next' button or pressing 'enter' on the keyboard. After 10 seconds the participants saw the next stimulus on their screen. During the experiment the participants received feedback in form of emoticons for their answers. The results were automatically categorized as 'correct' or 'wrong' via pattern matching with pre-defined answers: some stimuli had more than one possible translation and we also provided a list of so-called alternative correct answers. For example, the BG word път (păt) 'way' can be translated in RU as путь (put') or дорога (doroga), so both translations were counted as correct.

The analysis of the collected material[5] is based on the answers of 37 native speakers of Bulgarian (31 women and 6 men between 18 and 41 years of age, average 27 years) and 40 native speakers of Russian (32 women and 8 men between 18 and 71 years of age, average 33 years). The mean percentage of correctly translated items constitutes the intelligibility score of a given language (Table 1).

The results show that there is virtually no asymmetry in written intelligibility between BG and RU: the BG participants understand a slightly larger number of the RU words (74.67%) than the RU participants understand the BG words they are presented with (71.33%). This can be explained by the fact that there are only slight differences between the two languages on the graphic-orthographical level (for more details see (Stenger et al., 2017b)).

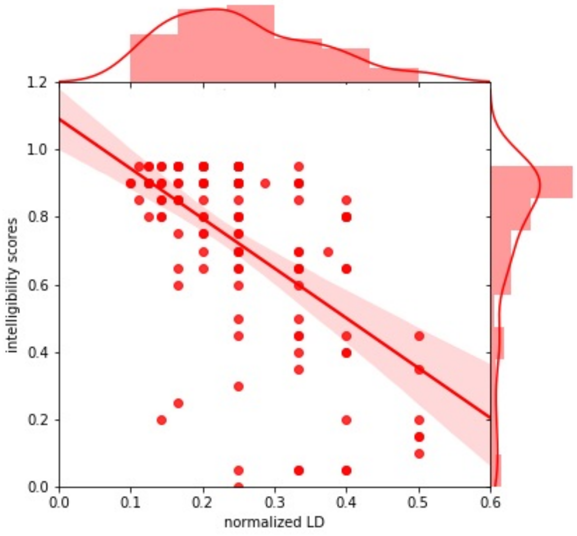
4 Results

4.1 Levenshtein distance and intelligibility score

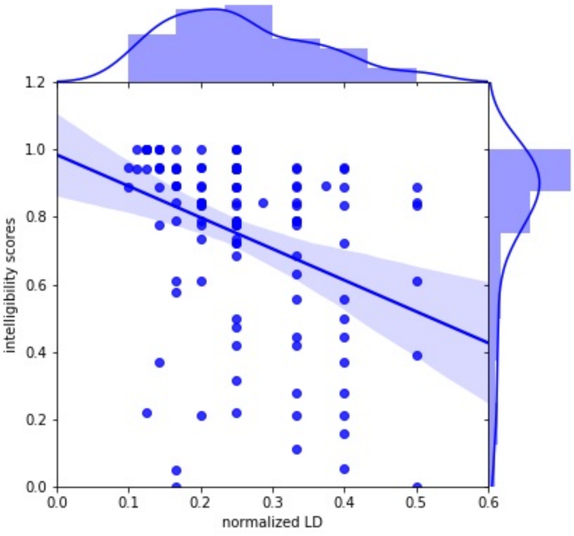
Using `incom.py` we compute the orthographic LD in both directions and further consider the normalized Levenshtein distance nLD between the 120 BG and RU cognates motivated by the assumption that a segmental difference in a word of two segments has a stronger impact on intelligibility than a segmental difference in a word of ten segments (Beijering et al., 2008; Stenger et al., 2017a). There is a general assumption that the higher the normalized LD, the more difficult it is to translate a given word (Gooskens, 2007; Vanhove and Berthele, 2015; Vanhove, 2016). Thus, we correlate the normalized LD and the intelligibility scores from our experiments for both language pairs. The correlation results are presented in Figure 2. We find a correlation between orthographic distance (normalized LD) and the intelligibility of BG words for RU readers of $r = –0.57$ ($p = 1.4e − 11$) and $r = –0.36$ ($p = 6.3e − 05$) for BG readers. Both correlations are significant and confirm the above hypothesis. However, the LD accounts for only 32% ($R^2 = 0.32$) of the variance in the intelligibility scores for RU readers and for only 13% ($R^2 = 0.13$) of the variance in the intelligibility scores for BG readers, leaving the majority of variance unexplained. Recall from Section 2 that LD is a symmetric measure, and therefore it does not capture any asymmetries between correspondences. If, for instance, the RU vowel character а always corresponds to а for a BG reader, but in the other direction, BG а can correspond to а, о or я for a RU reader, then a measure of linguistic distance is required to reflect both this difference in adaptation possibilities and the uncertainty involved in transforming а. Such asymmetries are effectively captured by the next two intelligibility measurements of word adaptation surprisal and conditional entropy, both of which are implemented in the `incom.py` tool-

[4] The time limit is chosen based on the experience from other reading intercomprehension experiments. The allocated time is supposed to be sufficient for typing even the longest words, but not long enough for using a dictionary or an online translation tool.

[5] For the present study we exclude those participants who have indicated knowledge of the stimuli language(s) in the questionnaire and analyze the results only of the initial challenge for each participant in order to avoid any learning effects.



(a) BG-RU



(b) RU-BG

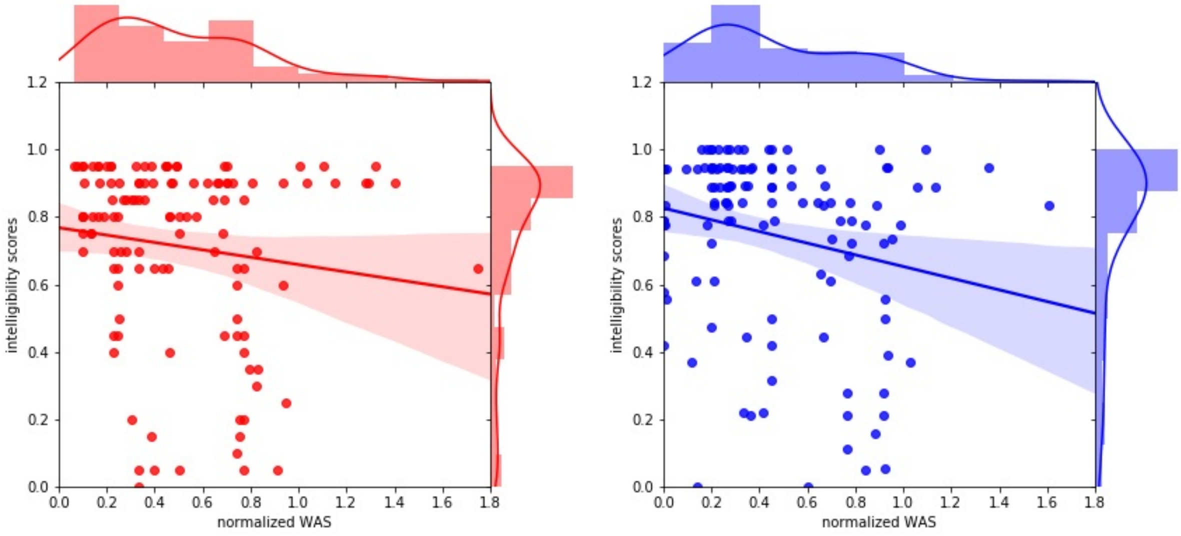
Figure 2: Normalized Levenshtein distance as a predictor for intelligibility. 2a Shows Russian native speakers reading Bulgarian. 2b Shows Bulgarian native speakers reading Russian.

box.

4.2 Word adaptation surprisal and intelligibility score

Word adaptation surprisal (WAS), in particular the normalized word adaptation surprisal (nWAS), helps us to predict and explain the effect of mismatched orthographic correspondences in cognate recognition. We assume that the smaller the normalized WAS, the easier it is to guess the cognate in an unknown, but (closely) related language. The correlation between the normalized WAS and the intelligibility scores is displayed in Figure 3. We find a low but significant negative correlation ($r = –0.22, p < 0.05$) between nWAS and written word intelligibility for BG readers. However, the negative correlation ($r = –0.13$) between nWAS and written word intelligibility for RU readers is not significant ($p = 0.14$). This can be explained by the fact that WAS values are given in bits and depend heavily on the probability distribution used.

Recall that with `incom.py`, we get the character adaptation surprisal (CAS) from our character adaptation probabilities (see Section 2 above). CAS and WAS values allow quantifying the unexpectedness both of individual character correspondences and of the whole cognate pair. This gives a quantification of the overall unexpectedness of the correct cognate. However, identical orthographic correspondences may still have a small surprisal value, for example, from a RU perspective the correspondence а:а has a surprisal value of 0.5986 bits resulting in an increase of the WAS value. Thus, we decided to manually modify our WAS calculation in such a way that all identical orthographic correspondences are measured with 0 bits. The calculated CAS values for mismatched orthographic correspondences remain unchanged in the modified calculation. Using the modified word adaption surprisal, we find a negative significant correlation between the modified nWAS and written word intelligibility also for RU readers ($r = –0.21$, $p < 0.05$). However, the modified nWAS accounts only for 12% ($R^2 = 0.123$) of the variance in the intelligibility scores for BG readers and the modified nWAS accounts for less than 5% ($R^2 = 0.044$) of the variance in the intelligibility scores for RU readers. This leaves the question why the correlation at the cognate level is so low. A possible explanation is that a cognate in an unknown closely related language will be easier to understand as it is more similar to the cognate in one's own language, because each cognate pair may have its own constellation of factors, affecting intelligibility, where one factor may overrule another factor, e.g., the number of orthographic neighbors in one's own language that



(a) BG-RU

(b) RU-BG

Figure 3: Normalized word adaptation surprisal as a predictor for intelligibility. 3a Shows Russian native speakers reading Bulgarian. 3b Shows Bulgarian native speakers reading Russian.

are very similar to the stimulus, the number of mismatched orthographic correspondences in the stimulus and their position, the word frequency in one's own language, the word length of stimulus etc. The estimated character values seem not to exactly reflect this constellation.

4.3 Conditional entropy and intelligibility score

For the BG–RU language pair the difference in the full conditional entropies (CE) is very small: 0.4853 bits for the BG to RU transformation and 0.4689 bits for the RU to BG transformation, with a very small amount of asymmetry of 0.0164 bits. These results predict that speakers of RU reading BG words are more uncertain than speakers of BG reading RU words. This is in accordance with the experimental results where the language combination with the slightly higher CE (RU speakers reading BG) had a slightly lower intelligibility score (see Table 1). Thus, CE can be a reliable measure when explaining even the small asymmetry in the mutual intelligibility.

Using `incom.py` we calculated entropy values of BG and RU characters in order to analyse asymmetries on the written (orthographic) level in more details. Figure 4 shows the entropy values of 6 BG characters е, ъ, а, щ, и, я, and the special symbol ∅ for RU readers and the entropy values of 5 RU characters о, е, я, у, л for BG readers (on the right). Note that the alignment of ∅ to any other character c corresponds to the case where Russian readers have to fill in a character. The entropy calculations reveal that, for example, BG readers should have more uncertainty with the RU vowel character о, while RU readers should have more difficulties with the adaptation of the BG vowel character е. This means that the mapping of the RU о to possible BG characters is more complex than the opposite direction. More precisely, the RU о can map into 4 BG vowel characters (о, а, ъ, е) or to *nothing* (∅), the BG е can map into 3 RU vowel characters (е, ё, or я). Certainly, in an intercomprehension scenario a BG or a RU reader does not know these mappings and the respective probabilities. However, the assumption is that the measure of complexity of the mapping can be used as an indicator for the degree of intelligibility (Moberg et al., 2007), because it reflects the difficulties with which a reader is confronted in 'guessing' the correct correspondence. Our experimental results indeed show that BG readers have greater problems with the RU о than RU readers with the BG character а or *nothing* (∅) in cognate pairs like RU–BG холод – хлад (cholod – chlad) 'cold', борода – брада (boroda – brada) 'beard', ворона – врана (vorona – vrana) 'crow'.

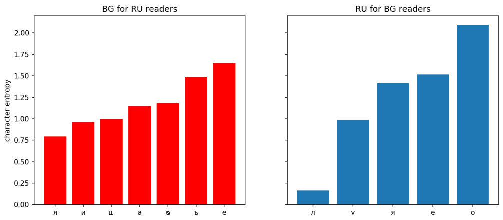


Figure 4: Character entropy values when translating from Russian to Bulgarian and vice versa.

5 Discussion and Outlook

Previous research in reading intercomprehension has shown that (closely) related languages may be differently distant from each other and their mutual intelligibility may be asymmetric. In this paper we present `incom.py` – a toolbox for computing linguistic distances and asymmetries. With `incom.py` we perform experiments on measuring and predicting the mutual intelligibility of Slavic languages, as exemplified by the language pair Bulgarian-Russian by means of the Levenshtein distance, word adaptation surprisal, and conditional entropy. Using a small corpus of parallel cognate lists we validated linguistic distances and asymmetries as predictors of mutual intelligibility based on stimuli obtained from written intelligibility tests. The results of our statistical analyses clearly support normalized Levenshtein distance as a reliable predictor of orthographic intelligibility at the word level for both language pairs tested. However, we find that only 32% (for RU readers) and 13% (for BG readers) of the variance in the intelligibility data is explained by the orthographic similarity quantified by means of the normalized Levenshtein distance. We find that the predictive power of the Levenshtein distance is different within the two language pairs. It must be mentioned here that the RU stimuli are in general longer (5.09 characters) than the BG stimuli (4.61 characters). Thus, the BG readers should intuitively delete more characters while the RU readers should add more characters in order to guess the correct cognate.

Previous research has shown that deletions and additions, the basic operations performed when computing Levenshtein distance, are not of equal value in the mutual intelligibility: it appears that deletions are more transparent for the participants in terms of subjective similarity than additions (Kaivapalu and Martin, 2017). This means that there is room for improvement in our orthographic distance algorithm. Word adaptation surprisal measures the complexity of a mapping, in particular, how predictable the particular correspondence in a language pair is. The surprisal values of correspondences are indeed different. However, they depend on their frequency and distribution in the particular cognate set. Most important and in contrast to Levenshtein distance, surprisal can be asymmetric. The character adaptation surprisal values between language A and language B are not necessarily the same as between language B and language A. This indicates an advantage of the surprisal-based method compared to Levenshtein distance. Our results show that the predictable potential of word adaptation surprisal was rather weak despite its modification. We assume that word adaptation surprisal should to a larger extent take into account relevant factors in reading intercomprehension, for example, orthographic neighbors (words that are very similar to the stimulus word and differ only in one character). Something we keep as future work.

Conditional entropy can reflect the difficulties humans encounter when mapping one orthographic system on another. The underlying hypothesis is that high predictability improves intelligibility, and therefore a low entropy value should correspond to a high intelligibility score. This result is as we expected. We have calculated conditional entropy for Bulgarian and Russian using a cognate word list from intelligibility tests. In our experiments, conditional entropy – like the intelligibility task – reveals asymmetry between Bulgarian and Russian on the orthographic level: the conditional entropy in Bulgarian for Russian readers is slightly higher than the conditional entropy in Russian for Bulgarian readers. This means that the slightly higher entropy is found in the language pair where there is slightly lower intelligibility. Thus, we were able to show that conditional entropy can be a reliable measure when explaining small asymmetries in intelligibility. In future work we plan to extend `incom.py` with additional functionality to compute distances and asymmetries on the phonological level. Additionally, it might be interesting to consider the morphological level which has been shown to be helpful when processing words for humans with limited reading abilities (Burani et al., 2008).

Acknowledgments

We would like to thank Klára Jágrová and Volha Petukhova for their helpful feedback on this paper. Furthermore, we thank the reviewers for their valuable comments. This work has been funded by Deutsche Forschungsgemeinschaft (DFG) under grant SFB 1102: Information Density and Linguistic Encoding.

References

Karin Beijering, Charlotte Gooskens, and Wilbert Heeringa. 2008. Predicting intelligibility and perceived linguistic distance by means of the Levenshtein algorithm. *Linguistics in the Netherlands* 25(1).

Gaetano Berruto. 2004. Sprachvarietät-sprache (Gesamtsprache, historische Sprache) .

Cristina Burani, Stefania Marcolini, Maria De Luca, and Pierluigi Zoccolotti. 2008. Morpheme-based reading aloud: Evidence from dyslexic and skilled italian readers. *Cognition* 108(1).

Andrea Fischer, Klára Jágrová, Irina Stenger, Tania Avgustinova, Dietrich Klakow, and Roland Marti. 2015. An orthography transformation experiment with Czech-Polish and Bulgarian-Russian parallel word sets. *Natural Language Processing and Cognitive Science 2015 Proceedings, Libreria Editrice Cafoscarina, Venezia* .

Charlotte Gooskens. 2007. The contribution of linguistic factors to the intelligibility of closely related languages. *Journal of multilingual and multicultural development* 28(6).

Wilbert Heeringa, Peter Kleiweg, Charlotte Gooskens, and John Nerbonne. 2006. Evaluation of string distance algorithms for dialectology. In *Proceedings of the workshop on linguistic distances*. Association for Computational Linguistics.

Klára Jágrová, Irina Stenger, Roland Marti, and Tania Avgustinova. 2016. Lexical and orthographic distances between Bulgarian, Czech, Polish, and Russian: A comparative analysis of the most frequent nouns. In *Language Use and Linguistic Structure: Proceedings of the Olomouc Linguistics Colloquium*.

Annekatrin Kaivapalu and Maisa Martin. 2017. Perceived similarity between written Estonian and Finnish: Strings of letters or morphological units? *Nordic Journal of Linguistics* 40(2).

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*. volume 10.

Jens Moberg, Charlotte Gooskens, John Nerbonne, and Nathan Vaillette. 2007. Conditional entropy measures intelligibility among related languages. *Proceedings of Computational Linguistics in the Netherlands* .

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48(3).

Håkan Ringbom. 2007. *Cross-linguistic similarity in foreign language learning*, volume 21. Multilingual Matters.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal* 27(3).

Irina Stenger, Tania Avgustinova, and Roland Marti. 2017a. Levenshtein distance and word adaptation surprisal as methods of measuring mutual intelligibility in reading comprehension of slavic languages. In *Computational Linguistics and Intellectual Technologies: International Conference 'Dialogue 2017'Proceedings*. volume 16.

Irina Stenger, Klára Jágrová, Andrea Fischer, Tania Avgustinova, Dietrich Klakow, and Roland Marti. 2017b. Modeling the impact of orthographic coding on Czech–Polish and Bulgarian–Russian reading intercomprehension. *Nordic Journal of Linguistics* 40(2).

Jan Vanhove. 2016. The early learning of interlingual correspondence rules in receptive multilingualism. *International Journal of Bilingualism* 20(5).

Jan Vanhove and Raphael Berthele. 2015. Item-related determinants of cognate guessing in multilinguals. *Crosslinguistic influence and crosslinguistic interaction in multilingual language learning* .

A Holistic Natural Language Generation Framework for the Semantic Web

Axel-Cyrille Ngonga Ngomo[1] **Diego Moussallem**[1] **Lorenz Bühmann**[2]
[1]Data Science Group, University of Paderborn, Germany
[2]AKSW Research Group, University of Leipzig, Germany
{first.lastname}@upb.de
lastname@informatik.uni-leipzig.de

Abstract

With the ever-growing generation of data for the Semantic Web comes an increasing demand for this data to be made available to non-semantic Web experts. One way of achieving this goal is to translate the languages of the Semantic Web into natural language. We present LD2NL, a framework for verbalizing the three key languages of the Semantic Web, i.e., RDF, OWL, and SPARQL. Our framework is based on a bottom-up approach to verbalization. We evaluated LD2NL in an open survey with 86 persons. Our results suggest that our framework can generate verbalizations that are close to natural languages and that can be easily understood by non-experts. Therewith, it enables non-domain experts to interpret Semantic Web data with more than 91% of the accuracy of domain experts.

1 Introduction

Natural Language Generation (NLG) is the process of automatically generating coherent Natural Language (NL) text from non-linguistic data (Reiter and Dale, 2000a). Recently, the field has seen an increased interest in the development of NLG systems focusing on verbalizing resources from Semantic Web (SW) data (Gardent et al., 2017). The SW aims to make information available on the Web easier to process for machines and humans. However, the languages underlying this vision, i.e., Resource Description Framework (RDF), SPARQL Query Language (SPARQL) and Web Ontology Language (OWL), are rather difficult to understand for non-expert users. For example, while the meaning of the OWL class expression `Class: Professor SubClassOf: worksAt SOME University` is obvious to every SW expert, this expression ("Every professor works at a university") is rather difficult to fathom for lay persons.

Previous works such as SPARQL2NL (Ngonga Ngomo et al., 2013) and SPARTIQULATION (Ell et al., 2012) have already shown the usefulness of the verbalization of SPARQL [1] and RDF in areas such as question answering (Lehmann et al., 2012) and the explanation of the output of systems based onSW technologies (Ngonga Ngomo et al., 2013). However, other SW languages are rarely investigated, such as OWL.

In this paper, we present an open-source holistic NLG framework for the SW, named LD2NL, which facilitates the verbalization of the three key languages of the SW, i.e., RDF, OWL, and SPARQL into NL. Our framework is based on a bottom-up paradigm for verbalizing SW data. Additionally, LD2NL builds upon *SPARQL2NL* as it is open-source and the paradigm it follows can be reused and ported to RDF and OWL. Thus, LD2NL is capable of generating either a single sentence or a summary of a given resource, rule, or query. To validate our framework, we evaluated LD2NL using experts 66 in Natural Language Processing (NLP) and SW as well as 20 non-experts who were lay users or non-users of SW. The results suggest that LD2NL generates texts which can be easily understood by humans. The version of LD2NL used in this paper, all experimental results will be publicly available.

2 Related Work

According to Gatt and Krahmer (2017), there has been a plenty of works which investigated the generation of NL texts from Semantic Web Technologies (SWT) as an input data (Cimiano et al., 2013;

[1] SPARQL is the query language for RDF data.

Duma and Klein, 2013; Ell and Harth, 2014; Biran and McKeown, 2015). However, the subject of research has only recently gained significant momentum due to the great number of published works in the WebNLG (Colin et al., 2016) challenge along with deep learning techniques (Sleimi and Gardent, 2016; Mrabet et al., 2016). RDF has also been showing promising benefits to the generation of benchmarks for evaluating NLG systems (Gardent et al., 2017; Perez-Beltrachini et al., 2016).

Despite the plethora of recent works written on handling RDF data, only a few have exploited the generation of NL from OWL and SPARQL. For instance, Androutsopoulos et al. (2013) generates sentences in English and Greek from OWL ontologies. Also, SPARQL2NL (Ngonga Ngomo et al., 2013) uses rules to verbalize atomic constructs and combine their verbalization into sentences. Therefore, our goal with LD2NL is to provide a complete framework to verbalize SW concepts rather than become the state of the art on the respective tasks.

3 Background

3.1 OWL

OWL[2] (OWL Working Group, 2009) is the de-facto standard for machine processable and interoperable ontologies on the SW. In its second version, OWL is equivalent to the description logic $\mathcal{SROIQ}(D)$. Such expressiveness has a higher computational cost but allows the development of interesting applications such as automated reasoning (Bühmann et al., 2016). OWL 2 ontologies consist of the following three different syntactic categories:

Entities, such as *classes*, *properties*, and *individuals*, are identified by IRIs. They form the primitive terms and constitute the basic elements of an ontology. Classes denote sets of individuals and properties link two individuals or an individual and a data value along a property. For example, a class `:Animal` can be used to represent the set of all animals. Similarly, the object property `:childOf` can be used to represent the parent-child relationship and the data property `:birthDate` assigns a particular birth date to an individual. Finally, the individual `:Alice` can be used to represent a particular person called ”Alice”.

Expressions represent complex notions in the domain being described. For example, a *class expression* describes a set of individuals in terms of the restrictions on the individuals' characteristics. OWL offers existential (**SOME**) or universal (**ONLY**) qualifiers and a variety of typical logical constructs, such as negation (**NOT**), other Boolean operators (**OR**, **AND**), and more constructs such as cardinality restriction (**MIN**, **MAX**, **EXACTLY**) and value restriction (**VALUE**), to create class expressions. Such constructs can be combined in arbitrarily complex class expressions `CE` according to the following grammar

```
CE = A | C AND D | C OR D | NOT C | R
    SOME C | R ONLY C | R MIN n | R MAX
     n | R EXACTLY n | R VALUE a | {a1
    ,...,am}
```

where `A` is an atomic class, `C` and `D` are class expressions, `R` is an object property, `a` as well as $\mathtt{a}_1$ to $\mathtt{a}_m$ with $\mathtt{m} \geq 1$ are individuals, and $\mathtt{n} \geq 0$ is an integer.

Axioms are statements that are asserted to be true in the domain being described. Usually, one distinguish between (1) *terminological* and (2) *assertional* axioms. (1) terminological axioms are used to describe the structure of the domain, i.e., the relationships between classes resp. class expressions. For example, using a subclass axiom (**SubClassOf:**), one can state that the class `:Koala` is a subclass of the class `:Animal`. Classes can be subclasses of other classes, thus creating a taxonomy. In addition, axioms can arrange properties in hierarchies (**SubPropertyOf:**) and can assign various characteristics (**Characteristics:**) such as transitivity or reflexivity to them. (2) Assertional axioms formulate facts about individuals, especially the classes they belong to and their mutual relationships. OWL can be expressed in various syntaxes with the most common computer readable syntax being RDF/XMLA more human-readable format is the Manchester OWL Syntax (MOS) (Horridge et al., 2006). For example, the class expression that models people who work at a university that is located in Spain could be as follows in MOS:

```
Person AND worksAt SOME (University AND
    locatedIn VALUE Spain)
```

Likewise, expressing that every professor works at a university would read as

```
Class: Professor
  SubClassOf: worksAt SOME University
```

[2] www.w3.org/TR/owl2-overview/

3.2 RDF

RDF (RDF Working Group, 2014) uses a graph-based data model for representing knowledge. Statements in RDF are expressed as so-called triples of the form `(subject, predicate, object)`. RDF subjects and predicates are Internationalized Resource Identifierss (IRIs) and objects are either IRIs or literals.[3] RDF literals always have a datatype that defines its possible values. A predicate denotes a *property* and can also be seen as a binary relation taking subject and object as arguments. For example, the following triple expresses that Albert Einstein was born in Ulm:

```
:Albert_Einstein :birthPlace :Ulm .
```

3.3 SPARQL

Commonly, the selection of subsets of RDF is performed using the SPARQL query language.[4] SPARQL can be used to express queries across diverse data sources. *Query forms* contain variables that appear in a solution result. They can be used to select all or a subset of the variables bound in a pattern match. They exist in four different instantiations, i.e., *SELECT*, *CONSTRUCT*, *ASK* and *DESCRIBE*. The *SELECT* query form is the most commonly used and is used to return rows of variable bindings. Therefore, we use this type of query in our explanation. *CONSTRUCT* allows to create a new RDF graph or modify the existing one through substituting variables in a graph templates for each solution. *ASK* returns a Boolean value indicating whether the graph contains a match or not. Finally, *DESCRIBE* is used to return all triples about the resources matching the query. For example, 1 represents the following query "Return all scientists who were born in Ulm".

```
SELECT ?person
WHERE {
    ?person a dbo:Scientist;
        dbo:birthPlace dbr:Ulm.
}
```

Listing 1: All scientists who were born in Ulm

4 LD2NL Framework

The goal of LD2NL is to provide an integrated system which generates a complete and correct NL representation for the most common used SW modeling languages RDF and OWL, and SPARQL. In terms of the standard model of NL generation proposed by Reiter & Dale (Reiter and Dale, 2000b), our steps mainly play the role of the micro-planner, with focus on aggregation, lexicalization, referring expressions and linguistic realization. In the following, we present our approach to formalizing NL sentences for each of the supported languages.

4.1 From RDF to NL

4.1.1 Lexicalization

The lexicalization of RDF triples must be able to deal with resources, classes, properties and literals.

Classes and resources The lexicalization of classes and resources is carried out as follows: Given a URI u we ask for the English label of u using a SPARQL query.[5] If such a label does not exist, we use either the fragment of u (the string after `#`) if it exists, else the string after the last occurrence of `/`. Finally this NL representation is realized as a noun phrase, and in the case of classes is also pluralized. As an example, `:Person` is realized as `people` (its label).

Properties The lexicalization of properties relies on the insight that most property labels are either nouns or verbs. While the mapping of a particular property `p` can be unambiguous, some property labels are not as easy to categorize. For examples, the label `crosses` can either be the plural form of the noun `cross` or the third person singular present form of the verb `to cross`. To automatically determine which realization to use, we relied on the insight that the first and last word of a property label are often the key to determining the type of the property: properties whose label begins with a verb (resp. noun or gerund) are most to be realized as verbs (resp. nouns). We devised a set of rules to capture this behavior, which we omit due to space restrictions. In some cases (such as `crosses`) none of the rules applied. In these cases, we compare the probability of $P(p|\texttt{noun})$ and $P(p|\texttt{verb})$ by measuring

$$P(p|X) = \frac{\sum\limits_{t \in synset(p|X)} \log_2(f(t))}{\sum\limits_{t' \in synset(p)} \log_2(f(t'))}, \qquad (1)$$

where $synset(p)$ is the set of all synsets of p, $synset(p|X)$ is the set of all synsets of p that are of

[3] For simplicity, we omit RDF blank nodes in subject or object position.

[4] http://www.w3.org/TR/sparql11-query

[5] Note that it could be any property which returns a NL representation of the given URI, see (Ell et al., 2011).

the syntactic class $X \in \{\texttt{noun}, \texttt{verb}\}$ and $f(t)$ is the frequency of use of p in the sense of the synset t according to WordNet. For

$$\frac{P(p|\texttt{verb})}{P(p|\texttt{noun})} \geq \theta, \tag{2}$$

we choose to realize `p` as a noun; else we realized it as a verb. For $\theta = 1$, for example, `dbo:crosses` is realized as a verb.

Literals Literals in an RDF graph usually consist of a *lexical form* `LF` and a *datatype IRI* `DT`, represented as `"LF"^^<DT>`. Optionally, if the datatype is `rdf:langString`, a non-empty *language tag* is specified and the literal is denoted as *language-tagged string*[6]. The realization of language-tagged strings is done by using simply the lexical form, while omitting the language tag. For example, `"Albert Einstein"@en` is realized as `Albert Einstein`. For other types of literals, we further differentiate between built-in and user-defined datatypes. For the former, we also use the lexical form, e.g. `"123"^^xsd:int` ⇒ `123`, while the latter are processed by using the literal value with its representation of the datatype IRI, e.g., `"123"^^dt:squareKilometre` as `123 square kilometres`.

4.1.2 Realizing single triples

The realization ρ of a triple `(s p o)` depends mostly on the verbalization of the predicate `p`. If `p` can be realized as a noun phrase, then a possessive clause can be used to express the semantics of `(s p o)`, more formally

1. $\rho(\texttt{s p o}) \Rightarrow$
   $\texttt{poss}(\rho(\texttt{p}),\rho(\texttt{s})) \wedge \texttt{subj}(\texttt{BE},\rho(\texttt{p}))$
   $\wedge\, \texttt{dobj}(\texttt{BE},\rho(\texttt{o}))$

For example, if $\rho(\texttt{p})$ is a relational noun like `birth place` e.g. in the triple `(:Albert_Einstein :birthPlace :Ulm)`, then the verbalization is `Albert Einstein's birth place is Ulm`. Note that `BE` stands for the verb “to be”. In case `p`'s realization is a verb, then the triple can be verbalized as follows:

2. $\rho(\texttt{s p o}) \Rightarrow$
   $\texttt{subj}(\rho(\texttt{p}),\rho(\texttt{s})) \wedge \texttt{dobj}(\rho(\texttt{p}),\rho(\texttt{o}))$

For example, in `(:Albert_Einstein :influenced :Nathan_Rosen)` $\rho(\texttt{p})$ is the verb `influenced`, thus, the verbalization is Albert Einstein influenced Nathan Rosen.

4.2 Realization - RDF Triples to NL

The same procedure of generating a single triple can be applied for the generation of each triple in a set of triples. However, the NL output would contain redundant information and consequently sound very artificial. Thus, the goal is to transform the generated description to sound more natural. To this end, we focus on two types of transformation rules (cf. (Dalianis and Hovy, 1996)): *ordering and clustering* and *grouping*. In the following, we describe the transformation rules we employ in more detail. Note that clustering and ordering (4.2.1) is applied before grouping (4.2.2).

4.2.1 Clustering and ordering rules

We process the input trees in descending order with respect to the frequency of the variables they contain, starting with the projection variables and only after that turning to other variables. As an example, consider the following triples about two of the most known people in the world:

```
:William_Shakespeare rdf:type :Writer .
:Albert_Einstein :birthPlace :Ulm .
:Albert_Einstein :deathPlace :Princeton
:Albert_Einstein rdf:type :Scientist .
:William_Shakespeare :deathDate
"1616-04-23"^^xsd:date .
```

The five triples are verbalized as given in 3a–3e. Clustering and ordering first take all sentences containing the subject `:Albert_Einstein`, i.e. 3b –3d, which are ordered such that copulative sentences (such as Albert Einstein is a scientist) come before other sentences, and then takes all sentences containing the remaining subject `:William_Shakespeare` in 3a and 3e resulting in a sequence of sentences as in 4.

3. (a) William Shakespeare is a writer.
   (b) Albert Einstein's birth place is Ulm.
   (c) Albert Einstein's death place is Princeton.
   (d) Albert Einstein is a scientist.
   (e) William Shakespeare's death date is 23 April 1616.

4. Albert Einstein is a scientist. Albert Einstein's birth place is Ulm. Albert Einstein's death place is Princeton. William Shakespeare's is a writer. William Shakespeare's death date is 23 April 1616.

4.2.2 Grouping

Dalianis and Hovy (1996) describe grouping as a process “collecting clauses with common elements and then collapsing the common elements”. The common elements are usually subject noun

[6] In RDF 1.0 literals have been divided into 'plain' literals with no type and optional language tags, and typed literals.

phrases and verb phrases (verbs together with object noun phrases), leading to *subject grouping* and *object grouping*. To maximize the grouping effects, we collapse common prefixes and suffixes of sentences, irrespective of whether they are full subject noun phrases or complete verb phrases. In the following we use $X_1, X_2, \ldots X_N$ as variables for the root nodes of the input sentences and Y as variable for the root node of the output sentence. Furthermore, we abbreviate a subject $\mathtt{subj}(X_i, s_i)$ as $\mathtt{s}_i$, an object $\mathtt{dobj}(X_i, o_i)$ as $\mathtt{o}_i$, and a verb $\mathtt{root}(\mathit{ROOT}_i, v_i)$ as $\mathtt{v}_i$.

Subject grouping collapses the predicates (i.e. verb and object) of two sentences if their subjects are the same, as specified in 5 (abbreviations as above).

5. $\rho(\mathtt{s}_1) = \rho(\mathtt{s}_2) \wedge \mathtt{cc}(\mathtt{v}_1, \mathit{coord})$ $\Rightarrow$ $\mathtt{root}(Y, \mathtt{coord}(\mathtt{v}_1, \mathtt{v}_2)) \wedge \mathtt{subj}(\mathtt{v}_1, \mathtt{s}_1) \wedge \mathtt{dobj}(\mathtt{v}_1, \mathtt{o}_1) \wedge \mathtt{subj}(\mathtt{v}_2, \mathtt{s}_1) \wedge \mathtt{dobj}(\mathtt{v}_1, \mathtt{o}_2)$

An example are the sentences given in 6, which share the subject Albert Einstein and thus can be collaped into a single sentence.

6. Albert Einstein is a scientist and Albert Einstein is known for general relativity.
⇒ Albert Einstein is a scientist and known for general relativity.

Object grouping collapses the subjects of two sentences if the realizations of the verbs and objects of the sentences are the same, where the *coord* $\in$ {and, or} is the coordination combining the input sentences X_1 and X_2, and $\mathtt{coord} \in \{\mathtt{conj}, \mathtt{disj}\}$ is the corresponding coordination combining the subjects.

7. $\rho(\mathtt{o}_1) = \rho(\mathtt{o}_2) \wedge \rho(\mathtt{v}_1) = \rho(\mathtt{v}_2) \wedge \mathtt{cc}(\mathtt{v}_1, \mathit{coord})$ $\Rightarrow$ $\mathtt{root}(Y, \mathrm{PLURAL}(\mathtt{v}_1)) \wedge \mathtt{subj}(\mathtt{v}_1, \mathtt{coord}(\mathtt{s}_1, \mathtt{s}_2)) \wedge \mathtt{dobj}(\mathtt{v}_1, \mathtt{o}_1)$

For example, the sentences in 8 share their verb and object, thus they can be collapsed into a single sentence. Note that to this end the singular auxiliary was needs to be transformed into its plural form were.

8. Benjamin Franklin was born in Boston. Leonard Nimoy was born in Boston. ⇒ Benjamin Franklin and Leonard Nimoy were born in Boston.

4.3 From OWL to NL

OWL 2 ontologies consist of Entities, Expressions and Axioms as introduced in subsection 3.1. While both expressions and axioms can be mapped to RDF[7], i.e. into a set of RDF triples, using this mapping and applying the triple-based verbalization on it would lead to a non-human understandable text in many cases. For example, the intersection of two classes `:A` and `:B` can be represented in RDF by the six triples

```
_:x rdf:type owl:Class .
_:x owl:intersectionOf _:y1 .
_:y1 rdf:first :A .
_:y1 rdf:rest _:y2 .
_:y2 rdf:first :B .
_:y2 rdf:rest rdf:nil .
```

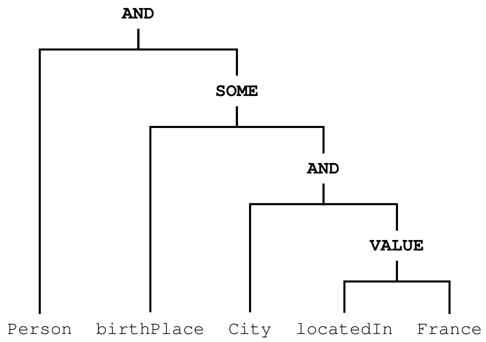
The verbalization of these triples would result in `Something that is a class and the intersection of something whose first is A and whose rest is something whose first is B and whose rest ist nil.`, which is obviously far away from how a human would express it in NL. Therefore, generating NL from OWL requires a different procedure based on its syntactic categories, OWL expressions and OWL axioms. We show the general rules for each of them in the following.

4.3.1 OWL Class Expressions

In theory, class expressions can be arbitrarily complex, but as it turned out in some previous analysis (Power and Third, 2010), in practice they seldom arise and can be seen as some corner cases. For example, an ontology could contain the following class expression about people and their birth place:

```
Person AND birthPlace SOME (City AND
    locatedIn VALUE France)
```

Class expressions do have a tree-like structure and can simply be parsed into a tree by means of the binary OWL class expressions constructors contained in it. For our example, this would result in the following tree:



Such a tree can be traversed in post-order, i.e. sub-trees are processed before their parent nodes recursively. For the sake of simplicity, we only process sub-trees that represent proper

[7] http://bit.ly/2Mc0vIw

class expression in our example, i.e. we omit `birthPlace`, `locatedIn`, and `France`. Moreover and again for simplicity, we'll explain the transformation process by starting from the right-hand side of the tree. Thus, in our example we begin with the class expression `City` which is transformed to `everything that is a city` and `locatedIn VALUE France` resulting in `everything that is located in France` by application of a rule. Both class expressions are used in the conjunction `City AND locatedIn VALUE France`. Thus, the next step would be to merge both phrases. An easy way is to use the coordinating conjunction `and`, i.e. `everything that is a city and everything that is located in France`. Although the output of this transformation is correct, it still contains unnecessarily redundant information. Therefore, we apply the aggregation procedure described in Section 4.2.2, i.e. we get `everything that is a city and located in France`. Yet, the aggregation can still be improved: if there is any atomic class in the conjunction, we know that this is more specific than the placeholder `everything`. Thus, we can replace it by the plural form of the class, finally resulting in `cities that are located in France`. The same procedure is applied for its parent class expression being the existential restriction

```
birthPlace SOME (City AND locatedIn
    VALUE France)
```

This will be transformed to `everything whose birth place is a city that is located in France`. Note, that we used the singular form here, assuming that the property `birthPlace` is supposed to be functional in the ontology. In the last step, we process the class expression `Person`, which gives us `everything that is a person`. Again, due to the conjunction we merge this result with with the previous one, such that in the end we get `people whose birth place is a city that is located in France`.

4.3.2 OWL Axioms

As we described in Section 4.3, OWL axioms can roughly be categorized into terminological and assertional axioms. Therefore, we have different procedures for processing each category:

Assertional Axioms (ABox Axioms) - Most assertional axioms assert individuals to atomic classes or relate individuals to another individual resp. literal value. For example, axioms about the type as well as birth place and birth date of Albert Einstein can be expressed by

```
Individual: Albert_Einstein
  Types: Person
  Facts: birthPlace Ulm, birthDate "
      1879-03-14"^^xsd:date
```

Those axioms can simply be rewritten as triples, thus, we can use the same procedure as we do for triples (Section 4.1.2). Converting them into NL gives us `Albert Einstein is a person whose birth place is Ulm and whose birth date is 14 March 1879`. OWL also allows for assigning an individual to a complex class expression. In that case we'll use our conversion of OWL class expressions as described in Section 4.3.1.

Terminological Axioms (TBox Axioms) - According to Power and Third (2010), most of the terminological axioms used in ontologies are subclass axioms. By definition, subclass and superclass can be arbitrarily complex class expressions $\mathtt{CE}_1$ and $\mathtt{CE}_2$, i.e. $\mathtt{CE}_1$ **`SubClassOf`** $\mathtt{CE}_2$, but in praxis it is quite often only used with atomic classes as subclass or even more simple with the superclass also beeing an atomic class. Nevertheless, we support any kind of subclass axiom and all other logical OWL axioms in LD2NL. For simplicity, we outline here how we verbalize subclass axioms in LD2NL. The semantics of a subclass axiom denotes that every individual of the subclass also belongs to the superclass. Thus, the verbalization seems to be relatively straightforward, i.e. we verbalize both class expressions and follow the template : every $\rho(\mathtt{CE}_1)$ is a $\rho(\mathtt{CE}_2)$. Obviously, this works pretty well for subclass axioms with atomic classes only. For example, the axiom

```
Class: Scientist
  SubClassOf: Person
```

is verbalized as `every scientist is a person`.

4.4 From SPARQL to NL

A SPARQL `SELECT` query can be regarded as consisting of three parts: (1) a *body section* `B`, which describes all data that has to be retrieved, (2) an *optional section* `O`, which describes the data items that can be retrieved by the query if they exist, and (3) a *modifier section* `M`, which describes all solution sequences, modifiers and aggregates that are to be applied to the result of the previous two sections of the query. Let *Var* be the set of all variables that can be used in a SPARQL query. In addition, let R be the set of all resources, P the set of all properties and L the set of all literals contained in the target knowledge base of the SPARQL queries at hand. We call $x \in Var \cup R \cup P \cup L$ an *atom*. The basic components of the body of a SPARQL query are triple patterns $(\texttt{s}, \texttt{p}, \texttt{o}) \in (Var \cup R) \times (Var \cup P) \times (Var \cup R \cup L)$. Let W be the set of all words in the dictionary of our target language. We define the realization function $\rho : Var \cup R \cup P \cup L \rightarrow W^*$ as the function which maps each atom to a word or sequence of words from the dictionary. The extension of ρ to all SPARQL constructs maps all atoms x to their realization $\rho(x)$ and defines how these atomic realizations are to be combined. We denote the extension of ρ by the same label ρ for the sake of simplicity. We adopt a rule-based approach to achieve this goal, where the rules extending ρ to all valid SPARQL constructs are expressed in a conjunctive manner. This means that for premises $P_1, \ldots, P_n$ and consequences $K_1, \ldots, K_m$ we write $P_1 \wedge \ldots \wedge P_n \Rightarrow K_1 \wedge \ldots \wedge K_m$. The premises and consequences are explicated by using an extension of the Stanford dependencies[8].

For example, a possessive dependency between two phrase elements e_1 and e_2 is represented as $\texttt{poss}(e_1, e_2)$. For the sake of simplicity, we slightly deviate from the Stanford vocabulary by not treating the copula to be as an auxiliary, but denoting it as BE. Moreover, we extend the vocabulary by the constructs conj and disj which denote the conjunction resp. disjunction of two phrase elements. In addition, we sometimes reduce the construct $\texttt{subj(y,x)} \wedge \texttt{dobj(y,z)}$ to the triple $\texttt{(x,y,z)} \in W^3$.

5 Experiments

We evaluated our approach in three different experiments based on human ratings. We divided the volunteers into two groups—domain experts and non-experts. The group of domain experts comprised 66 persons while there were 20 non-experts forming the second group. In the first experiment, an OWL axiom and its verbalization were shown to the experts who were asked to rate the verbalization regarding the two following measures according to Gardent et al. (2017): (1) Adequacy: Does the text contain only and all the information from the data? (2) Fluency: Does the text sound fluent and natural?. For both measures the volunteers were asked to rate on a scale from 1 (Very Bad) to 5 (Very Good). The experiment was carried out using 41 axioms of the Koala ontology.[9] Because of the complexity of OWL axioms, only domain experts were asked to perform this experiment.

In the second experiment, a set of triples describing a single resource and their verbalization were shown to the volunteers. The experts were asked to rate the verbalization regarding adequacy, fluency and *completeness*, i.e., whether all triples have been covered. The non-experts were only asked to rate the fluency. The experiment was carried out using 6 DBpedia resources.In the third experiment, the verbalization of an OWL class and 5 resources were shown to the human raters. For non-experts, the resources have been verbalized as well, while for domain experts the resources were presented as triples. The task of the raters was to identify the resource that fits the class description and, thus, is an instance of the class. We used 4 different OWL axioms and measured the amount of correct identified class instances.

Results In our first series of experiments, the verbalization of OWL axioms, we achieved an average adequacy of 4.4 while the fluency reached 4.38. In addition, more than 77% of the verbalizations were assigned the maximal adequacy (i.e., were assigned a score of 5, see Fig. 1). The maximal score for fluency was achieved in more than 69% of the cases (see Fig. 1). This clearly indicates that the verbalization of axioms generated by LD2NL can be easily understood by domain experts and contains all the information necessary to access the input OWL class expression.

Experiments on the verbalization of summaries for RDF resources revealed that verbalizing resource summaries is a more difficult task. While the adequacy of the verbalization was assigned an average score of 3.92 by experts (see Fig. 2), the fluency was assigned a average score of 3.47 by experts and 3.0 by non-experts (see Fig. 2). What these results suggest is that (1) our framework generates sentences that are close to that which a domain expert would also generate (adequacy). However (2) while the sentence is grammatically

[8] For a complete description of the vocabulary, see https://stanford.io/2EzMjmo.

[9] https://bit.ly/2K8BWts

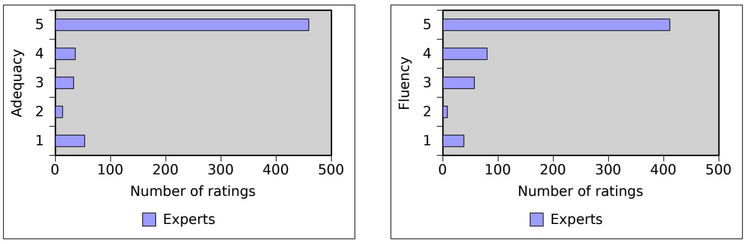


Figure 1: Experiment I: adequacy (left) and fluency (right) ratings

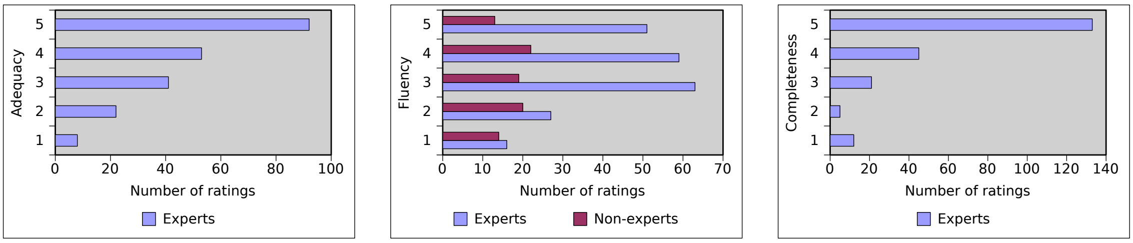


Figure 2: Experiment II: adequacy (left), fluency (middle) and completeness (left) results

sufficient for the experts, it is regarded by non-domain experts (which were mostly linguists, i.e., the worst-case scenario for such an evaluation) as being grammatically passably good but still worthy of improvement. The completeness rating achieves a score of 4.31 on average (see Fig. 2). This was to be expected as we introduced a rule to shorten the description of resources that contain more than 5 triples which share a common subject and predicate. Finally, we measured how well the users and experts were able to understand the meaning of the text generated by our approach. As expected, the domain experts outperform the non-expert users by being able to find the answers to 87.2% of the questions. The score achieved by non-domain experts, i.e., 80%, still suggest that our framework is able to bridge the gap pertaining to understand RDF and OWL for non-experts from 0% to 80%, which is more than 91.8% of the performance of experts.

Discussion Our evaluation results suggest that the verbalization of these languages is a non-trivial task that can be approached by using a bottom-up approach. As expected, the verbalization of short expressions leads to sentences which read as if they have been generated by a human. However, due to the complexity of the semantics that can be expressed by the languages at hand, long expressions can sound mildly artificial. Our results however also suggest that although the text generated can sound artificial, it is still clear enough to enable non-expert users to achieve results that are comparable to those achieved by experts. Hence, our first conclusion is that our framework clearly serves its purpose. Still, potential improvements can be derived from the results achieved during the experiments. In particular, we will consider the used of attention-based encoder-decoder networks to improve the fluency of complex sentences.

6 Conclusion and Future Work

In this paper, we presented LD2NL, a framework for verbalizing SW languages, especially on RDF and OWL while including the SPARQL verbalization provided by SPARQL2NL. Our evaluation with 86 persons revealed that our framework generates NL that can be understood by lay users. While the OWL verbalization was close to NL, the RDF was less natural but still sufficient to convey the meaning expressed by the corresponding set of triples. In future work, we aim to extend LD2NL to verbalize the languages SWRL (Horrocks et al., 2004) and SHACL (Knublauch and Kontokostas, 2017).

Acknowledgments

This work was supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) through the projects LIMBO (no. 19F2029I) and OPAL (no. 19F2028A). This work was supported by the German Federal Ministry of Economics and Technology (BMWI) in the projects RAKI (no. 01MD19012D) as well as by the BMBF project SOLIDE (no. 13N14456).

References

Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. 2013. Generating natural language descriptions from OWL ontologies: The natural owl system. *J. Artif. Int. Res.* 48(1):671–715.

Or Biran and Kathleen McKeown. 2015. Discourse planning with an n-gram model of relations. In *EMNLP*. pages 1973–1977.

Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. 2016. Dl-learnera framework for inductive learning on the semantic web. *Journal of Web Semantics* 39:15–24.

Philipp Cimiano, Janna Lüker, David Nagel, and Christina Unger. 2013. Exploiting ontology lexica for generating natural language texts from rdf data. In *Proceedings of the 14th European Workshop on Natural Language Generation*. ACL, Sofia, Bulgaria, pages 10–19.

Emilie Colin, Claire Gardent, Yassine Mrabet, Shashi Narayan, and Laura Perez-Beltrachini. 2016. The webnlg challenge: Generating text from dbpedia data. In *Proceedings of the 9th INLG conference*. pages 163–167.

H. Dalianis and E.H. Hovy. 1996. Aggregation in natural language generation. In G. Adorni and M. Zock, editors, *Trends in natural language generation: an artificial intelligence perspective*, Springer, volume 1036 of *Lecture Notes in Artificial Intelligence*, pages 88–105.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *IWCS*. pages 83–94.

Basil Ell and Andreas Harth. 2014. A language-independent method for the extraction of rdf verbalization templates. In *INLG*. pages 26–34.

Basil Ell, Denny Vr, and Elena Simperl. 2012. SPARTIQULATION Verbalizing SPARQL queries. In *In Proceedings of ILD Workshop, ESWC*.

Basil Ell, Denny Vrandecic, and Elena Paslaru Bontas Simperl. 2011. Labels in the web of data. In *Proceedings of ISWC*. Springer, volume 7031, pages 162–176.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *Proceedings of ACL*.

Albert Gatt and Emiel Krahmer. 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *arXiv preprint arXiv:1703.09902* .

Matthew Horridge, Nick Drummond, John Goodwin, Alan L Rector, Robert Stevens, and Hai Wang. 2006. The Manchester OWL syntax. In *OWLed*. volume 216.

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosofand, and Mike Dean. 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission.

Holger Knublauch and Dimitris Kontokostas. 2017. Shapes constraint language (shacl). *W3C Candidate Recommendation* 11(8).

Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Jon Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, David Liu, and Sören Auer. 2012. DEQA: Deep Web Extraction for Question Answering. In *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II*. pages 131–147.

Yassine Mrabet, Pavlos Vougiouklis, Halil Kilicoglu, Claire Gardent, Dina Demner-Fushman, Jonathon Hare, and Elena Simperl. 2016. Aligning texts and knowledge bases with semantic sentence simplification. *WebNLG 2016* .

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, I Don't Speak SPARQL: Translating SPARQL Queries into Natural Language. In *Proceedings of the 22Nd International Conference on World Wide Web*. ACM, New York, NY, USA, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: translating sparql queries into natural language. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, pages 977–988.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. SPARQL2NL: Verbalizing SPARQL queries. In *22nd International World Wide Web Conference, Rio de Janeiro, Brazil, May 13-17*. pages 329–332.

W3C OWL Working Group. 2009. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation.

Laura Perez-Beltrachini, Rania Sayed, and Claire Gardent. 2016. Building rdf content for data-to-text generation. In *COLING*. pages 1493–1502.

Richard Power and Allan Third. 2010. Expressing OWL Axioms by English Sentences: Dubious in Theory, Feasible in Practice. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. ACL, Stroudsburg, PA, USA, pages 1006–1013.

W3C RDF Working Group. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation.

Ehud Reiter and Robert Dale. 2000a. *Building natural language generation systems*. Cambridge university press.

Ehud Reiter and Robert Dale. 2000b. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Amin Sleimi and Claire Gardent. 2016. Generating paraphrases from dbpedia using deep learning. *WebNLG 2016* page 54.

Building a Comprehensive Romanian Knowledge Base for Drug Administration

Bogdan Nicula[1], Mihai Dascalu[1], Maria-Dorinela Sirbu[1], Stefan Trăușan-Matu[1] and Alexandru Nuta[2]

[1]University Politehnica of Bucharest, 313 Splaiul Independentei, 060042, Bucharest, Romania
[2]All Business Management, Str. Vasile Conta 19, sc. A, et. 1, ap. 8, Bucharest, Romania

Abstract

Information on drug administration is obtained traditionally from doctors and pharmacists, as well as leaflets which provide in most cases cumbersome and hard-to-follow details. Thus, the need for medical knowledge bases emerges to provide access to concrete and well-structured information which can play an important role in informing patients. This paper introduces a Romanian medical knowledge base focused on drug-drug interactions, on representing relevant drug information, and on symptom-disease relations. The knowledge base was created by extracting and transforming information using Natural Language Processing techniques from both structured and unstructured sources, together with manual annotations. The resulting Romanian ontologies are aligned with larger, well-established, English medical ontologies. Our knowledge base supports queries regarding drugs (e.g., active ingredients, concentration, expiration date), drug-drug interaction, symptom-disease relations, as well as drug-symptom relations (e.g., searching for the drug that might be most useful for treating a given set of symptoms).

1 Introduction

The conventional way of accessing information regarding drug administration and storage strategies, recommendations and precautions, effects and side-effects is via medical leaflets. However, most of the times the text is too complex, too cluttered with information, that the leaflet ends up being ignored by the consumer. At the same time, when consumers must take multiple drugs as result of overlapping treatment schemes, it becomes increasingly difficult for them to keep in mind all mentioned precautions and counter-indications. This paper presents a knowledge base built upon medical leaflets and existing medical ontologies aimed at aiding Romanian consumers when adding new drugs to their treatment schemes.

The knowledge base consists of a set of two ontologies built from information extracted from websites of pharmaceutical producers and national agencies, which were combined with English medical ontologies. The first ontology is used to better structure the leaflet content and provide easy access to information concerning administration and storage strategies, together with possible side-effects. This ontology is aligned with a larger English-based ontology - DINTO (Herrero-Zazo et al., 2015) - in order to discover incompatibilities between drugs and to warn the user whether two administered drugs might interact one with another. The second ontology is focused on diseases. Translations for both diseases and symptoms were added for Romanian language, allowing customers to lookup possible explanations for their symptoms. Moreover, the gap between the two ontologies is filled in by indexing the description and recommendation texts for the drug leaflets, therefore enabling users to directly search what drugs might help them deal with certain symptoms.

These tools are not supposed to replace in any manner actual pharmacists, and an extra opinion from a pharmacist or a physician is always recommended when asking for a drug, given a set of symptoms. The aim of our system is to provide support and easy access to essential information, when no similar solutions are available for Romanian language.

Proceedings of Recent Advances in Natural Language Processing, pages 829–836,
Varna, Bulgaria, Sep 2–4 2019.

The second section of this paper covers similar knowledge bases, as well as systems providing similar information for English language. The third section describes the data extraction and the architecture of our knowledge base. The last two sections focus on the current results, shortcomings and ways of further improving our knowledge base.

2 Related Work

2.1 Medical Ontologies

Multiple knowledge bases for English language exist, covering different medical areas of interest. Part of them were developed by authors of the Open Biological and Biomedical Ontology (OBO) Foundry (Smith et al., 2007) which focuses on collaborations and on defining a common set of principles for developing medical ontologies. The foundry's mission is to develop a set of interoperable ontologies that are well formed and scientifically accurate. These ontologies are built using semantic web technologies (Berners-Lee et al., 2001) and they are usually made available in OWL (McGuinness et al., 2004) format. Over 150 ontologies are currently listed on the OBO webpage (http://www.obofoundry.org/).

Out of all the ontologies developed by OBO members, the following knowledge bases are relevant for the functionalities presented in this paper:

- CHEBI (Chemical Entities of Biological Interest), containing information regarding a diverse set of chemical compounds relevant for biological interests (Hastings et al., 2015).
- DINTO (The Drug-Drug Interactions Ontology), covering information on how 2 active ingredients interact one with another; DINTO is integrated with CHEBI.
- DOID (Human Disease Ontology), a taxonomy of human diseases (Kibbe et al., 2014).
- SYMP (Symptom Ontology), including symptoms which may be indicative of a disease. SYMP was developed as part of the Gemina project (Schriml et al., 2009) and is integrated with DOID.

Besides OBO, other detailed medical ontologies have been created, such as FMA (The Foundational Model of Anatomy Ontology) (Rosse and Mejino Jr, 2003) developed by the University of Washington, which focuses on the structure of the human body. Generic ontologies are also available, such as DBpedia (Bizer et al., 2009) built by using data from Wikipedia, but they do not offer information relevant for the task at hand.

2.2 Medical Applications

Several applications offering drug-related information exist for English Language. They allow users to search a drug by name, illness or medical procedure, and offer the possibility of testing whether two drugs are incompatible due to harmful interactions between their active substances.

One of the most known portals making medical information available and easy to interpret for non-professional users is WebMD. WebMD offers two applications, Medscape and the WebMD app. Medscape is focused on the more practical aspects of healthcare, offering features such as identifying pills based on a set of physical features, computing the body mass index (BMI) or other relevant metrics based on user's input, and searching for nearby medical professionals and hospitals. The WebMD app is focused more on offering theoretical insights regarding drugs and diseases. It offers the possibility of searching a disease based on a list of symptoms, searching for remedies based on age, gender and severity of the symptoms, and it can notify the user when the US Food and Drug Administration (FDA) has published new information regarding a drug from the user's treatment profile.

Other applications, such as Drugs.com, offer similar features, but also take into account community feedback as a mechanism for informing users. The application facilitates communication within its user-base, allowing customers to find relevant and useful information from peers who underwent similar experiences. This application also differentiates itself from the rest by offering two different views based on the user's medical proficiency. Users with little medical knowledge are directed towards pages with simple and easy-to-grasp information, while experts are provided access to more complex content, which includes more scientific terms.

A specific sub-category of applications is focused on providing drug-administration assistants. One such example is CareZone. These systems allow users to register all drugs on their current treatment scheme, and offer the possibility of entering and keeping track of different medical parameters, such as blood sugar levels. Apart from that, the application can also be used to set up reminders for administering drugs.

3 Method

3.1 Corpus

There is no established medical or biological ontology for Romanian language. However, public information is readily available in both structured and unstructured format. Medical leaflets are available either as .pdf files or integrated in web pages made available by both private drug producers (e.g., Biofarm) and by state authorities (e.g., The National Agency for Drugs and Medical Devices - ANMDM). The web page of ANMDM also contains structured information (e.g., active substances, concentrations, therapeutical role) for all approved drugs. Figure 1 contains an overview of extracted information from the considered data sources; specific details are provided in the follow-up subsections.

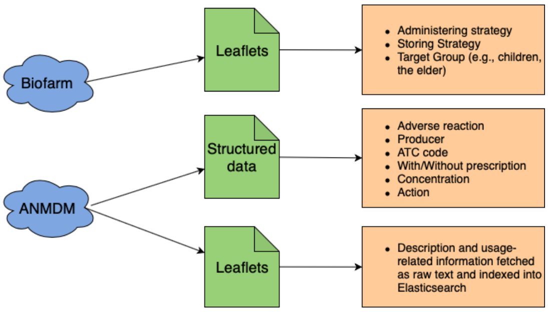

Biofarm
ANMDM
Leaflets
Structured data
Leaflets
• Administering strategy
• Storing Strategy
• Target Group (e.g., children, the elder)
• Adverse reaction
• Producer
• ATC code
• With/Without prescription
• Concentration
• Action
• Description and usage-related information fetched as raw text and indexed into Elasticsearch

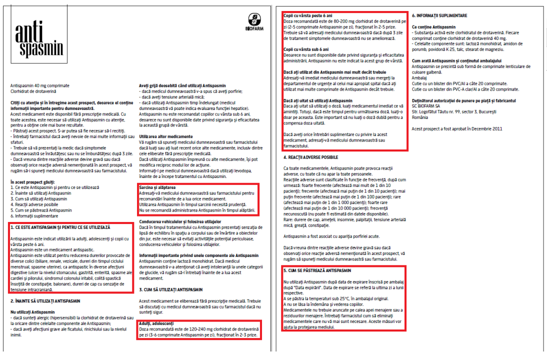
Figure 1: Considered data sources.

The functionalities supported by our knowledge base can be split into three different categories:

- Accessing drug-related information, including inferences of drug-drug interactions, given two or more drugs.
- Finding the most probable disease given a set of symptoms, or listing the set of symptoms that correspond to a certain disease.
- Finding the drug that is most likely to address a given symptom (e.g., flu medication in case of fever and coughing).

3.1.1 Information Regarding Drugs

Both ANMDM structured web information regarding drugs, as well as medical leaflets obtained from ANMDM and private drug producers were used to create a drugs ontology. A total of 220 leaflets from Biofarm (Figure 2) and 1138 leaflets from ANMDM were parsed (Figures 3-4), containing information on 15,093 drugs having 1330 different active substances.


anti spasmin
BIOFARM

Antispasmin 40 mg comprimate
Clorhidrat de drotaverină

Citiţi cu atenţie şi în întregime acest prospect, deoarece el conţine informaţii importante pentru dumneavoastră.
Acest medicament este disponibil fără prescripţie medicală. Cu toate acestea, este necesar să utilizaţi Antispasmin cu atenţie, pentru a obţine cele mai bune rezultate.
- Păstraţi acest prospect. S-ar putea să fie necesar să-l recitiţi.
- Întrebaţi farmacistul dacă aveţi nevoie de mai multe informaţii sau sfaturi.
- Trebuie să vă prezentaţi la medic dacă simptomele dumneavoastră se înrăutăţesc sau nu se îmbunătăţesc după 3 zile.
- Dacă vreuna dintre reacţiile adverse devine gravă sau dacă observaţi orice reacţie adversă nemenţionată în acest prospect, vă rugăm să-i spuneţi medicului dumneavoastră sau farmacistului.

În acest prospect găsiţi:
1. Ce este Antispasmin şi pentru ce se utilizează
2. Înainte să utilizaţi Antispasmin
3. Cum să utilizaţi Antispasmin
4. Reacţii adverse posibile
5. Cum se păstrează Antispasmin
6. Informaţii suplimentare

1. CE ESTE ANTISPASMIN ŞI PENTRU CE SE UTILIZEAZĂ

Antispasmin este indicat utilizării la adulţi, adolescenţi şi copii cu vârsta peste 6 ani.
Antispasmin este un medicament antispastic.
Antispasmin este utilizat pentru reducerea durerilor provocate de diverse colici (biliare, renale, vezicale, dureri din timpul ciclului menstrual, spasme uterine), ca antispastic în diverse afecţiuni digestive (ulcer la nivelul stomacului, gastrită, enterită, spasme ale cardiei şi pilorului, sindromul colonului iritabil, colită spastică însoţită de constipaţie, balonare), dureri de cap cu senzaţie de tensiune intracraniană.

2. ÎNAINTE SĂ UTILIZAŢI ANTISPASMIN

Nu utilizaţi Antispasmin
- dacă sunteţi alergic (hipersensibil) la clorhidrat de drotaverină sau la oricare dintre celelalte componente ale Antispasmin;
- dacă aveţi afecţiuni grave ale ficatului, rinichiului sau la nivelul inimii.

Aveţi grijă deosebită când utilizaţi Antispasmin
- dacă medicul dumneavoastră v-a spus că aveţi porfirie;
- dacă aveţi tensiune arterială mică;
- dacă utilizaţi Antispasmin timp îndelungat (medicul dumneavoastră vă poate indica evaluarea funcţiei hepatice).
Antispasmin nu este recomandat copiilor cu vârsta sub 6 ani, deoarece nu sunt disponibile date privind siguranţa şi eficacitatea la această grupă de vârstă.

Utilizarea altor medicamente
Vă rugăm să spuneţi medicului dumneavoastră sau farmacistului dacă luaţi sau aţi luat recent orice alte medicamente, inclusiv dintre cele eliberate fără prescripţie medicală.
Dacă utilizaţi Antispasmin împreună cu alte medicamente, îşi pot modifica reciproc modul lor de acţiune.
Informaţi-l pe medicul dumneavoastră dacă utilizaţi levodopa, înainte de a începe tratamentul cu Antispasmin.

Sarcina şi alăptarea
Adresaţi-vă medicului dumneavoastră sau farmacistului pentru recomandări înainte de a lua orice medicament.
Utilizarea Antispasmin în timpul sarcinii necesită prudenţă.
Nu se recomandă administrarea Antispasmin în timpul alăptării.

Conducerea vehiculelor şi folosirea utilajelor
Dacă în timpul tratamentului cu Antispasmin prezentaţi senzaţia de lipsă de echilibru în spaţiu a corpului sau de învârtire a obiectelor din jur, este necesar să evitaţi activităţile potenţial periculoase, conducerea vehiculelor şi folosirea utilajelor.

Informaţii importante privind unele componente ale Antispasmin
Antispasmin conţine lactoză monohidrat. Dacă medicul dumneavoastră v-a atenţionat că aveţi intoleranţă la unele categorii de glucide, vă rugăm să-l întrebaţi înainte de a lua acest medicament.

3. CUM SĂ UTILIZAŢI ANTISPASMIN

Acest medicament se eliberează fără prescripţie medicală. Trebuie să discutaţi cu medicul dumneavoastră sau cu farmacistul dacă nu sunteţi sigur.

Adulţi, adolescenţi
Doza recomandată este de 120-240 mg clorhidrat de drotaverină pe zi (3-6 comprimate Antispasmin pe zi), fracţionat în 2-3 prize.

Copii cu vârsta peste 6 ani
Doza recomandată este de 80-200 mg clorhidrat de drotaverină pe zi (2-5 comprimate Antispasmin pe zi), fracţionat în 2-5 prize.
Trebuie să vă adresaţi medicului dumneavoastră dacă după 3 zile de tratament simptomele dumneavoastră nu se ameliorează.

Copii cu vârsta sub 6 ani
Deoarece nu sunt disponibile date privind siguranţa şi eficacitatea administrării, Antispasmin nu este indicat la acest grup de vârstă.

Dacă aţi utilizat din Antispasmin mai mult decât trebuie
Adresaţi-vă imediat medicului dumneavoastră sau mergeţi la departamentul de urgenţe al celui mai apropiat spital dacă aţi utilizat mai multe comprimate de Antispasmin decât trebuie.

Dacă aţi uitat să utilizaţi Antispasmin
Daca aţi uitat să utilizaţi o doză, luaţi medicamentul imediat ce vă amintiţi. Totuşi, dacă este timpul pentru următoarea doză, luaţi-o doar pe aceasta. Este important să nu luaţi o doză dublă pentru a compensa doza uitată.

Dacă aveţi orice întrebări suplimentare cu privire la acest medicament, adresaţi-vă medicului dumneavoastră sau farmacistului.

4. REACŢII ADVERSE POSIBILE

Ca toate medicamentele, Antispasmin poate provoca reacţii adverse, cu toate că nu apar la toate persoanele.
Reacţiile adverse sunt clasificate în funcţie de frecvenţă, după cum urmează: foarte frecvente (afectează mai mult de 1 din 10 pacienţi); frecvente (afectează mai puţin de 1 din 10 pacienţi); mai puţin frecvente (afectează mai puţin de 1 din 100 pacienţi); rare (afectează mai puţin de 1 din 1 000 pacienţi); foarte rare (afectează mai puţin de 1 din 10 000 pacienţi); frecvenţă necunoscută (nu poate fi estimată din datele disponibile).
Rare: durere de cap, ameţeli, insomnie, palpitaţii, tensiune arterială mică, greaţă, constipaţie.

Antispasmin a fost asociat cu apariţia porfiriei acute.

Dacă vreuna dintre reacţiile adverse devine gravă sau dacă observaţi orice reacţie adversă nemenţionată în acest prospect, vă rugăm să spuneţi medicului dumneavoastră sau farmacistului.

5. CUM SE PĂSTREAZĂ ANTISPASMIN

Nu utilizaţi Antispasmin după data de expirare înscrisă pe ambalaj după "Data expirării". Data de expirare se referă la ultima zi a lunii respective.
A se păstra la temperaturi sub 25°C, în ambalajul original.
A nu se lăsa la îndemâna şi vederea copiilor.
Medicamentele nu trebuie aruncate pe calea apei menajere sau a reziduurilor menajere. Întrebaţi farmacistul cum să eliminaţi medicamentele care nu vă mai sunt necesare. Aceste măsuri vor ajuta la protejarea mediului.

6. INFORMAŢII SUPLIMENTARE

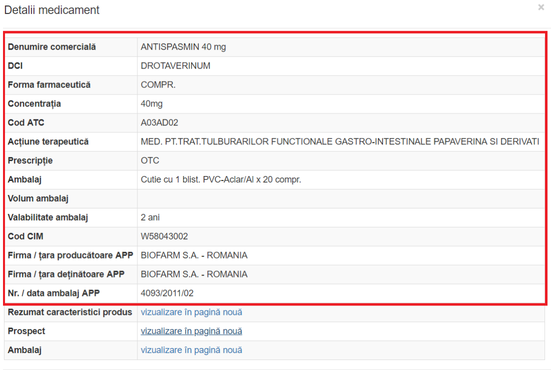
Ce conţine Antispasmin
- Substanţa activă este clorhidratul de drotaverină. Fiecare comprimat conţine clorhidrat de drotaverină 40 mg.
- Celelalte componente sunt: lactoză monohidrat, amidon de porumb, povidonă K 25, talc, stearat de magneziu.

Cum arată Antispasmin şi conţinutul ambalajului
Antispasmin se prezintă sub formă de comprimate lenticulare de culoare galbenă.
Ambalaj
Cutie cu un blister din PVC/Al a câte 20 comprimate.
Cutie cu un blister din PVC-A clar/Al a câte 20 comprimate.

Deţinătorul autorizaţiei de punere pe piaţă şi fabricantul
SC BIOFARM SA
Str. Logofătul Tăutu nr. 99, sector 3, Bucureşti
România

Acest prospect a fost aprobat în Decembrie 2011

Figure 2: Content extracted from Biofarm leaflets.



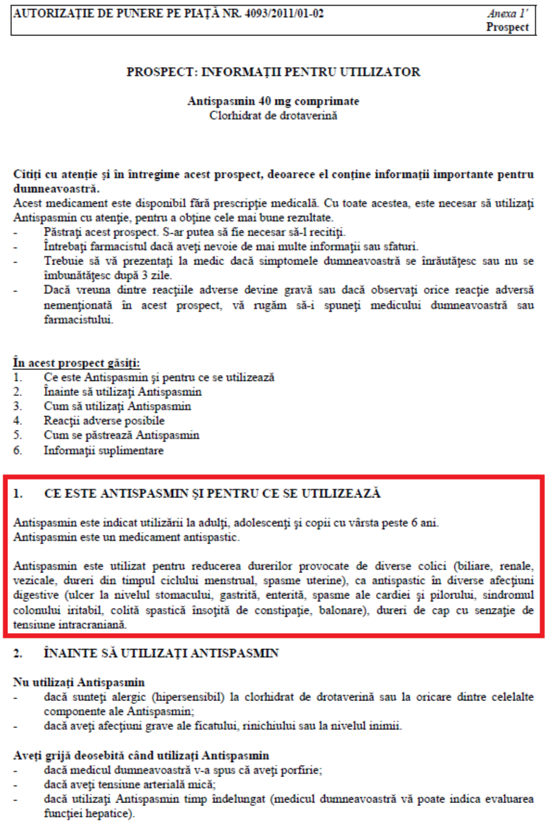
Detalii medicament ×

Denumire comercială	ANTISPASMIN 40 mg
DCI	DROTAVERINUM
Forma farmaceutică	COMPR.
Concentrația	40mg
Cod ATC	A03AD02
Acțiune terapeutică	MED. PT.TRAT.TULBURARILOR FUNCTIONALE GASTRO-INTESTINALE PAPAVERINA SI DERIVATI
Prescripție	OTC
Ambalaj	Cutie cu 1 blist. PVC-Aclar/Al x 20 compr.
Volum ambalaj	
Valabilitate ambalaj	2 ani
Cod CIM	W58043002
Firma / țara producătoare APP	BIOFARM S.A. - ROMANIA
Firma / țara deținătoare APP	BIOFARM S.A. - ROMANIA
Nr. / data ambalaj APP	4093/2011/02
Rezumat caracteristici produs	vizualizare în pagină nouă
Prospect	vizualizare în pagină nouă
Ambalaj	vizualizare în pagină nouă

Figure 3: Structured information scrapped from ANMDM.

The ontology was built from scratch and the attributes for each drug instance were either copied from the structured ANMDM web entry, or were extracted from the corresponding medical leaflets (such as the administering strategy). The extraction was done partially via a rule-based approach, but manual extraction was necessary for some of the more complex entries, when no reliable pattern could be identified.

As it can be seen in Figure 5, the ontology is centered on the Drug class, which represents a certain type of drug, but not an actual product that can be bought in stores. All instances of the Drug



AUTORIZAȚIE DE PUNERE PE PIAȚĂ NR. 4093/2011/01-02 *Anexa 1'* **Prospect**

PROSPECT: INFORMAŢII PENTRU UTILIZATOR

Antispasmin 40 mg comprimate
Clorhidrat de drotaverină

Citiţi cu atenţie şi în întregime acest prospect, deoarece el conţine informaţii importante pentru dumneavoastră.
Acest medicament este disponibil fără prescripţie medicală. Cu toate acestea, este necesar să utilizaţi Antispasmin cu atenţie, pentru a obţine cele mai bune rezultate.

- Păstraţi acest prospect. S-ar putea să fie necesar să-l recitiţi.
- Întrebaţi farmacistul dacă aveţi nevoie de mai multe informaţii sau sfaturi.
- Trebuie să vă prezentaţi la medic dacă simptomele dumneavoastră se înrăutăţesc sau nu se îmbunătăţesc după 3 zile.
- Dacă vreuna dintre reacţiile adverse devine gravă sau dacă observaţi orice reacţie adversă nemenţionată în acest prospect, vă rugăm să-i spuneţi medicului dumneavoastră sau farmacistului.

<u>În acest prospect găsiţi:</u>
1. Ce este Antispasmin şi pentru ce se utilizează
2. Înainte să utilizaţi Antispasmin
3. Cum să utilizaţi Antispasmin
4. Reacţii adverse posibile
5. Cum se păstrează Antispasmin
6. Informaţii suplimentare

1. CE ESTE ANTISPASMIN ȘI PENTRU CE SE UTILIZEAZĂ

Antispasmin este indicat utilizării la adulţi, adolescenţi şi copii cu vârsta peste 6 ani.
Antispasmin este un medicament antispastic.

Antispasmin este utilizat pentru reducerea durerilor provocate de diverse colici (biliare, renale, vezicale, dureri din timpul ciclului menstrual, spasme uterine), ca antispastic în diverse afecţiuni digestive (ulcer la nivelul stomacului, gastrită, enterită, spasme ale cardiei şi pilorului, sindromul colonului iritabil, colită spastică însoţită de constipaţie, balonare), dureri de cap cu senzaţie de tensiune intracraniană.

2. ÎNAINTE SĂ UTILIZAŢI ANTISPASMIN

Nu utilizaţi Antispasmin
- dacă sunteţi alergic (hipersensibil) la clorhidrat de drotaverină sau la oricare dintre celelalte componente ale Antispasmin;
- dacă aveţi afecţiuni grave ale ficatului, rinichiului sau la nivelul inimii.

Aveţi grijă deosebită când utilizaţi Antispasmin
- dacă medicul dumneavoastră v-a spus că aveţi porfirie;
- dacă aveţi tensiune arterială mică;
- dacă utilizaţi Antispasmin timp îndelungat (medicul dumneavoastră vă poate indica evaluarea funcţiei hepatice).

Figure 4: Content extracted from ANMDM leaflets.

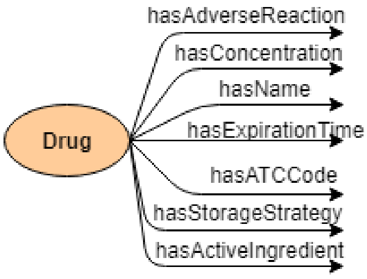


Figure 5: Romanian drug ontology - drug class simplified view.

class contain the same combination of active ingredients, but they may have different concentrations, different names, different expiration times, etc.

This ontology was aligned to its larger English counterpart, DINTO. The data from the two ontologies was merged at active substance level. Difficulties were encountered as, even though most active substances use conventional names for chemical entities derived from Latin, their names differ from Romanian to English (for instance, ”acidum ascorbicum” in the Romanian ontology is equivalent with ”ascorbic acid” in DINTO). This alignment was done in 2 phases. First, 500 active substances were merged because they either represented perfect matches, or they matched after applying a small set of conventional changes (e.g., removing the ”-um” prefix from the Romanian version). Second, the remaining 800 active substances were matched by analyzing the closest correspondent in the other ontology in terms of Levenshtein distance (Levenshtein, 1966), followed by a manual validation of the match.

3.1.2 Information Regarding Symptoms and Diseases

The English DOID and SYMP ontologies were used as a starting point (see Figure 6). Both the names of the symptoms and of the diseases were translated into Romanian via Google Translate, and then they were manually corrected in order to eliminate mistakes. Names containing polysemic words proved to be most difficult for the automated translation – for instance, a symptom describing a change in the pupil, was mistakenly interpreted as something related to a school student, not to the human eye.

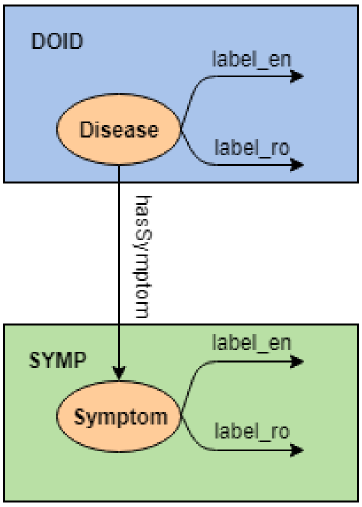


Figure 6: Romanian disease ontology - simplified view.

In total, names for over 900 symptoms and 10000 diseases were translated, allowing Romanian users to use the full benefits offered by the DOID ontology, without the need of English proficiency.

3.1.3 Connecting Symptoms to Drugs

There are no datasets containing information on what drugs should be administered for certain symptom in Romanian language. However, a usage section exists in each medical leaflet describing cases in which the drug should be taken (e.g., to numb pain, to reduce coughing, to lower body temperature and eliminate fever, etc.). Thus, the usage section for each of the 1138 leaflets was extracted from the ANMDM website and was indexed into Elasticsearch (Divya and Goyal, 2013), allowing users to find drugs with the usage description that most closely fits the set of provided symptoms.

3.2 Architecture and Processing Pipeline

The knowledge base can be hosted on a single server consisting of two different applications (see Figure 7). First, an instance of a Fuseki semantic repository server (Jena, 2019) was used to host the aforementioned ontologies: the Romanian drug ontology, DINTO, and a merge between DOID and SYMP called DOID-merged, to which we have added extra labels for Romanian. This repository allows users to query the ontologies via the SPARQL query language (Prud'hommeaux and Seaborne, 2008). Second, an Elasticsearch instance stores unstructured leaflets and can be queried in order to find drugs that are most likely helpful in relieving one or more symptoms.

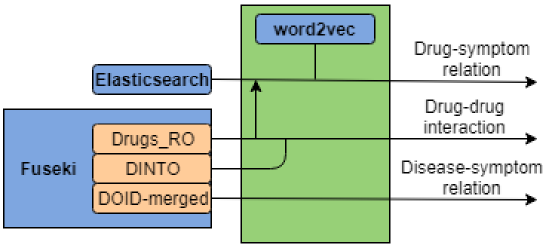


Figure 7: Knowledge base architecture.

On top of the two applications that act as information sources, a user interface allows users lacking experience on semantic web technologies to easily access the information. The user requests made at these endpoints are transformed into valid SPARQL and Elasticsearch queries. Furthermore, this interface can act as an autocorrect, by suggesting symptoms and drugs present in our dataset.

In the case of querying for the most helpful drugs given a set of symptoms, the relevance of the Elasticsearch information retrieval is improved by adding semantic information. For a given set of candidate texts from leaflets, each corresponding to a different drug, a word2vec (Mikolov et al., 2013) model trained on a 1-billion word Romanian corpus is used to compute the semantic similarity between it and the given set of symptoms. This is done by computing aggregate embeddings for the two text representations, and then computing cosine similarity. The Elasticsearch query score and the cosine similarity are both min-max normalized with regard to the set of candidate symptom-drug pairs extracted with Elasticsearch, their average is computed, and the candidate having the best average score is selected in the end.

Extra functionalities are added on top of the Fuseki repository and Elasticsearch server, such as suggesting possible matches when the string representing a searched drug/disease/symptom was not found. These suggestions are made by using a Levenshtein edit distance. In the case of symptoms, however, the same sensation can be expressed in multiple ways; thus, we rely on a very strict Levenshtein distance search to account for small typing errors. If this fails, a semantic search is used, looking for the symptom in the knowledge base for which the word2vec embedding is closest to the embedding of the input text.

4 Results

4.1 Drug Information and Drug-Drug Interactions

Our knowledge base offers access to structured information regarding the 15,093 drugs and 1,330 active substances. The information regarding drugs includes both numeric attributes (e.g., time until expiration), as well as text attributes (e.g., usage recommendations which were not standardized as format).

Users can search for the list of drugs with which any given one may interact because the Romanian drug ontology was aligned with DINTO. This search is done at active substance level. In case of drugs containing a combination of active substances, we consider that drugs A and B may interact if, for at least one active substance from A, there is at least one active substance from B with which it interacts. For example, if the user wants to check the interaction between *"OTOTIS"* which is based on a combination of two active ingredients (namely *"ciprofloxacinum"* and *"fluocinolonum"*) and *"ENAFILZIL"* which is based on *"sildenafilum"*, the knowledge base will search if either *"ciprofloxacinum"* or *"fluocinolonum"* interact with *"sildenafilum"*. As *"ciprofloxacinum"* interacts with *"sildenafilum"*, the system will conclude that the 2 drugs interact. If the same user wants to find the list of all the drugs which interact with *"OTOTIS"*, a list of all the drugs containing at least one active substance that interacts with either of *"ciprofloxacinum"* or *"fluocinolonum"* is generated and it will contain 35 entries.

4.2 Symptom-Disease Information

Our consolidated knowledge base contains 900 symptoms and 10,000 diseases from DOID and SYMP with names translated into Romanian language. The user can enter a list of symptoms to search for the disease that matches most symptoms. As mentioned before, if a symptom does not exist, two sequential attempts are performed to find its closest correspondent in our knowledge base. First, symptoms with a Levenshtein distance of 2 or less are searched. Second, if no result is found in the previous step, a word2vec embedding of the input string is computed, and the symptom from the knowledge base having the closest embedding to it is considered its equivalent.

For example, if a user searches for diseases that have *"mic de statură"* (eng, *"short stature"*) as a symptom, the results would be *"trichorhinophalangeal syndrome type II"*, *"Albright's hereditary osteodystrophy"*, *"spondyloepimetaphyseal dysplasia, strudwick type"* or *"Renpenning syndrome"*, as these are the only diseases linked to that symptom according to DOID and SYMP. If users make 1-2 typos when writing the symptom, the most relevant symptoms is suggested, and they can redo the search with the correct version. If they enter *"corp mic"* (eng, *"small body"*) or *"scund"* (eng, *"short"*), the first recommendation would fail, but the second one would suggest "short stature" as the most similar symptom based on a semantic similarity search.

4.3 Connecting Symptoms to Relevant Drugs

Considering a search for a combination of *"tuse, febră, durere de cap"* (eng, *"coughing, fever, headache"*), several types of analgesics, aspirin, paracetamol, and 2 types of cough syrup are recommended. The bottom results focus on yellow fever or other diseases that contain only a part of the symptoms specified as input.

In most simple use cases, the first entries are relevant. However, the symptom of a disease can be, in some cases, the effect of a drug that is targeted against a totally different disease. For instance, if the user searches symptoms related to diarrhea, such as *"scaun moale"* (eng, *"loose stool"*), some of the first drugs to be recommended are laxatives, but this would not be the wisest choice of medication. This happens because the effect of the medicine is, in some cases, mentioned alongside with the symptoms it should alleviate.

5 Conclusions

This paper presents a medical knowledge base for Romanian language focused on drugs. It is built using medical leaflets in Romanian and structured information regarding the drugs that was extracted from the ANMDM website. The knowledge base is also integrated with English ontologies in order to make more powerful inferences, such as searching for drug-drug interactions. The provided information can be structured into three main categories: drug related information, disease-symptom information, and drug-symptom information. To our knowing, this is the most comprehensive effort of building a knowledge base for Romanian drugs, their counter-indications, as well as potential relations to exhibited conditions.

The drug related information was extracted directly from official Romanian sources, thus it can be considered reliable. In order to keep the knowledge base up to date, the sources need to be crawled periodically in order to ensure that new information is always added, and that deprecated records are eliminated promptly. Information concerning drug-drug interactions is based on the DINTO ontology, which was last updated in 2016 and is still relevant. Nevertheless, we warn users that the information presented by our services is not a valid substitute for the opinion of a medical professional or a pharmacist. In the future, apart from drug-drug interactions, the knowledge base could also take into account pre-existing conditions or dietary choices which may interact with a certain treatment scheme. Part of this information is already available in DINTO, but it would need to be translated and integrated in our knowledge base.

The disease-symptom information is based on the DOID and SYMP ontologies. The name of the diseases and symptoms were automatically translated using Google Translate, and then manually corrected, if necessary. The two ontologies are still actively maintained; thus, the knowledge base needs to refresh this information from time to time in order to get the latest version. As is the case with the previous category, this information is not exhaustive and cannot substitute the knowledge of a professional.

The drug-symptom information is based only on medical leaflets crawled from the ANMDM website, which needs to be updated every several weeks. In some cases, the drug-symptom queries are very effective, for instance when searching for drugs targeting flu-like symptoms, such as fever and coughing. In other cases, the queries mistake the symptoms that the drug should address, with the drug's effect. These types of mistakes cannot be avoided for the time being due to manner in which the information was indexed. If more structured information regarding the drug-symptom relation could be extracted from the leaflets, either manually or by using different NLP techniques, these outlier cases would be addressed.

Our knowledge base provides real aid for Romanian users requiring drug-related information, and no similar initiatives exist at national level. The system cannot substitute the knowledge of a professional, and there are still problems to be addressed, but it is still an easy-to-use and useful tool for informing a user on medical treatments. Further improvements will be explored, including the orientation towards a personal health assistant for drug administration, similar in some degree to Babylon Health AI (https://www.babylonhealth.com/ai).

6 Future Work

In the future, we aim to expand even further our knowledge base. This can be done by indexing medical leaflets from other drug producers, as well as extracting more complex information from leaflets - for instance, contraindications expressed as rules (e.g. do not take certain antibiotics, such as tetracycline, with milk, other dairy products, calcium supplements, or antacids). Furthermore, we aim to integrate our knowledge base with other information sources, such as the Unified Medical Language System (Bodenreider, 2004).

7 Acknowledgements

The work presented in this paper has been funded by the "Intelligent platform for drugs administration – PIAM", subsidiary contract no. 1267/22.01.2018, from the NETIO project ID: P_40_270, MySMIS Code: 105976.

References

Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American* 284(5):34–43.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web* 7(3):154–165. https://doi.org/10.1016/j.websem.2009.07.002.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research* 32(suppl_1):D267–D270.

Manda Sai Divya and Shiv Kumar Goyal. 2013. Elasticsearch: An advanced and quick search technique to handle voluminous data. *Compusoft* 2(6):171.

Janna Hastings, Gareth Owen, Adriano Dekker, Marcus Ennis, Namrata Kale, Venkatesh Muthukrishnan, Steve Turner, Neil Swainston, Pedro Mendes, and Christoph Steinbeck. 2015. Chebi in 2016: Improved services and an expanding collection of metabolites. *Nucleic acids research* 44(D1):D1214–D1219.

María Herrero-Zazo, Isabel Segura-Bedmar, Janna Hastings, and Paloma Martínez. 2015. Dinto: using owl ontologies and swrl rules to infer drug–drug interactions and their mechanisms. *Journal of chemical information and modeling* 55(8):1698–1707.

Apache Jena. 2019. Apache jena fuseki documentation. http://jena.apache.org/documentation/fuseki2/.

Warren A Kibbe, Cesar Arze, Victor Felix, Elvira Mitraka, Evan Bolton, Gang Fu, Christopher J Mungall, Janos X Binder, James Malone, Drashtti Vasant, et al. 2014. Disease ontology 2015 update: an expanded and updated database of human diseases for linking biomedical knowledge through disease data. *Nucleic acids research* 43(D1):D1071–D1078.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*. volume 10, pages 707–710.

Deborah L McGuinness, Frank Van Harmelen, et al. 2004. Owl web ontology language overview. *W3C recommendation* 10(10):2004.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv* http://arxiv.org/abs/1301.3781.

Eric Prud'hommeaux and Andy Seaborne. 2008. SPARQL Query Language for RDF. *W3C Recommendation* 2009(January):1–106. http://www.w3.org/TR/rdf-sparql-query/.

Cornelius Rosse and José LV Mejino Jr. 2003. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of biomedical informatics* 36(6):478–500.

Lynn M Schriml, Cesar Arze, Suvarna Nadendla, Anu Ganapathy, Victor Felix, Anup Mahurkar, Katherine Phillippy, Aaron Gussman, Sam Angiuoli, Elodie Ghedin, et al. 2009. Gemina, genomic metadata for infectious agents, a geospatial surveillance pathogen database. *Nucleic acids research* 38(suppl_1):D754–D764.

Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. 2007. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology* 25(11):1251.

Summary Refinement through Denoising

Nikola I. Nikolov, Alessandro Calmanovici, Richard H.R. Hahnloser
Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland
{niniko, dcalma, rich}@ini.ethz.ch

Abstract

We propose a simple method for post-processing the outputs of a text summarization system in order to refine its overall quality. Our approach is to train text-to-text rewriting models to correct information redundancy errors that may arise during summarization. We train on synthetically generated noisy summaries, testing three different types of noise that introduce out-of-context information within each summary. When applied on top of extractive and abstractive summarization baselines, our summary denoising models yield metric improvements while reducing redundancy.[1]

1 Introduction

Text summarization aims to produce a shorter, informative version of an input text. While extractive summarization only selects important sentences from the input, abstractive summarization generates content without explicitly re-using whole sentences (Nenkova et al., 2011). In recent years, a number of successful approaches have been proposed for both extractive (Nallapati et al., 2017; Narayan et al., 2018) and abstractive (Chen and Bansal, 2018; Gehrmann et al., 2018) summarization paradigms. Despite these successes, many state-of-the-art systems remain plagued by overly high output redundancy (See et al. (2017); see Figure 3), which we set out to reduce.

In this paper, we propose a simple method (Figure 1, Section 3) for post-processing the outputs of a text summarization system in order to improve their overall quality. Our approach is to train dedicated text-to-text rewriting models to correct errors that may arise during summarization, focusing specifically on reducing information redundancy within each individual summary. To achieve this, we synthesize from clean summaries noisy summaries that contain diverse information redundancy errors, such as sentence repetition and out-of-context information (Section 3.2).

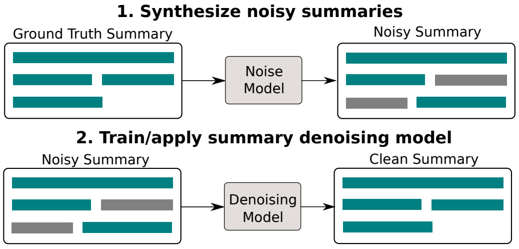


Figure 1: Overview of our approach to summary denoising. We alter ground truth summaries to generate a noisy dataset, on which we train denoising models to restore the original summaries.

In our experiments (Section 5), we show that denoising yields metric improvements and reduces redundancy when applied on top of several extractive and abstractive baselines. The generality of our method makes it a useful post-processing step applicable to any summarization system, that standardizes the summaries and improves their overall quality, ensuring fewer redundancies across the text.

2 Background

Post-processing of noisy human or machine-generated text is a topic that has recently been gathering interest. Automatic error correction (Rozovskaya and Roth, 2016; Xie et al., 2018) aims to improve the grammar or spelling of a text. In machine translation, automatic post editing of translated outputs (Chatterjee et al., 2018) is commonly used to further improve the translation quality, standardise the translations, or adapt them

[1] Code available at https://github.com/ninikolov/summary-denoising.

to a different domain (Isabelle, 2007).

In (Xie et al., 2018), authors synthesize grammatically incorrect sentences from correct ones using backtranslation (Sennrich et al., 2016a), which they use for grammar error correction. They enforce hypothesis variety during decoding by adding noise to beam search. Another work that is close to ours is (Fevry and Phang, 2018), where authors introduce redundancy on the word level in order to build an unsupervised sentence compression system. In this work, we take a similar approach, but instead focus on generating information redundancy errors on the sentence rather than the word level.

3 Approach

Our approach to summary refinement consists of two steps. First, we use a dataset of clean ground truth summaries to **generate** noisy summaries using several different types of synthetic noise. Second, we train text rewriting models to correct and **denoise** the noisy summaries, restoring them to their original form. The learned denoising models are then used to post-process and refine the outputs of a summarization system.

3.1 Generating Noisy Summaries

To generate noisy datasets, we rely on an existing parallel dataset of articles and clean ground truth summaries $S = \{s_0, , ..., s_j\}$. We iterate over each of the summaries and perturb them with noise, according to a *sentence noise distribution* $p_{noise} = [p_0, p_1, ..., p_N]$. p_{noise} defines the probability of adding noise to a specific number of sentences within each summary (from 0 up to a maximum of N noisy sentences), with $\sum p_{noise} = 1$.

For all experiments in this work, we use $p_{noise} = [0.15, 0.85]$ in order to ensure consistency, meaning that ˜15% of our noisy summaries contain no noisy sentences, while ˜85% contain one noisy sentence. Initial experiments showed that distributions which enforce larger or smaller amounts of noise lead to stronger or weaker denoising effects. Our choice of noise distribution showed good results on the majority of systems that we tested; we leave a more rigorous investigation of the choice of distribution to future work.

In addition to adding noise, we generate 3 noisy summaries for each clean summary by picking multiple random sentences to noise. This step increases the dataset size while introducing variety.

3.2 Types of Noise

We experiment with three simple types of noise, all of which introduce *information redundancy* into a summary. Our aim is to train denoising models that minimize repetitive or peripheral information within summaries.

Repeat picks random sentences from the summary and repeats them at the end. Repetition of phrases or even whole sentences is a problem commonly observed in text generation with RNNs (See et al., 2017), which motivates efforts to detect and minimize repetitions.

Replace picks random sentences from the summary, and replaces them with the closest sentence from the article. This type of noise helps the model to learn to *refine* sentences from the generated summaries, paraphrasing sentences when they are too long or contain redundant information.

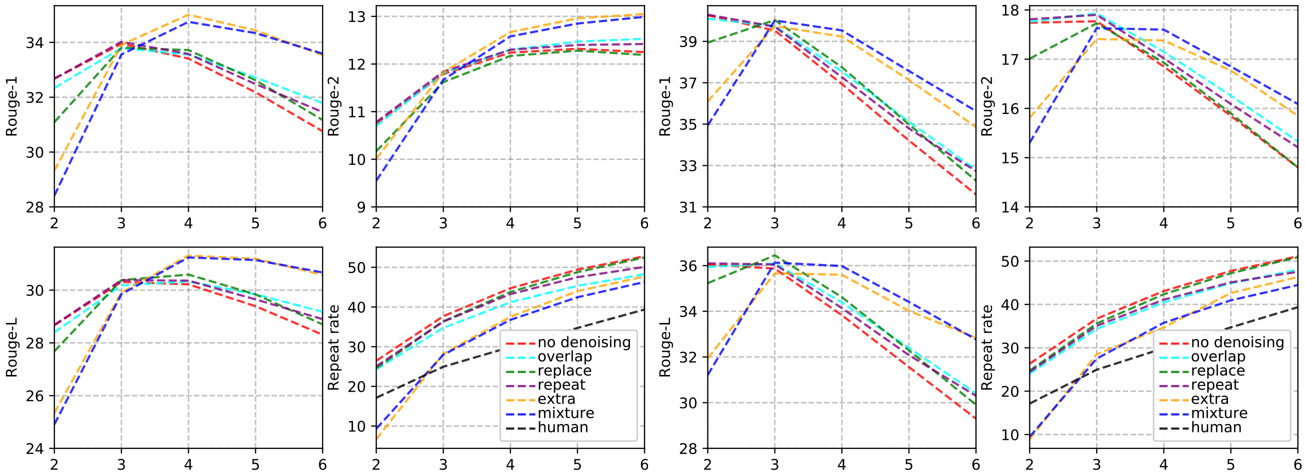
Extra picks random sentences from the article, paraphrases them, and inserts them into the summary, preserving the order of the sentences as they appear in the article. With this type of noise, a model learns to delete sentences which are out of context or contain redundant information. To paraphrase the sentences, we use the sentence paraphrasing model from (Chen and Bansal, 2018), trained on matching sentence pairs from the CNN/Daily Mail dataset.

Mixture mixes all the above noise types uniformly into a single dataset, keeping the same dataset size as for the individual noise types. With mixture, we explore whether the benefits of each noise type can be combined into a single model.

4 Experimental Setup

Dataset We use the CNN/Daily Mail dataset[2] (Hermann et al., 2015) of news articles and summaries in the form of bullet points, and follow the preprocessing pipeline from (Chen and Bansal, 2018). We use the standard split of the dataset, consisting of 287k news-summary pairs for training and 13k pairs for validation. We follow Section 3.1 to generate noisy versions of the datasets to be used during training. During testing, instead of clean summaries that contain noisy sentences, we input summaries produced by existing extractive or abstractive summarization systems.

[2] https://github.com/abisee/cnn-dailymail



(a) Denoising the LexRank system.

(b) Denoising the RNN-Ext system.

Figure 2: Metric results (Rouge-1/2/L and Repeat rate) on denoising extractive summarization systems. The x-axis in all plots is the number of extracted sentences. `human` is the result of the ground truth summaries (only for the Repeat rate).

Denoising models For all of our denoising experiments, we use a standard bidirectional LSTM encoder-decoder model (Sutskever et al., 2014) with 1000 hidden units and an attention mechanism (Bahdanau et al., 2014), and train on the subword-level (Sennrich et al., 2016b), capping the vocabulary size to 50k tokens for all experiments[3]. We train all models until convergence using the Adam optimizer (Kingma and Ba, 2015).

In addition to our neural denoising models, we implement a simple denoising baseline, `overlap`, based on unigram overlap between sentences in a summary. `overlap` deletes sentences which overlap more than 80%[4] with any other sentence in the summary and can therefore be considered as redundant.

Evaluation We report the ROUGE-1/2/L metrics (Lin, 2004). We also report the *Repeat* rate (Nikolov et al., 2018) $rep(\mathbf{s}) = \frac{\sum_i o(\overline{s_i}, s_i)}{|\mathbf{s}|}$ which is the average unigram overlap o of each sentence s_i in a text with the remainder of the text (where $\overline{s_i}$ denotes the complement of sentence s_i). Since the repeat rate measures the overlapping information across all sentences in a summary, lower values signify that a summary contains many unique sentences, while higher values indicate potential information repetition or redundancy within a summary.

5 Results

5.1 Extractive Summarization

We experiment with denoising two extractive systems: LexRank (Erkan and Radev, 2004) is an unsupervised graph-based approach which measures the centrality of each sentence with respect to the other sentences in the document. RNN-Ext is a more recent supervised LSTM sentence extractor module from (Chen and Bansal, 2018), trained on the CNN/Daily Mail dataset. It extracts sentences from the article sequentially. Both extractive systems require the number of sentences to be extracted to be given as a hyperparameter, in our experiments we test with summary lengths ranging from 2 to 6 sentences[5].

The results on extractive summarization are in Figure 2a for *LexRank* and Figure 2b for *RNN-Ext*, where we plot the metric scores for varying numbers of extracted sentences for each of the two systems. For both LexRank and RNN-Ext, we observe ROUGE improvements after denoising over the baseline systems without denoising. The `repeat` and `replace` methods yielded more modest improvements of 0.5-1 ROUGE-L points, performing comparably to the simple `overlap` baseline. The most effective noise types are `extra` and `mixture`, yielding improvements of up to 2 ROUGE-L points for LexRank and up to 3.5 ROUGE-L points for RNN-Ext. The superior performance to `overlap` indicates that the addi-

[3] We use the `fairseq` library `https://github.com/pytorch/fairseq`

[4] We empirically found that this threshold is sufficiently high to prevent unnecessary deletion and sufficiently low to detect near-identical sentences.

[5] The average sentence count of a summary in the CNN/Daily Mail dataset is 3.88.

System	Denoising approach	ROUGE-1	ROUGE-2	ROUGE-L	Repeat	#Sent	#Tok
Human	-	-	-	-	28.86	3.88	61.21
Article	-	14.95	8.54	14.41	70.5	26.9	804
Article	Mixture	30.47	13.97	28.24	53.43	10.67	304.7
RNN	-	35.61	15.04	32.7	51.9	2.93	58.46
RNN	Overlap	36.41	15.92	33.73	26.84	2.39	47.31
RNN	Repeat	**36.5**	**15.94**	**33.79**	27.65	2.41	48.34
RNN	Replace	35.2	14.86	32.4	51.51	2.98	57.0
RNN	Extra	33.95	14.58	31.2	37.19	2.21	42.82
RNN	Mixture	35.08	15.3	32.44	27.27	2.2	42.14
RNN-RL	-	**40.88**	**17.8**	**38.54**	39.29	4.93	72.82
RNN-RL	Overlap	40.76	17.69	38.43	37.71	4.83	71.02
RNN-RL	Repeat	40.84	17.76	38.49	38.78	4.86	71.69
RNN-RL	Replace	40.78	17.72	38.46	39.24	4.93	72.2
RNN-RL	Extra	39.12	16.7	36.76	34.04	3.84	55.43
RNN-RL	Mixture	40.11	17.33	37.76	35.45	4.18	61.15

Table 1: Results on denoising abstractive summarization. **Repeat** is the Repeat rate, while **#Sent** and **#Tok** are the average numbers of sentences or tokens in the summaries. Best **ROUGE** results for each model are in bold. *Human* is the result of the ground truth summaries, while *Article* uses the original article as the summary.

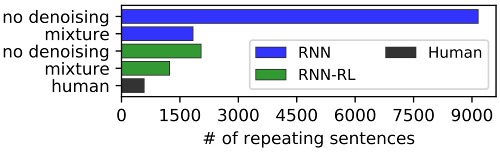


Figure 3: Number of sentence repetitions before and after denoising.

tional denoising operations learned by our models (see Figure 4a) are beneficial and can lead to more polished summaries that also may contain abstractive elements.

The gains from denoising are greater for longer summaries of more than two sentences. Long summaries are more likely to be affected by redundancy. For shorter summaries, denoising might lead to deletion of important information, thus denoising needs to be applied more carefully in such cases. Furthermore, for all sentence lengths and noise types, we observe a reduction in the Repeat rate after denoising, demonstrating that our approach is effective at reducing redundancy.

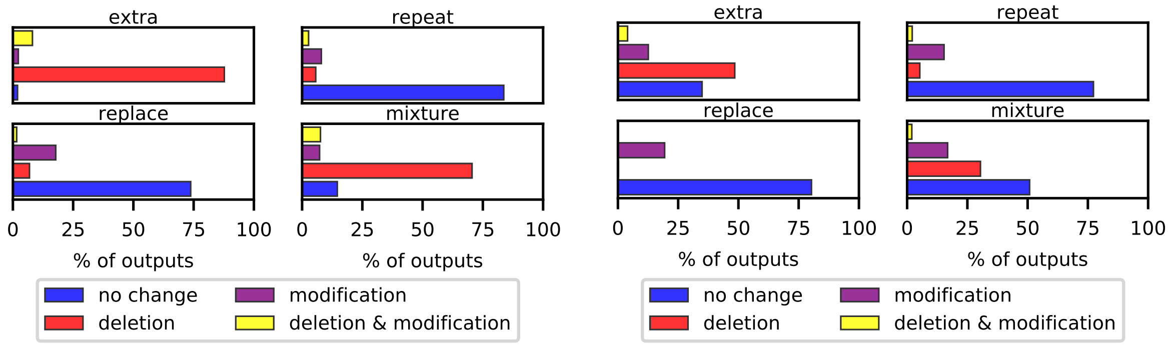
In Table 1, we additionally include the result from using the whole articles (*Article*) as input to our `mixture` model. Denoising is effective in this case, indicating that our approach may be promising for developing abstractive summarization systems that are fully unsupervised, similar to recent work in unsupervised sentence compression (Fevry and Phang, 2018).

5.2 Abstractive Summarization

For abstractive summarization, we test two systems. The first is a standard LSTM encoder-decoder model with an attention mechanism (*RNN*), identical to our denoising network from Section 4. The second, *RNN-RL*, is a state-of-the-art abstractive system proposed in (Chen and Bansal, 2018) that combines extractive and abstractive summarization using reinforcement learning. We train *RNN* ourselves, while for *RNN-RL*, we use the outputs provided by the authors.

Our metric results from denoising abstractive summarization are in Table 1. In Figure 3, we also compute the approximate number of sentence repetitions on the test set, by calculating the number of sentences that overlap significantly ($> 80\%$) with at least one other sentence in the summary.

For the *RNN* model, the `repeat` noise helps to remove repetition, halving our repetition metric, while boosting the ROUGE scores. This result is similar to our much simpler `overlap` baseline based on sentence deletion. The other noise types help to reduce redundancy, bringing the *Repeat* rate closer to that of *Human* summaries. This, however, comes at the cost of a decrease in ROUGE. For *RNN-RL*, while denoising helps to reduce repetition, none of our noise types managed to yield ROUGE improvements. One reason for this may be that this model already comes with a built-in mechanism for reducing redundancy which relies on sentence reranking (Chen and Bansal, 2018). However, as shown in Figure 3 (and in our example in Table 2), this model still generates many more sentence repetitions than found in human summaries. In overall, our approach is effective at reducing redundant information in abstractive summaries, how-



(a) RNN-Ext extractive system, extracting 5 sentences.

(b) RNN abstractive system.

Figure 4: Types of denoising operations applied to an extractive (left) and an abstractive (right) system (averaged over our test set).

ever this comes with a potential loss of information, which can lead to a reduction in ROUGE. Thus, our denoising methods are currently better suited for extractive than for absctractive summarization. Our work therefore calls for the development of novel types of synthetic noise that target abstractive summarization.

5.3 Analysis of Model Outputs

In Figure 4, we quantify the types of operations (deletion or modification of one or more sentences, or no change) our denosing models performed on the summaries produced by the extractive *RNN-Ext* (Figure 4a) and abstractive *RNN* system (Figure 4b). The `replace` and `repeat` noises are the most conservative, leaving over 75% of the summaries unchanged. `extra` is the most prone to delete sentences, while `repeat` and `replace` are most prone to modify sentences. We see a similar pattern for both extractive and abstractive summarization, with an increase of deletion for longer summaries produced by the extractive system. This indicates that our approach flexibly learns to switch between operations depending on the properties of the noisy input summary.

In Table 2 we show example outputs from denoising extractive and abstractive summaries produced for a sports article from our test set. All baseline summarization systems produced outputs that contain redundancy: for example, the first three sentences generated by the *RNN* system, and the 3rd and 4th sentences produced by the *RNN-RL* system are almost identical. To denoise the summaries, our models used diverse operations such as deletion of one or two sentences (e.g. *RNN* system, *Repeat* noise), rewriting (e.g. *RNN-RL* system, *Replace* noise, where *”dinorah santana , the player s agent , said her client had rejected the offer of a three-year contract extension”* is paraphrased to *”the player s agent said she had rejected the offer of a three-year contract”*), or even a combination of deletion and rewriting (e.g. *RNN-RL* system, *Repeat* noise).

6 Conclusion

We proposed a general framework for improving the outputs of a text summarization system based on denoising. Our approach is independent of the type of the system, and is applicable to both abstractive and extractive summarization paradigms. It could be useful as a post-processing step in a text summarization pipeline, ensuring that the summaries meet specific standards related to length or quality.

Our approach is effective at reducing information repetition present in existing summarization systems, and can even lead to ROUGE improvements, especially for extractive summarization. Denoising abstractive summarization proved to be more challenging, and our simple noise types did not yield significant ROUGE improvements for a state-of-the-art system. Our focus in future work, will, therefore, be to estimate better models of the noise present in abstractive summarization, to reduce information redundancy without a loss in quality, as well as to target other aspects such as the grammaticality or cohesion of the summary.

<table>
<tr><td colspan="3"><b>Ground truth</b> <i>(Rep=38.38)</i><b>:</b>
1. dani alves has spent seven seasons with the catalan giants
2. alves has four spanish titles to his name with barcelona
3. the brazil defender has also won the champions league twice with barca</td></tr>
<tr><td><b>RNN-Ext-4</b></td><td><b>RNN</b></td><td><b>RNN-RL</b></td></tr>
<tr><td><i>No denoising (R-1=33.6,Rep=45):</i>
1. dani alves looks set to leave barcelona this summer after his representative confirmed the brazilian right-back had rejected the club 's final contract offer
2. alves has enjoyed seven successful years at barcelona , winning four spanish titles and the champions league twice
3. but the 31-year-old has been unable to agree a new deal with the catalan club and will leave the nou camp this summer
4. dinorah santana , the player 's agent and ex-wife , said at a press conference on thursday that her client had rejected the offer of a three-year contract extension , which was dependent on the player taking part in 60 per cent of matches for the club</td><td><i>No denoising (R-1=34,Rep=79.6):</i>
1. dani alves has been unable to agree a new deal with catalan club
2. the brazilian has been unable to agree a new deal with catalan club
3. alves has been unable to agree a new deal with catalan club
4. alves has been linked with a number of clubs including manchester united and manchester city</td><td><i>No denoising (R-1=31,Rep=51.6):</i>
1. dani alves looks set to leave barcelona this summer
2. alves has enjoyed seven successful years at barcelona
3. alves has been unable to agree a deal with the catalan club
4. the 31-year-old has been unable to agree a new deal
5. dinorah santana , the player 's agent , said her client had rejected the offer of a three-year contract extension</td></tr>
<tr><td><i>Replace (R-1=36.6,Rep=46.6):</i>
1. Same
2. Same
3. Same
4. the player 's agent and ex-wife said at a press conference on thursday that her client had rejected the offer of a three-year contract extension</td><td><i>Replace (R-1=34, Rep=79.6):</i>
1. Same
2. Same
3. Same
4. Same</td><td><i>Replace (R-1=31, Rep=52.6):</i>
1. Same
2. Same
3. Same
4. Same
5. the player 's agent said she had rejected the offer of a three-year contract</td></tr>
<tr><td><i>Repeat (R-1=33.6,Rep=45):</i>
1. Same
2. Same
3. Same
4. Same</td><td><i>Repeat (R-1=28, Rep=41.4):</i>
1. Same
2. Deleted
3. Deleted
4. Same</td><td><i>Repeat (R-1=24.2, Rep=36.1):</i>
1. Same
2. Same
3. Deleted
4. alves has been unable to agree a new deal
5. Same</td></tr>
<tr><td><i>Extra (R-1=43.6,Rep=36.8):</i>
1. Same
2. Same
3. the 31-year-old has been unable to agree a new deal with the catalan club and will leave the nou camp this summer
4. Deleted</td><td><i>Extra (R-1=37.2,Rep=92.8):</i>
1. Same
2. Same
3. Same
4. Deleted</td><td><i>Extra (R-1=37, Rep=60.8):</i>
1. Same
2. Same
3. Same
4. Same
5. Deleted</td></tr>
<tr><td><i>Mixture (R-1=43, Rep=36.2):</i>
1. Same
2. Same
3. Same
4. Deleted</td><td><i>Mixture (R-1=28, Rep=41.43):</i>
1. Same
2. Deleted
3. Deleted
4. Same</td><td><i>Mixture (R-1=37, Rep=60.8):</i>
1. Same
2. Same
3. Same
4. Same
5. Deleted</td></tr>
</table>

Table 2: Examples for denoising extractive and abstractive summarization. *Same* indicates a summary sentence has been unchanged, while *Deleted* indicates sentence deletion. In brackets, *R-1* denotes the *Rouge-1* score, while *Rep* denotes the *Repeat* rate.

Acknowledgments

We acknowledge support from the Swiss National Science Foundation (grant 31003A_156976).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the wmt 2018 shared task on automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. pages 710–725.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22:457–479.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising autoencoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Association for Computational Linguistics, pages 413–422. http://aclweb.org/anthology/K18-1040.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 4098–4109. http://aclweb.org/anthology/D18-1443.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

P Isabelle. 2007. Domain adaptation of mt systems through automatic postediting. *Proc. 10th Machine Translation Summit (MT Summit XI), 2007* .

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* .

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 1747–1759. https://doi.org/10.18653/v1/N18-1158.

Ani Nenkova, Sameer Maskey, and Yang Liu. 2011. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts of ACL 2011*. Association for Computational Linguistics, page 3.

Nikola Nikolov, Michael Pfeiffer, and Richard Hahnloser. 2018. Data-driven summarization of scientific articles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Paris, France.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 2205–2215.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proc. of ACL*. Association for Computational Linguistics, pages 86–96. https://doi.org/10.18653/v1/P16-1009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proc. of ACL*. Association for Computational Linguistics, pages 1715–1725. https://doi.org/10.18653/v1/P16-1162.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 619–628. https://doi.org/10.18653/v1/N18-1057.

Large-Scale Hierarchical Alignment for Data-driven Text Rewriting

Nikola I. Nikolov and **Richard H.R. Hahnloser**
Institute of Neuroinformatics, University of Zürich and ETH Zürich, Switzerland
{niniko, rich}@ini.ethz.ch

Abstract

We propose a simple unsupervised method for extracting pseudo-parallel monolingual sentence pairs from comparable corpora representative of two different text styles, such as news articles and scientific papers. Our approach does not require a seed parallel corpus, but instead relies solely on hierarchical search over pre-trained embeddings of documents and sentences. We demonstrate the effectiveness of our method through automatic and extrinsic evaluation on text simplification from the normal to the Simple Wikipedia. We show that pseudo-parallel sentences extracted with our method not only supplement existing parallel data, but can even lead to competitive performance on their own.[1]

1 Introduction

Parallel corpora are indispensable resources for advancing monolingual and multilingual text rewriting tasks. Due to the scarce availability of parallel corpora, and the cost of manual creation, a number of methods have been proposed that can perform large-scale sentence alignment: automatic extraction of *pseudo-parallel* sentence pairs from raw, comparable[2] corpora. While pseudo-parallel data is beneficial for machine translation (Munteanu and Marcu, 2005), there has been little work on large-scale sentence alignment for monolingual text-to-text rewriting tasks, such as simplification (Nisioi et al., 2017) or style transfer (Liu et al., 2016). Furthermore, the majority of existing methods (e.g. Marie and Fujita (2017); Grégoire and Langlais (2018)) assume access to some parallel training data. This impedes their application to cases where there is *no parallel data available whatsoever*, which is the case for the majority of text rewriting tasks, such as style transfer.

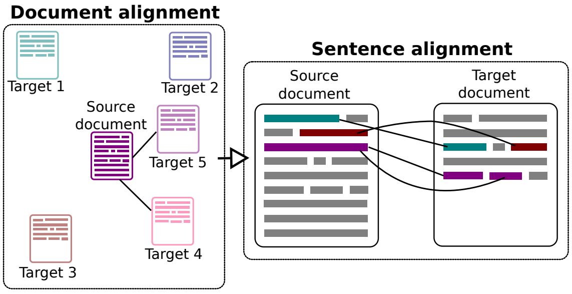


Figure 1: Illustration of large-scale hierarchical alignment (**LHA**). For each document in a *source* dataset, **document alignment** retrieves matching documents from a *target* dataset. In turn, **sentence alignment** retrieves matching sentence pairs from within each document pair.

In this paper, we propose a simple unsupervised method, Large-scale Hierarchical Alignment (**LHA**) (Figure 1; Section 3), for extracting pseudo-parallel sentence pairs from two raw monolingual corpora which contain documents in two different author styles, such as scientific papers and press releases. LHA hierarchically searches for document and sentence nearest neighbors within the two corpora, extracting sentence pairs that have high semantic similarity, yet preserve the stylistic characteristics representative of their original datasets. LHA is robust to noise, fast and memory efficient, enabling its application to datasets on the order of hundreds of millions of sentences. Its generality makes it relevant to a wide range of monolingual text rewriting tasks.

We demonstrate the effectiveness of LHA on automatic benchmarks for alignment (Section 4), as well as extrinsically, by training neural machine translation (NMT) systems on the task of text simplification from the normal Wikipedia to the Simple Wikipedia (Section 5). We show that

[1] Code available at https://github.com/ninikolov/lha.

[2] Corpora that contain documents on similar topics.

Proceedings of Recent Advances in Natural Language Processing, pages 844–853,
Varna, Bulgaria, Sep 2–4 2019.

pseudo-parallel datasets obtained by LHA are not only useful for augmenting existing parallel data, boosting the performance on automatic measures, but can even be competitive on their own.

2 Background

2.1 Data-Driven Text Rewriting

The goal of text rewriting is to transform an input text to satisfy specific constraints, such as simplicity (Nisioi et al., 2017) or a more general author style, such as political (e.g. democratic to republican) or gender (e.g. male to female) (Prabhumoye et al., 2018; Shen et al., 2017). Rewriting systems can be valuable when preparing a text for multiple audiences, such as simplification for language learners (Siddharthan, 2002) or people with reading disabilities (Inui et al., 2003). They can also be used to improve the accessibility of technical documents, e.g. to simplify terms in clinical records for laymen (Abrahamsson et al., 2014).

Text rewriting can be cast as a data-driven task in which transformations are learned from large collections of parallel sentences. Limited availability of high-quality parallel data is a major bottleneck for this approach. Recent work on Wikipedia and the Simple Wikipedia (Coster and Kauchak, 2011; Kajiwara and Komachi, 2016) and on the Newsela dataset of simplified news articles for children (Xu et al., 2015) explore supervised, data-driven approaches to text simplification. Such approaches typically rely on statistical (Xu et al., 2016) or neural (Štajner and Nisioi, 2018) machine translation.

Recent work on unsupervised approaches to text rewriting without parallel corpora is based on variational (Fu et al., 2017) or cross-aligned (Shen et al., 2017) autoencoders that learn latent representations of content separate from style. In (Prabhumoye et al., 2018), authors model style transfer as a back-translation task by translating input sentences into an intermediate language. They use the translations to train separate English decoders for each target style by combining the decoder loss with the loss of a style classifier, separately trained to distinguish between the target styles.

2.2 Large-Scale Sentence Alignment

The goal of **sentence alignment** is to extract from raw corpora sentence pairs suitable as training examples for text-to-text rewriting tasks such as machine translation or text simplification. When the documents in the corpora are *parallel* (labelled document pairs, such as identical articles in two languages), the task is to identify suitable sentence pairs from each document. This problem has been extensively studied both in the multilingual (Brown et al., 1991; Moore, 2002) and monolingual (Hwang et al., 2015; Kajiwara and Komachi, 2016; Štajner et al., 2018) case. The limited availability of parallel corpora led to the development of **large-scale sentence alignment** methods, which is also the focus of this work. The aim of these methods is to extract pseudo-parallel sentence pairs from raw, non-aligned corpora. For many tasks, millions of examples occur naturally within existing textual resources, amply available on the internet.

The majority of previous work on large-scale sentence alignment is in machine translation, where adding pseudo-parallel pairs to an existing parallel dataset has been shown to boost the translation performance (Munteanu and Marcu, 2005; Uszkoreit et al., 2010). The work that is most closely related to ours is (Marie and Fujita, 2017), where authors use pre-trained word and sentence embeddings to extract rough translation pairs in two languages. Subsequently, they filter out low-quality translations using a classifier trained on parallel translation data. More recently, (Grégoire and Langlais, 2018) extract pseudo-parallel translation pairs using a Recurrent Neural Network (RNN) classifier. Importantly, these methods assume that *some parallel training data is already available*, which impedes their application in settings where there is no parallel data whatsoever, which is the case for many text rewriting tasks such as style transfer.

There is little work on large-scale sentence alignment focusing specifically on monolingual tasks. In (Barzilay and Elhadad, 2003), authors develop a hierarchical alignment approach of first clustering paragraphs on similar topics before performing alignment on the sentence level. They argue that, for monolingual data, pre-clustering of larger textual units is more robust to noise compared to fine-grained sentence matching applied directly on the dataset level.

3 Large-Scale Hierarchical Alignment (LHA)

Given two datasets that contain comparable documents written in two different author styles: a

source dataset $\mathbf{S^d}$ consisting of N_S documents $\mathbf{S^d} = \{s^d_1, ..., s^d_{N_S}\}$ (e.g. all Wikipedia articles) and a **target** dataset $\mathbf{T^d}$ consisting of N_T documents $\mathbf{T^d} = \{t^d_1, ..., t^d_{N_T}\}$ (e.g. all articles from the Simple Wikipedia), our approach to large-scale alignment is hierarchical, consisting of two consecutive steps: **document** alignment followed by **sentence** alignment (see Figure 1).

3.1 Document Alignment

For each source document s^d_i, document alignment retrieves K nearest neighbours $\{t^d_{i_1}, ..., t^d_{i_K}\}$ from the target dataset. In combination, these form K pseudo-parallel document pairs $\{(s^d_i, t^d_{i_1}), ..., (s^d_i, t^d_{i_K})\}$. Our aim is to select document pairs with high semantic similarity, potentially containing good pseudo-parallel sentence pairs representative of the document styles of each dataset.

To find nearest neighbours, we rely on two components: document embedding and approximate nearest neighbour search. For each dataset, we pre-compute document embeddings $e_d()$ as $I_s = [e_d(s^d_1), ..., e_d(s^d_{N_S})]$ and $I_t = [e_d(t^d_1), ..., e_d(t^d_{N_T})]$. We employ nearest neighbour search methods[3] to partition the embedding space, enabling fast and efficient nearest neighbour retrieval of similar documents across I_s and I_t. This enables us to find K nearest *target* document embeddings in I_t for each *source* embedding in I_s. We additionally filter document pairs whose similarity is below a manually selected threshold θ_d. In Section 4, we evaluate a range of different document embedding approaches, as well as alternative similarity metrics.

3.2 Sentence Alignment

Given a pseudo-parallel document pair (s^d, t^d) that contains a **source** document $s^d = \{s^s_1, ..., s^s_{N_J}\}$ consisting of N_J sentences and a **target** document $t^d = \{t^s_1, ..., t^s_{N_M}\}$ consisting of N_M sentences, sentence alignment extracts pseudo-parallel sentence pairs (s^s_i, t^s_j) that are highly similar.

To implement sentence alignment, we first embed each sentence in s^d and t^d and compute an inter-sentence similarity matrix P among all sentence pairs in s^d and t^d. From P we extract K nearest neighbours for each source and each target sentence. We denote the nearest neighbours of s^s_i as $NN(s^s_i) = \{t^s_{i_1}, \ldots, t^s_{i_K}\}$ and the nearest neighbours of t^s_j as $NN(t^s_j) = \{s^s_{j_1}, \ldots, s^s_{j_K}\}$. We remove all sentence pairs with similarity below a manually set threshold θ_s. We then merge *all* overlapping sets of nearest sentences in the documents to produce pseudo-parallel sentence sets (e.g. $(\{s^s_e, s^s_i\}, \{t^s_j, t^s_k, t^s_l\})$ when source sentence i is closes to target sentences j, k, and l and target sentence j is closest to source sentences e and i). This approach, inspired from (Štajner et al., 2018), provides the flexibility to model multi-sentence interactions, such as sentence splitting or compression, as well as individual sentence-to-sentence reformulations. Note that when $K = 1$, we only retrieve individual sentence pairs.

The final output of sentence alignment is a list of pseudo-parallel sentence pairs with high semantic similarity and preserved stylistic characteristics of each dataset. The pseudo-parallel pairs can be used to either augment an existing parallel dataset (as in Section 5), or independently, to solve a new author style transfer task for which there is no parallel data available (see the supplementary material for an example).

3.3 System Variants

The aforementioned framework provides the flexibility of exploring diverse variants, by exchanging document/sentence embeddings or text similarity metrics. We compare all variants in an automatic evaluation in Section 4.

Text embeddings We experiment with four text embedding methods:

1. *Avg*, is the average of the constituent word embeddings of a text[4], a simple approach that has proved to be a strong baseline for many text similarity tasks.

2. In *Sent2Vec*[5] (Pagliardini et al., 2018), the word embeddings are specifically optimized towards additive combinations over the sentence using an unsupervised objective function. This approach performs well on many unsupervised and supervised text similarity tasks, often outperforming more sophisticated supervised recurrent or convolutional architectures, while remaining very fast to compute.

[3] We use the Annoy library `https://github.com/spotify/annoy`.

[4] We use the Google News 300-dim Word2Vec models.

[5] We use the public unigram Wikipedia model.

3. *InferSent*[6] (Conneau et al., 2017) is a supervised sentence embedding approach based on bidirectional LSTMs, trained on natural language inference data.

4. *BERT*[7] (Devlin et al., 2019) is a state-of-the-art supervised sentence embedding approach based on the Transformer architecture.

Word Similarity We additionally test four word-based approaches for computing text similarity. Those can be used either on their own, or to refine the nearest neighbour search across documents or sentences.

1. We compute the unigram string overlap $o(\mathbf{x}, \mathbf{y}) = \frac{|\{\mathbf{y}\} \cap \{\mathbf{x}\}|}{|\{\mathbf{y}\}|}$ between source tokens $\mathbf{x}$ and target tokens $\mathbf{y}$ (excluding punctuation, numbers and stopwords).

2. We use the *BM25* ranking function (Robertson et al., 2009), an extension of TF-IDF.

3. We use the Word Mover's Distance (*WMD*) (Kusner et al., 2015), which measures the distance the embedded words of one document need to travel to reach the embedded words of another document. WMD has recently achieved good results on text retrieval (Kusner et al., 2015) and sentence alignment (Kajiwara and Komachi, 2016).

4. We use the Relaxed Word Mover's Distance (*RWMD*) (Kusner et al., 2015), which is a fast approximation of the WMD.

4 Automatic Evaluation

We perform an automatic evaluation of LHA using an annotated sentence alignment dataset (Hwang et al., 2015). The dataset contains 46 article pairs from Wikipedia and the Simple Wikipedia. The 67k potential sentence pairs were manually labelled as either *good* simplifications (277 pairs), *good* with a *partial* overlap (281 pairs), *partial* (117 pairs) or *non-valid*. We perform three comparisons using this dataset: evaluating document and sentence alignment separately, as well as jointly.

For sentence alignment, the task is to retrieve the 277 good sentence pairs out of the $67k$ possible sentence pairs in total, while minimizing the number of false positives. To evaluate document alignment, we add 1000 randomly sampled articles from Wikipedia and the Simple Wikipedia as noise, resulting in 1046 article pairs in total. The goal of document alignment is to identify the original 46 document pairs out of 1046×1046 possible document combinations.

This set-up additionally enables us to **jointly** evaluate document and sentence alignment, which best resembles the target effort of retrieving good sentence pairs from noisy documents. The two aims of the joint alignment task are to identify the *good* sentence pairs from within either $1M$ document or $125M$ sentence pairs, in the latter case without relying on any document-level information whatsoever.

4.1 Results

Our results are summarized in Tables 1 and 2. For all experiments, we set $K = 1$ and report the maximum F1 score ($\mathbf{F1}_{max}$) obtained from varying the document threshold θ_d and the sentence threshold θ_s. We also report the percentage of true positive (**TP**) document or sentence pairs that were retrieved when the F1 score was at its maximum, as well as the average speed of each approach (**doc/s** and **sent/s**). The speed becomes of a particular concern when working with large datasets consisting of millions of documents and hundreds of millions of sentences.

On document alignment, (Table 1, left) the *Sent2Vec* approach achieved the best score, outperforming the other embedding methods including the word-based similarity measures. On sentence alignment (Table 1, right), the WMD achieves the best performance, matching the result from (Kajiwara and Komachi, 2016). When evaluating document and sentence alignment jointly (Table 2), we compare our hierarchical approach (*LHA*) to global alignment applied directly on the sentence level (*Global*). *Global* computes the similarities between all $125M$ sentence pairs in the entire evaluation dataset. LHA significantly outperforms Global, successfully retrieving three times more valid sentence pairs, while remaining fast to compute. This result demonstrates that document alignment is beneficial, successfully filtering some of the noise, while also reducing the overall number of sentence similarities to be computed.

The *Sent2Vec* approach to LHA achieves good performance on document and sentence align-

[6] We use the GloVe-based model provided by the authors.

[7] We use the base 12-layer model provided by the authors.

Table 1: Automatic evaluation of Document (left) and Sentence alignment (right). **EDim** is the embedding dimensionality. **TP** is the percentage of true positives obtained at $\mathbf{F1}_{max}$. Speed is calculated on a single CPU thread.

			Document alignment				**Sentence alignment**			
	Approach	**EDim**	$\mathbf{F1}_{max}$	**TP**	θ_d	**doc/s**	$\mathbf{F1}_{max}$	**TP**	θ_s	**sent/s**
Embedding	Average word embeddings (Avg)	300	0.66	43%	0.69	260	0.675	46%	0.82	1458
	Sent2Vec (Pagliardini et al., 2018)	600	**0.78**	**61%**	0.62	343	0.692	48%	0.69	1710
	InferSent† (Conneau et al., 2017)	4096	-	-	-	-	0.69	49%	0.88	110
	BERT† (Devlin et al., 2019)	768	-	-	-	-	0.65	43%	0.89	25
Word sim	Overlap	-	0.53	29%	0.66	120	0.63	40%	0.5	1600
	BM25 (Robertson et al., 2009)	-	0.46	16%	0.257	60	0.52	27%	0.43	20K
	RWMD (Kusner et al., 2015)	300	0.713	51%	0.67	60	0.704	50%	0.379	1050
	WMD (Kusner et al., 2015)	300	0.49	24%	0.3	1.5	**0.726**	**54%**	0.353	180
	(Hwang et al., 2015)	-	-	-	-	-	0.712	-	-	-
	(Kajiwara and Komachi, 2016)	-	-	-	-	-	0.724	-	-	-

†: These models are specifically designed for sentence embedding, hence we do not test them on document alignment.

Table 2: Evaluation on large-scale sentence alignment: identifying the *good* sentence pairs without any document-level information. We pre-compute the embeddings and use the Annoy ANN library. For the WMD-based approaches, we re-compute the top 50 sentence nearest neighbours of Sent2Vec.

Approach	$\mathbf{F1}_{max}$	**TP**	**time**
LHA (Sent2Vec)	0.54	31%	33s
LHA (Sent2Vec + WMD)	**0.57**	**33%**	1m45s
Global (Sent2Vec)	0.339	12%	15s
Global (WMD)	0.291	12%	30m45s

ment, while also being the fastest to compute. We therefore use it as the default approach for the following experiments on text simplification.

5 Empirical Evaluation

To test the suitability of pseudo-parallel data extracted with LHA, we perform empirical experiments on text simplification from the normal Wikipedia to the Simple Wikipedia. We chose simplification because some parallel data are already available for this task, allowing us to experiment with mixing parallel and pseudo-parallel datasets. In the supplementary material[8] we experiment with an additional task for which there is no parallel data: style transfer from scientific journal articles to press releases.

We compare the performance of neural machine translation (NMT) systems trained under three different scenarios: 1) using **existing** parallel data for training; 2) using a **mixture** of parallel and pseudo-parallel data extracted with LHA; and 3) using pseudo-parallel data **on its own**.

5.1 Experimental Setup

NMT model For all experiments, we use a single-layer LSTM encoder-decoder model (Cho et al., 2015) with an attention mechanism (Bahdanau et al., 2015). We train our models on the subword level (Sennrich et al., 2015), capping the vocabulary size to 50k. We re-learn the subword rules separately for each dataset, and train until convergence using the Adam optimizer (Kingma and Ba, 2015). We use beam search with a beam of 5 to generate all final outputs.

Evaluation metrics We report a diverse range of automatic metrics and statistics. *SARI* (Xu et al., 2016) is a recently proposed metric for text simplification which correlates well with simplicity in the output. SARI takes into account the total number of changes (additions, deletions) of the input when scoring model outputs. *BLEU* (Papineni et al., 2002) is a precision-based metric for machine translation commonly used for evaluation of text simplification (Xu et al., 2016; Štajner and Nisioi, 2018) and of style transfer (Shen et al., 2017). Recent work has indicated that BLEU is not suitable for assessment of simplicity (Sulem et al., 2018), it correlates better with meaning preservation and grammaticality, in particular when using multiple references. We also report the average Levenshtein distance (LD) from the model outputs to the input (LD_{src}) or the target reference (LD_{tgt}). On simplification tasks, LD correlates well with meaning preservation and grammaticality (Sulem et al., 2018), complementing BLEU.

Extracting pseudo-parallel data We use LHA with *Sent2Vec* (see Section 3) to extract pseudo-parallel sentence pairs for text simplification. To ensure some degree of lexical similarity, we exclude pairs whose string overlap (defined in Section 3.3) is below 0.4, and pairs in which the target sentence is more than 1.5 times longer than the source sentence. We use $K = 5$ in all of our align-

[8] Available in our arXiv paper at `https://arxiv.org/abs/1810.08237`

Table 3: Datasets used to extract pseudo-parallel monolingual sentence pairs in our experiments.

Dataset	Type	Documents	Tokens	Sentences	Tok. per sent.	Sent. per doc.
Wikipedia	Articles	5.5M	2.2B	92M	25 ± 16	17 ± 32
Simple Wikipedia	Articles	134K	62M	2.9M	27 ± 68	22 ± 34
Gigaword	News	8.6M	2.5B	91M	28 ± 12	11 ± 7

Table 4: Example pseudo-parallel pairs extracted by our Large-scale hierarchical alignment (LHA) method.

Dataset	Source	Target
wiki-simp-65	However, Jimmy Wales, Wikipedia's co-founder, denied that this was a crisis or that Wikipedia was running out of admins, saying, "The number of admins has been stable for about two years, there's really nothing going on."	But the co-founder Wikipedia, Jimmy Wales, did not believe that this was a crisis. He also did not believe Wikipedia was running out of admins.
wiki-news-74	Prior to World War II, Japan's industrialized economy was dominated by four major zaibatsu: Mitsubishi, Sumitomo, Yasuda and Mitsui.	Until Japan 's defeat in World War II , the economy was dominated by four conglomerates , known as " zaibatsu " in Japanese . These were the Mitsui , Mitsubishi , Sumitomo and Yasuda groups .

Table 5: Statistics of the pseudo-parallel datasets extracted with LHA. μ_{tok}^{src} and μ_{tok}^{tgt} are the mean src/tgt token counts, while $\%_{s>2}^{src}$ and $\%_{s>2}^{tgt}$ report the percentage of items that contain more than one sentence.

Dataset	Pairs	μ_{tok}^{src}	μ_{tok}^{tgt}	$\%_{s>2}^{src}$	$\%_{s>2}^{tgt}$
wiki-simp-72	25K	26.72	22.83	16%	11%
wiki-simp-65	80K	23.37	15.41	17%	7%
wiki-news-74	133K	25.66	17.25	19%	2%
wiki-news-70	216K	26.62	16.29	19%	2%

ment experiments, which enables extraction of up to 5 sentence nearest neighbours.

Parallel data As a parallel baseline dataset, we use an existing dataset from (Hwang et al., 2015). The dataset consists of 282K sentence pairs obtained after aligning the parallel articles from Wikipedia and the Simple Wikipedia. This dataset allows us to compare our results to previous work on data-driven text simplification. We use two versions of the dataset in our experiments: `full` contains all 282K pairs, while `partial` contains 71K pairs, or 25% of the full dataset.

Evaluation data We evaluate our simplification models on the testing dataset from (Xu et al., 2016), which consists of 358 sentence pairs from the normal Wikipedia and the Simple Wikipedia. In addition to the ground truth simplifications, each input sentence comes with 8 additional references, manually simplified by Amazon Meachanical Turkers. We compute *BLEU* and *SARI* on the 8 manual references.

Pseudo-parallel data We align two dataset pairs, obtaining pseudo-parallel sentence pairs for text simplification (statistics of the datasets we use for alignment are in Table 3). First, we align the normal `Wikipedia` to the `Simple Wikipedia` using document and sentence similarity thresholds $\theta_d = 0.5$ and $\theta_s = \{0.72, 0.65\}$, producing two datasets: `wiki-simp-72` and `wiki-simp-65`. Because LHA uses no document-level information in this dataset, alignment leads to new sentence pairs, some of which may be distinct from the pairs present in the existing parallel dataset. We monitor for and exclude pairs that overlap with the testing dataset. Second, we align `Wikipedia` to the `Gigaword` news article corpus (Napoles et al., 2012), using $\theta_d = 0.5$ and $\theta_s = \{0.74, 0.7\}$, resulting in two additional pseudo-parallel datasets: `wiki-news-74` and `wiki-news-70`. With these datasets, we investigate whether pseudo-parallel data extracted from a *different domain* can be beneficial for text simplification. We use slightly higher sentence alignment thresholds for the news articles because of the domain difference.

We find that the majority of the pairs extracted contain a single sentence, and 15-20% of the source examples and 5-10% of the target examples contain multiple sentences (see Table 5 for additional statistics). Most multi-sentence examples contain two sentences, while 0.5-1% contain 3 to 5 sentences. Two example aligned outputs are in Table 4 (additional examples are available in the supplementary material). They suggest that our method is capable of extracting high-quality pairs that are similar in meaning, even spanning across multiple sentences.

Randomly sampled pairs We also experiment with adding random sentence pairs to the parallel dataset (`rand-100K`, `rand-200K` and `rand-300K` datasets, containing 100K, 200K and 300K random pairs, respectively). The

Table 6: Empirical results on text simplification from Wikipedia to the Simple Wikipedia. The highest SARI/BLEU results from each category are in bold. `input` and `reference` are not generated using Beam Search.

Method or Dataset	Total pairs (% pseudo)	Beam hypothesis 1					Beam hypothesis 2				
		SARI	BLEU	μ_{tok}	LD_{src}	LD_{tgt}	SARI	BLEU	μ_{tok}	LD_{src}	LD_{tgt}
input	-	26	99.37	22.7	0	0.26	-	-	-	-	-
reference	-	38.1	70.21	22.3	0.26	0	-	-	-	-	-
NTS	282K (0%)	30.54	84.69	-	-	-	35.78	77.57	-	-	-
Parallel + Pseudo-parallel or Randomly sampled data (Using full parallel dataset, 282K parallel pairs)											
baseline-282K	282K (0%)	30.72	85.71	18.3	0.18	0.37	36.16	82.64	19	0.19	0.36
+ wiki-simp-72	307K (8%)	30.2	87.12	19.43	0.14	0.34	36.02	81.13	19.03	0.19	0.36
+ wiki-simp-65	362K (22%)	**30.92**	**89.64**	19.8	0.13	0.33	36.48	83.56	19.37	0.18	0.35
+ wiki-news-74	414K (32%)	30.84	89.59	19.67	0.13	0.33	36.57	**83.85**	19.13	0.18	0.35
+ wiki-news-70	498K(43%)	30.82	89.62	19.6	0.13	0.33	36.45	83.11	18.98	0.19	0.36
+ rand-100K	382K (26%)	30.52	88.46	19.7	0.14	0.34	**36.96**	82.86	19	0.2	0.36
+ rand-200K	482K (41%)	29.47	80.65	19.3	0.18	0.36	34.36	74.67	18.93	0.23	0.38
+ rand-300K	582K (52%)	28.68	75.61	19.57	0.23	0.4	32.34	68.9	18.35	0.3	0.43
Parallel + Pseudo-parallel data (Using partial parallel dataset, 71K parallel pairs)											
baseline-71K	71K (0%)	**31.16**	69.53	17.45	0.29	0.44	32.92	67.29	19.14	0.3	0.44
+ wiki-simp-65	150K (52%)	31.0	**81.52**	18.26	0.21	0.38	**35.12**	**77.38**	18.16	0.25	0.39
+ wiki-news-70	286K(75%)	31.01	80.03	17.82	0.23	0.4	34.14	76.44	17.31	0.28	0.43
Pseudo-parallel data only											
wiki-simp-all	104K (100%)	29.93	60.81	18.05	0.36	0.47	30.13	57.46	18.53	0.39	0.49
wiki-news-all	348K (100%)	22.06	28.51	13.68	0.6	0.63	23.08	29.62	14.01	0.6	0.64
pseudo-all	452K (100%)	**30.24**	**71.32**	17.82	0.3	0.43	**31.41**	**65.65**	17.65	0.33	0.45

random pairs are uniformly sampled from the Wikipedia and the Simple Wikipedia, respectively. With the random pairs, we aim to investigate how model performance changes as we add an increasing number of sentence pairs that are non-parallel but are still representative of the two dataset styles.

5.2 Automatic Evaluation

The simplification results in Table 6 are organized in several sections according to the type of dataset used for training. We report the results of the top two beam search hypotheses produced by our models, considering that the second hypothesis often generates simpler outputs (Štajner and Nisioi, 2018).

In Table 6, `input` is copying the normal Wikipedia input sentences, without making any changes. `reference` reports the score of the original Simple Wikipedia references with respect to the other 8 references available for this dataset. `NTS` is the previously best reported result on text simplification using neural sequence models (Štajner and Nisioi, 2018). `baseline-`{`282K, 71K`} are our parallel LSTM baselines, trained on 282K and 71K parallel pairs, respectively.

The models trained on a *mixture* of *parallel* and *pseudo-parallel* data generate longer outputs on average, and their output is more similar to the `input`, as well as to the original Simple Wikipedia `reference`, in terms of the LD. Adding pseudo-parallel data frequently yields BLEU improvements on both Beam hypotheses: over the *NTS* system, as well as over our baselines trained solely on parallel data. The BLEU gains are larger when using the smaller parallel dataset, consisting of 71K sentence pairs. In terms of SARI, the scores remain either similar or slightly better than the baselines, indicating that simplicity in the output is preserved. The second Beam hypothesis yields higher SARI scores than the first one, in agreement with (Štajner and Nisioi, 2018). Interestingly, adding out-of-domain pseudo-parallel news data (`wiki-news-`$\star$ datasets) results in an increase in BLEU despite the potential change in style of the target sequence.

Larger pseudo-parallel datasets can lead to bigger improvements, however noisy data can result in a decrease in performance, motivating careful data selection. In our *parallel and random* setup, we find that an increasing number of random pairs added to the parallel data progressively degrades model performance. However, those models still manage to perform surprisingly well, even when over half of the pairs in the dataset are random. Thus, neural machine translation can successfully learn target transformations despite substantial data corruption, demonstrating robustness to noisy or non-parallel data for certain tasks.

When training solely on *pseudo-parallel* data, we observe lower performance on average in comparison to parallel models. However, the results are encouraging, demonstrating the potential of our approach in tasks for which there is no parallel data available. As expected, the out-of-domain news data (`wiki-news-all`) is less suitable for simplification than the in-domain data (wiki-simp-all), because of the change in output style of the former. Results are best when mixing all pseudo-parallel pairs into a single dataset (pseudo-all). Having access to a small amount of *in-domain* pseudo-parallel data, in addition to *out-of-domain* pairs, seems to be beneficial to the success of our approach.

5.3 Human Evaluation

Due to the challenges of automatic evaluation of text simplification systems (Sulem et al., 2018), we also perform a human evaluation. We asked 8 fluent English speakers to rate the grammaticality, meaning preservation, and simplicity of model outputs produced for 100 randomly selected sentences from our test set. We exclude any model outputs which leave the input unchanged. Grammaticality and meaning preservation are rated on a Likert scale from 1 (Very bad) to 5 (Very good). Simplicity of the output sentences, in comparison to the input, is rated following (Štajner et al., 2018), between: -2 (much more difficult), -1 (somewhat more difficult), 0 (equally difficult), 1 (somewhat simpler) and 2 (much simpler).

The results are reported in Table 7, where we compare our parallel baseline (baseline-272K in Table 6) to our best model trained on a mixture of parallel and pseudo-parallel data (wiki-simp-65) and our best model trained on pseudo-parallel data only (pseudo-all). We also evaluate the original Simple Wikipedia references (reference) for comparison. In terms of simplicity, our pseudo-parallel systems are closer to the result of reference than is baseline-272K, indicating that they better match the target sentence style. baseline-272K and wiki-simp-65 perform similarly to the references in terms of grammaticality, with baseline-272K having a small edge. In terms of meaning preservation, both do worse than the references, with wiki-simp-65 having a small edge. pseudo-all performs worse on both grammaticality and meaning preservation, but is on par with the simplicity result of wiki-simp-65.

In Table 8, we also show example outputs of our best models (additional examples are available in the supplementary material). The models trained on parallel plus additional pseudo-parallel data produced outputs that preserve the meaning of 'Jeddah' as a city better than our parallel baseline, while correctly simplifying *principal* to *main*. The model trained solely on pseudo-parallel data produces a similar output, apart from wrongly replacing *jeddah* with *islam*.

Table 7: Human evaluation of the Grammaticality (**G**), Meaning preservation (**M**) and Simplicity (**S**) of model outputs (on the first Beam hypothesis).

Method	G	M	S
reference	4.53	4.34	0.69
baseline-272K	4.51	3.68	0.9
+ wiki-simp-65	4.39	3.76	0.74
pseudo-all	4.02	2.96	0.77

Table 8: Example model outputs (first Beam hypothesis).

Method	Example
input	jeddah is the **principal** gateway to mecca , islam ' s holiest city , which able-bodied muslims are required to visit at least once in their lifetime .
reference	jeddah is the **main** gateway to mecca , the holiest city of islam , where able-bodied muslims must go to at least once in a lifetime .
baseline-282K	it is the highest gateway to mecca , islam .
+ wiki-sim-65	jeddah is the **main** gateway to mecca , islam 's holiest city .
+ wiki-news-74	it is the **main** gateway to mecca , islam ' s holiest city .
pseudo-all	islam is the **main** gateway to mecca , islam 's holiest city .

6 Conclusion

We developed a hierarchical method for extracting pseudo-parallel sentence pairs from two monolingual comparable corpora composed of different text styles. We evaluated the performance of our method on automatic alignment benchmarks and extrinsically on automatic text simplification. We find improvements arising from adding pseudo-parallel sentence pairs to existing parallel datasets, as well as promising results when using the pseudo-parallel data on its own.

Our results demonstrate that careful engineering of pseudo-parallel datasets can be a successful approach for improving existing monolingual text-to-text rewriting tasks, as well as for tackling novel tasks. The pseudo-parallel data could also be a useful resource for dataset inspection and analysis. Future work could focus on improvements of our system, such as refined approaches to sentence pairing.

Acknowledgments

We acknowledge support from the Swiss National Science Foundation (grant 31003A_156976).

References

Emil Abrahamsson, Timothy Forni, Maria Skeppstedt, and Maria Kvist. 2014. Medical text simplification using synonym replacement: Adapting assessment of word difficulty to a compounding language. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 57–65.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR 2015*.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 25–32. Association for Computational Linguistics.

Peter F Brown, Jennifer C Lai, and Robert L Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 169–176. Association for Computational Linguistics.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *Proc. of EMNLP 2017*.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 665–669. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proc. of NAACL 2019*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2017. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*.

Francis Grégoire and Philippe Langlais. 2018. Extracting parallel sentences with bidirectional recurrent neural networks to improve machine translation. *arXiv preprint arXiv:1806.05559*.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *HLT-NAACL*, pages 211–217.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: a project note. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 9–16. Association for Computational Linguistics.

Tomoyuki Kajiwara and Mamoru Komachi. 2016. Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1147–1158.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Matt J Kusner, Yu Sun, Nicholas I Kolkin, Kilian Q Weinberger, et al. 2015. From word embeddings to document distances. In *ICML*, volume 15, pages 957–966.

Gus Liu, Pol Rosello, and Ellen Sebastian. 2016. Style transfer with non-parallel corpora.

Benjamin Marie and Atsushi Fujita. 2017. Efficient extraction of pseudo-parallel sentences from raw monolingual data using word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 392–398.

Robert C Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Conference of the Association for Machine Translation in the Americas*, pages 135–144. Springer.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 85–91.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Language Engineering Conference, 2002. Proceedings*, pages 64–71. IEEE.

Sanja Štajner and Sergiu Nisioi. 2018. A detailed evaluation of neural sequence-to-sequence models for in-domain and cross-domain text simplification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resource Association.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Bleu is not suitable for the evaluation of text simplification. In *EMNLP*.

Jakob Uszkoreit, Jay M Ponte, Ashok C Popat, and Moshe Dubiner. 2010. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1101–1109. Association for Computational Linguistics.

Sanja Štajner, Marc Franco-Salvador, Paolo Rosso, and Simone Paolo Ponzetto. 2018. CATS: A Tool for Customized Alignment of Text Simplification Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Dependency-Based Relative Positional Encoding for Transformer NMT

Yutaro Omote and **Akihiro Tamura** and **Takashi Ninomiya**
Ehime University
{omote@ai.cs, tamura@cs, ninomiya@cs}.ehime-u.ac.jp

Abstract

In this paper, we propose a novel model for Transformer neural machine translation that incorporates syntactic distances between two source words into the relative position representations of a self-attention mechanism. In particular, the proposed model encodes pair-wise relative depths on a source dependency tree, which are the differences between the depths of two source words, in the encoder's self-attention. Experiments show that our proposed model achieved a 0.5 point gain in BLEU on the Asian Scientific Paper Excerpt Corpus Japanese-to-English translation task.

1 Introduction

Machine translation (MT) has been actively studied for many decades. In recent years, neural machine translation (NMT) has become dominant. In particular, the Transformer model (Vaswani et al., 2017), which is based solely on attention mechanisms, has advanced the state-of-the-art performance on various translation tasks and has become the focus of many MT researchers nowadays. Unlike recurrent neural network (RNN) based models (Sutskever et al., 2014; Luong et al., 2015) or convolutional neural network (CNN) based models (Gehring et al., 2017), the Transformer model attends to words in the same sentence, i.e., a source sentence or a target sentence, through the self-attention mechanisms in each encoder and decoder. In addition, it encodes the positional information of each word, such as the word order, as positional encoding (PE) so that recurrent and convolutional structures are excluded and training can be parallelized. Since NMT appeared, translation performance has been improved by using the syntactic information, such as phrase structures or dependency structures, of the source-side, target-side, or both (Ding and Palmer, 2005; Chen et al., 2017; Eriguchi et al., 2017; Wu et al., 2018). In semantic role labeling (SRL), though the task is not MT, Strubell et al. (2018) improved a Transformer-based model through the learning of self-attention weighting on the basis of syntactic information, i.e., dependency structures. Hence, it is expected that the performance of Transformer NMT will be improved by incorporating syntax information.

In this paper, we aim to improve Transformer NMT by using dependency structures. Some researchers have improved Transformer NMT by modifying self-attention. Shaw et al. (2018) used relative position information between two words encoded in self-attention in addition to the absolute position information of words.

Inspired by Shaw et al. (2018), we propose a novel Transformer NMT model that incorporates the relationships between two words on source dependency structures into relative position representations in self-attention. In particular, the proposed model adds a vector that encodes relative positional relationships between words on source dependency structures to a word embedding vector. It adds only dependency information to the word embedding vector; hence, there is no need to change the whole Transformer's mechanism or objective function, and it is easy to adapt the mechanism to other extended Transformer models because it is highly extensible. Strubell et al. (2018)'s method is different from our work in that their task is SRL, and to learn the attention between words directly from dependency structures, they largely changed the Transformer's model and objective function.

We evaluate the effectiveness of the proposed model on the WAT'18 Asian Scientific Paper Excerpt Corpus (ASPEC) Japanese-to-English translation task. The experimental results demonstrate that our approach achieves a 0.5 point gain in BLEU over baseline Transformers (Vaswani et al., 2017; Shaw et al., 2018).

2 Related Work

NMT performance has been improved by using the syntactic information of source language sentences, target language sentences, or both.

Some researchers have focused on phrase structures as syntactic information. Aharoni and Goldberg (2017) incorporated target-side phrase structures into NMT, and Eriguchi et al. (2016) and Ma et al. (2018) incorporated source-side phrase structures. Our work is different from their research in that we focus on dependency structures rather than phrase structures. In addition, while their models are based on RNN-based NMT models, we aim to improve a Transformer NMT model.

Other researchers have focused on dependency structures as syntactic information. Chen et al. (2017) proposed a hybrid NMT model of RNNs and CNNs to incorporate syntactic information into an encoder. Their model first learns source dependency representations to compute dependency context vectors by using CNNs. The RNN-based encoder-decoder model learns a translation model, which is provided with the CNNs' syntactic information. Sennrich and Haddow (2016) proposed an RNN-based NMT model that combines embedding vectors of linguistic features such as part-of-speech tags and dependency relation labels on a source sentence with the embedded representations of the source words. Eriguchi et al. (2017) proposed a hybrid model, called NMT+RNNG, that learns parsing and translation by combining recurrent neural network grammar into an RNN-based NMT.

Most existing dependency-based NMT models, including the above-mentioned models, are improvements over RNN-based NMT models, which, in terms of structure, differ greatly from the Transformer model. Because we make the proposed model consider dependency information in self-attention, which is the Transformer's characteristic structure, the usage of dependency information is different from their models.

Recently, Wu et al. (2018) and Ma et al. (2019) incorporated syntactic information into Transformer NMT. Wu et al. (2018) proposed a dependency-based NMT model that uses dependency trees for both source and target languages. Their model encodes source sentences with two extra sequences linearized from source dependency trees and jointly generates both target sentences and their dependency trees. They applied their model not only to bi-directional RNNs but also to the Transformer, but did not improve the Transformer's architecture. In contrast, we improve the Transformer model so that it incorporates source dependency information by encoding pair-wise relative depths on a source dependency tree, which are the differences between the depths of two source words, in the encoder's self-attention.

Ma et al. (2019) proposed several strategies for improving NMT with neural syntax distance (NSD), which has been used for constituent parsing (Shen et al., 2018), and dependency-based NSD, which is an extension of the original NSD for dependency trees. In their work, they proposed a syntactic PE for Transformer NMT in order to incorporate positions on a dependency tree for each word via an absolute PE mechanism. In contrast, our model uses relative dependency-based distances between two words via a relative PE mechanism in the encoder's self-attention.

3 Background

In this section, we first describe the baseline of our proposed model, the Transformer model. Then, we describe a Transformer model that employs relative PE.

3.1 Transformer

Transformer (Vaswani et al., 2017) is an encoder-decoder model that has a distinct architecture based on self-attention. Figure 1 shows the architecture of the model. Unlike RNN-based NMT and CNN-based NMT, Transformer does not have a recurrent or convolutional configuration of networks. Instead, it encodes source sentences as intermediate representations by using self-attention and decodes them by using self-attention and encoder-decoder attention.

The encoder maps an input sequence $(\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})$ to a sequence of vector representations $Z = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_n})$. Given Z, the decoder generates an output sequence $(\boldsymbol{y_1}, \ldots, \boldsymbol{y_{n'}})$. In both the encoder and decoder, the embedding layer converts input tokens (source tokens in the

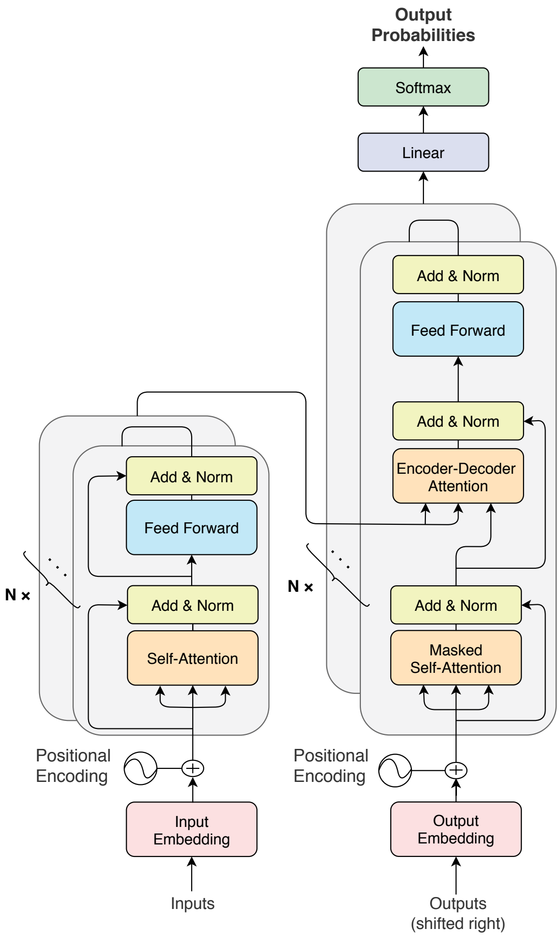


Figure 1: Architecture of Transformer

encoder and target tokens in the decoder) to vectors of dimension d_{model}. Because the information on proximity between tokens is not considered in self-attention itself, the information on a token's position is embedded by using positional encoding (PE). Specifically, PE provides a matrix that represents the absolute position information of tokens in a sentence, and Transformer adds PE to the embedding matrix of the input tokens. Each element of PE is computed by the following equations, which are sine and cosine functions of different frequencies.

$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{model}}),$$
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}}),$$

where pos is the position of each input token, i is the dimension of each element, and d_{model} is the embedding dimension of an input token. The input of the encoder's or decoder's first layer is the embedding matrix added with the positional encoding.

The encoder's layer has two sub-layers. The first sub-layer is a multi-head self-attention mechanism, and the second layer is a simple, position-wise fully connected feed-forward network (FFN). The decoder's layer has three sub-layers. The first sub-layer is a masked multi-head self-attention mechanism, the second sub-layer is a multi-head encoder-decoder attention mechanism, and the third sub-layer is the FFN.

Residual connection (He et al., 2016) is applied to the sub-layers, followed by layer normalization (Ba et al., 2016), i.e., the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the output of the original sub-layer.

The self-attention and the encoder-decoder attention employ a multi-head attention mechanism. The multi-head attention first computes h dot-product attentions after linearly mapping three input vectors, $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$[*1] $\in \mathbb{R}^{1\times d_{model}}$, from d_{model} dimension to d_k dimension with parameter matrices, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model}\times d_k}(i = 1, \ldots, h)$, where d_{model} is the dimension of input vectors, and $d_k = d_{model}/h$. In what follows, each dot-product attention is referred to as a head (H_i $(i = 1, \ldots, h)$).

$$H_i = Attention(\boldsymbol{q'}, \boldsymbol{k'}, \boldsymbol{v'}), \quad (1)$$

$$Attention(\boldsymbol{q'}, \boldsymbol{k'}, \boldsymbol{v'}) = softmax(\frac{\boldsymbol{q'}\boldsymbol{k'}^T}{\sqrt{d_k}})\boldsymbol{v'}, \quad (2)$$

$$\boldsymbol{q'} = \boldsymbol{q}W_i^Q, \boldsymbol{k'} = \boldsymbol{k}W_i^K, \boldsymbol{v'} = \boldsymbol{v}W_i^V. \quad (3)$$

Then, multi-head attention linearly maps concatenated heads with a parameter matrix, $W^o \in \mathbb{R}^{d_{model}\times d_{model}}$.

$$MultiHead(\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}) = Concat(H_1, \ldots, H_h)W^O. \quad (4)$$

The encoder's self-attention computes Equation 4 by substituting the intermediate states of the encoder, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, for $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$. Specifically, each head computes the following weighted sum.

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij}\boldsymbol{x}_j W^V, \quad (5)$$

where $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ are the outputs of the self-attention. Each coefficient, α_{ij}, is computed by using a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n}\exp(e_{ik})}, \quad (6)$$

[*1] In this paper, we treat a vector as a row vector according to the original paper (Vaswani et al., 2017) unless otherwise noted.

where e_{ij} is computed:

$$e_{ij} = \frac{(\boldsymbol{x}_i W^Q)(\boldsymbol{x}_j W^K)^T}{\sqrt{d_z}}, \tag{7}$$

where d_z is the dimension of $\boldsymbol{z}_i$.

The decoder's self-attention computes Equation 4 by substituting the intermediate states of the decoder for $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$. During inference, however, it is not possible for the decoder to get the information on the words that will be generated later when predicting a word, i.e., only the intermediate states of the sub-sequence that has been generated can be used for self-attention. Hence, masked self-attention is introduced to the decoder's self-attention so as not to calculate the self-attention between a predicted word and succeeding words. Masked self-attention is calculated by changing Equation 7:

$$e_{ij} = \begin{cases} \frac{(\boldsymbol{x}_i W^Q)(\boldsymbol{x}_j W^K)^T}{\sqrt{d_z}} & (i \geq j), \\ -\infty & (otherwise). \end{cases} \tag{8}$$

The coefficient representing the strength of the relationship between a certain word and the word located behind it ($i < j$) becomes zero, and it can be controlled so as not to consider the relationship. Hence, Equation 6 is changed:

$$\alpha_{ij} = \begin{cases} \frac{\exp{(e_{ij})}}{\sum_{k=1}^{n} \exp{(e_{ik})}} & (i \geq j), \\ 0 & (otherwise). \end{cases} \tag{9}$$

In the encoder-decoder attention, the intermediate states of the decoder are used for $\boldsymbol{q}$, and the outputs of the encoder are used for $\boldsymbol{k}, \boldsymbol{v}$.

The FFN for input $\boldsymbol{x}$ compute as follows:

$$FFN(\boldsymbol{x}) = max(0, \boldsymbol{x}W_1 + \boldsymbol{b}_1)W_2 + \boldsymbol{b}_2, \tag{10}$$

where $W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}$ are parameter matrices, and $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ are biases.

3.2 Transformer with Relative Positional Encoding

Shaw et al. (2018) proposed an extended transformer model that captures the pairwise relationships between input elements in terms of relative positions in both the encoder and decoder. In their method, the relationships between the intermediate representations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, i.e., relative position information between the i-th and j-th words in an input sentence, are represented by vectors $\boldsymbol{a}_{ij}^V, \boldsymbol{a}_{ij}^K \in \mathbb{R}^{d_k}$. The relative position representations are added to the output of the sub-layer to be the input to the next layer. Specifically, the following equation is used instead of Equation 5.

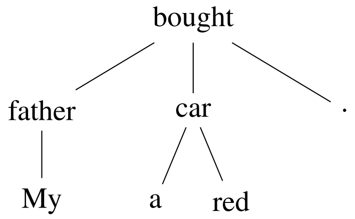


Figure 2: Example of Dependency Tree

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij}(\boldsymbol{x}_j W^V + \boldsymbol{a}_{ij}^V). \tag{11}$$

The following equation is also used for the substitution of Equation 7 in order to consider relative position relationships between words in calculating e_{ij}:

$$e_{ij} = \frac{\boldsymbol{x}_i W^Q (\boldsymbol{x}_j W^K + \boldsymbol{a}_{ij}^K)^T}{\sqrt{d_z}}. \tag{12}$$

Shaw et al. (2018) assume that relative position information is not useful when the distance is long. They define the maximum relative position as a constant k. In addition, the relative position relationships between two words are captured by $2k + 1$ unique labels as follows, considering that succeeding words are in a positive direction and preceding words are in a negative direction.

$$\boldsymbol{a}_{ij}^K = \boldsymbol{w}_{clip(j-i,k)}^K, \tag{13}$$

$$\boldsymbol{a}_{ij}^V = \boldsymbol{w}_{clip(j-i,k)}^V, \tag{14}$$

$$clip(x, k) = max(-k, min(k, x)), \tag{15}$$

where $\boldsymbol{w}^K = (\boldsymbol{w}_{-k}^K, \ldots, \boldsymbol{w}_k^K)$ and $\boldsymbol{w}^V = (\boldsymbol{w}_{-k}^V, \ldots, \boldsymbol{w}_k^V)$ ($\boldsymbol{w}_k^K, \boldsymbol{w}_k^V \in \mathbb{R}^{d_k}$) are relative position representations to be learned.

4 Dependency-Based Relative Positional Encoding for Transformer

In this section, we explain our proposed method, which encodes relative positions on source dependency trees in Transformer. We first introduce the inter-word distance on source dependency trees and then explain dependency-based relative positional encoding, which provides relative position representations on the trees. The dependency-based encoding is incorporated into

	My	father	bought	a	red	car	.
My	0	-1	-2	0	0	-1	-1
father	1	0	-1	1	1	0	0
bought	2	1	0	2	2	1	1
a	0	-1	-2	0	0	-1	-1
red	0	-1	-2	0	0	-1	-1
car	1	0	-1	1	1	0	0
.	1	0	-1	1	1	0	0

Table 1: Examples of Dependency-based Inter-Word Distances

the self-attention mechanism, following the idea of relative positional encoding (Shaw et al., 2018).

The inter-word distance on dependency trees is defined as the relative depth between two words in dependency trees. The relative depth $dist_{ij}$ between node n_i and node n_j corresponding to word w_i and word w_j is defined as follows:

$$dist_{ij} = depth(n_j) - depth(n_i), \tag{16}$$

where $depth(n)$ is the depth of node n in a dependency tree. For example, in Figure 2, the depth of "bought" (w_3) relative to "My" (w_1) is calculated by $dist_{1,3} = 0 - 2 = -2$. Table 1 shows a list of the inter-word distances on the dependency tree shown in Figure 2.

The relative position between node n_i and node n_j in a source dependency tree is represented by vectors, $\boldsymbol{b}^V_{ij}, \boldsymbol{b}^K_{ij} \in \mathbb{R}^{d_k}$, and the following equations are used instead of Equations 11 and 12.

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij}(\boldsymbol{x}_j W^V + \boldsymbol{b}^V_{ij}), \tag{17}$$

$$e_{ij} = \frac{\boldsymbol{x}_i W^Q (\boldsymbol{x}_j W^K + \boldsymbol{b}^K_{ij})^T}{\sqrt{d_z}}. \tag{18}$$

We assume that the influence of a distance decreases if the distance is longer than some certain threshold. We limit the maximum distance to a constant l. The relative position representations $\boldsymbol{b}^V_{ij}$ and $\boldsymbol{b}^K_{ij}$ between node n_i and node n_j in a dependency tree are defined with inter-word distance labels:

$$\boldsymbol{b}^K_{ij} = \boldsymbol{w}^K_{clip(dist_{ij},l)}, \tag{19}$$

$$\boldsymbol{b}^V_{ij} = \boldsymbol{w}^V_{clip(dist_{ij},l)}. \tag{20}$$

Using these expressions, the encoder's self-attention networks learn the relative position representations on a source dependency structure. We call this model $Transformer_{dep}$.

We also describe a hybrid model that learns both relative position representations on dependency structures and relative position representations for linear relations in sentences, i.e., the relative positional encoding explained in Section 3.2. This hybrid method is called $Transformer_{dep+rel}$. The $Transformer_{dep+rel}$ model uses the sum of $\boldsymbol{a}^V_{ij}$ and $\boldsymbol{b}^V_{ij}$ and the sum of $\boldsymbol{a}^K_{ij}$ and $\boldsymbol{b}^K_{ij}$ as relative position information between two words. $\boldsymbol{z}_i$ and e_{ij} are defined in $Transformer_{dep+rel}$ as follows:

$$\boldsymbol{z}_i = \sum_{j=1}^{n} \alpha_{ij}(\boldsymbol{x}_j W^V + \boldsymbol{a}^V_{ij} + \boldsymbol{b}^V_{ij}), \tag{21}$$

$$e_{ij} = \frac{\boldsymbol{x}_i W^Q (\boldsymbol{x}_j W^K + \boldsymbol{a}^K_{ij} + \boldsymbol{b}^K_{ij})^T}{\sqrt{d_z}}. \tag{22}$$

5 Experiments

5.1 Experimental Setup

We experimented on the WAT'18 Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016) by using the Japanese-to-English language pair. We tokenized English sentences by using Moses (Koehn et al., 2007) and Japanese sentences by using KyTea (Neubig et al., 2011). We also parsed the dependency of the Japanese sentences by using EDA[*2].

For model learning, we used 1,341,417 sentence pairs of 50 words or less for both the English and Japanese sentences from the first 1.5 million sentence pairs of the training data (train-1.txt, train-2.txt). The Japanese dictionary was comprised of words that appeared 7 times or more in the training data, and the English dictionary was comprised of words that appeared 10 times or more in the training data. The other words were replaced with $\langle UNK \rangle$ tags representing unknown words. We used 1,790 sentences (dev.txt) as validation data and 1,812 sentences (test.txt) as test data.

We compared our models, $Transformer_{dep}$ and $Transformer_{dep+rel}$, with two baseline Transformer NMT models, $Transformer_{abs}$ (Vaswani et al., 2017), which learns absolute position representations, and $Transformer_{rel}$ (Shaw et al., 2018), which learns relative position representations in a sentence.

Hyper-parameters of all Transformer models were determined, following the settings of Vaswani et al. (2017). We set the number of stacks

[*2] http://www.ar.media.kyoto-u.ac.jp/tool/EDA/

Model	BLEU
$Transformer_{abs}$	25.91
$Transformer_{rel}$	26.72
$Transformer_{dep}$	26.10
$Transformer_{dep+rel}$	27.22

Table 2: Experimental Results

$\boldsymbol{b}_{ij}^V$	$\boldsymbol{b}_{ij}^K$	BLEU
✓	✓	16.71
×	✓	16.60
✓	×	15.62
×	×	8.69

Table 3: Experimental Results for Ablating Relative Position Representations $\boldsymbol{b}_{ij}^V, \boldsymbol{b}_{ij}^K$

of the encoder and decoder layers to 6, the number of heads to 8, and the embedding dimension to 512. We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$. We used the same warm-up and decay strategy for the learning rate as Vaswani et al. (2017), with $4,000$ warm-up steps.

The maximum distances of the relative position for linear relations in the sentences and dependency-based relations were set as $k = 2$, $l = 2$ for $Transformer_{rel}$, $Transformer_{dep}$, and $Transformer_{dep+rel}$[*3]. The batch size was 256, the number of epochs was 30, and the model with the best accuracy for the validation data was applied to the test data. In our experiments, target sentences were generated by a greedy algorithm.

5.2 Results

The evaluation results are shown in Table 2. We used BLEU to evaluate the translation performance. As shown in the table, $Transformer_{dep}$ improved by 0.19 BLEU points against $Transformer_{abs}$. This means that the dependency-based positional encoding was effective for the Transformer model. Although the effectiveness of our dependency-based positional encoding ($Transformer_{dep}$) was not as great as the relative positional encoding ($Transformer_{rel}$), the hybrid model ($Transformer_{dep+rel}$) achieved the best result among these models. $Transformer_{dep+rel}$ improved by 1.31 BLEU points against $Transformer_{abs}$ and by 0.50 BLEU points against $Transformer_{rel}$. From these results, on the Japanese-to-English translation task, the performance of Transformer NMT can be improved by incorporating source dependency structures into relative position representations.

5.3 Discussion

Shaw et al. (2018) verified the effectiveness of both $\boldsymbol{a}_{ij}^V$ and $\boldsymbol{a}_{ij}^K$, representing relative positional relationships. They showed that the translation accuracy was comparative when $\boldsymbol{a}_{ij}^V$ was removed from their model, but the translation accuracy decreased when $\boldsymbol{a}_{ij}^K$ was removed. This means that $\boldsymbol{a}_{ij}^K$ was an effective representation, but $\boldsymbol{a}_{ij}^V$ was less effective. In this section, to confirm the effectiveness of the dependency-based representations, we conducted ablation experiments on $\boldsymbol{b}_{ij}^V$ and $\boldsymbol{b}_{ij}^K$. We evaluated the Japanese-English translation performance for $Transformer_{dep}$. The absolute positional encoding was removed from all models, following the settings of Shaw et al. (2018)'s verification. Specifically, we evaluated (i) the $Transformer_{dep}$ model that used only $\boldsymbol{b}_{ij}^V$, where Equation 18 was changed to Equation 7, (ii) the $Transformer_{dep}$ model that used only $\boldsymbol{b}_{ij}^K$, where Equation 17 was changed to Equation 5, and (iii) the $Transformer_{dep}$ model that used neither $\boldsymbol{b}_{ij}^V$ and $\boldsymbol{b}_{ij}^K$, i.e., $Transformer_{abs}$ without the absolute positional encoding.

The settings for the ablation experiment were as follows. In model training, we used the first 100,000 sentence pairs of 50 words or less for both English and Japanese sentences, which were extracted from the training data (train-1.txt). Both the Japanese dictionary and the English dictionary were comprised of words that appeared 2 times or more in the training data, and the other words were treated as unknown words with UNK tags. The batch size was 100, and the number of epochs was 50. Other settings were the same as the main experiments in Section 5.1.

The results are shown in Table 3. Table 3 shows that $Transformer_{dep}$ using only $\boldsymbol{b}_{ij}^V$ was 1.09 points lower than the baseline $Transformer_{dep}$, which used both $\boldsymbol{b}_{ij}^V$ and

[*3] We chose $k = 2$ because Shaw et al. (2018) showed that BLEU scores for $k \geq 2$ are nearly unchanged. l was tuned on development data.

$\boldsymbol{b}_{ij}^K$, while $Transformer_{dep}$ using only $\boldsymbol{b}_{ij}^K$ was 0.11 points slightly lower than the baseline $Transformer_{dep}$. Table 3 also shows that the $Transformer_{dep}$ that used neither $\boldsymbol{b}_{ij}^V$ and $\boldsymbol{b}_{ij}^K$ was 8.02 points lower than the baseline $Transformer_{dep}$, which was significantly worse.

These results were consistent with the experimental results in Shaw et al. (2018). The dependency-based relative position representations, $\boldsymbol{b}_{ij}^K$ and $\boldsymbol{b}_{ij}^V$, were shown to be effective, but $\boldsymbol{b}_{ij}^K$ was more effective than $\boldsymbol{b}_{ij}^V$.

6 Conclusion

In this paper, we proposed a novel Transformer NMT model that incorporates syntactic distances between two source words into the relative positional encoding of an encoder's self-attention mechanism. We demonstrated that our proposed model improved the translation accuracy, in terms of BLUE score, on the ASPEC Japanese-to-English translation task.

For future work, we would like to improve our model by introducing relative positional encoding to target dependency structures, i.e., dependency-based relative positional encoding for decoders. For example, we would like to integrate our encoding into the dependency-based decoder in (Wu et al., 2018).

Acknowledgement

The research results have been achieved by "Research and Development of Deep Learning Technology for Advanced Multilingual Speech Translation", the Commissioned Research of National Institute of Information and Communications Technology (NICT) , JAPAN. This work was partially supported by JSPS KAKENHI Grant Number JP18K18110.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 132–140. https://doi.org/10.18653/v1/P17-2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita, and Tiejun Zhao. 2017. Neural machine translation with source dependency representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2846–2852. https://doi.org/10.18653/v1/D17-1304.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Association for Computational Linguistics, pages 541–548. http://aclweb.org/anthology/P05-1067.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 823–833. https://doi.org/10.18653/v1/P16-1078.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 72–78. https://doi.org/10.18653/v1/P17-2012.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*. PMLR, International Convention Centre, Sydney, Australia, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 770–778. https://doi.org/10.1109/CVPR.2016.90.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and*

Poster Sessions. Association for Computational Linguistics, Prague, Czech Republic, pages 177–180. https://www.aclweb.org/anthology/P07-2045.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2019. Improving neural machine translation with neural syntactic distance. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 2032–2037. https://www.aclweb.org/anthology/N19-1205.

Chunpeng Ma, Akihiro Tamura, Masao Utiyama, Tiejun Zhao, and Eiichiro Sumita. 2018. Forest-based neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 1253–1263. https://www.aclweb.org/anthology/P18-1116.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 2204–2208.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 529–533. http://aclweb.org/anthology/P11-2093.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Association for Computational Linguistics, pages 83–91. https://doi.org/10.18653/v1/W16-2209.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, pages 464–468. https://doi.org/10.18653/v1/N18-2074.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 1171–1180. https://doi.org/10.18653/v1/P18-1108.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 5027–5038. http://aclweb.org/anthology/D18-1548.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Shuangzhi Wu, Dongdong Zhang, Zhirui Zhang, Nan Yang, Mu Li, and Ming Zhou. 2018. Dependency-to-dependency neural machine translation. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 26(11):2132–2141. https://doi.org/10.1109/TASLP.2018.2855968.

From Image to Text in Sentiment Analysis via Regression and Deep Learning

Daniela Onita
Faculty of Mathematics
and Computer Science
University of Bucharest
danielaonita25@gmail.com

Liviu P. Dinu
Faculty of Mathematics
and Computer Science
University of Bucharest
ldinu@fmi.unibuc.ro

Birlutiu Adriana
Computer Science Department
1 Decembrie 1918 University
of Alba Iulia
adriana.birlutiu@uab.ro

Abstract

Images and text represent types of content which are used together for conveying user emotions in online social networks. These contents are usually associated with a sentiment category. In this paper, we investigate an approach for mapping images to text for three types of sentiment categories: positive, neutral and negative. The mapping from images to text is performed using a Kernel Ridge Regression model. We considered two types of image features: *i)* RGB pixel-values features, and *ii)* features extracted with a deep learning approach. The experimental evaluation was performed on a Twitter data set containing both text and images and the sentiment associated with these. The experimental results show a difference in performance for different sentiment categories, in particular the mapping that we propose performs better for the positive sentiment category in comparison with the neutral and negative ones. Furthermore, the experimental results show that the more complex deep learning features perform better than the RGB pixel-value features for all sentiment categories and for larger training sets.

1 Introduction

A quick look at an image is sufficient for a human to say a few words related to that image. However, this very easy task for humans is a very difficult task for the existing computer vision systems. The majority of previous work in computer vision has focused on labeling images with a fixed set of visual categories. However, even though closed vocabularies of visual concepts are a convenient modeling assumption, they are quite restrictive when compared to the vast amount of rich descriptions and impressions that a human can compose.

Some approaches that address the challenge of generating image descriptions have been proposed (Kulkarni et al., 2013; Karpathy and Fei-Fei, 2015). However, these models only rely on objective image descriptors, and do not take into account the subjectivity which appears when describing an image on social networks.

In this work, we want to take a step forward towards the goal of generating subjective descriptions of images that are close to the natural language that is used in social networks. Figure 1 gives a hint to the motivation of our work by showing several samples which were used in the experimental evaluation. Each sample consists of an image and the subjective text associated to it, and has has a sentiment associated to it: negative, neutral or positive.

The goal of our work is to generate subjective descriptions of images. The main challenge towards this goal is in the design of a model that is rich enough to simultaneously reason about contents of images and their representations in natural language domain. Additionally, the model should be free of assumptions about specific templates or categories and instead rely on learning from the training data. The model will go beyond the simple description of an image and give also a subjective impression that the image could make upon a certain person. An example of this is shown in the image from bottom-right of Figure 1, in which we do not have a captioning or description of the animal in the image but the subjective impression that the image makes upon the looker.

Our core insight is that we can map images to subjective natural text by leveraging the image-sentence data set in a supervised learning approach in which the image represents the input and the

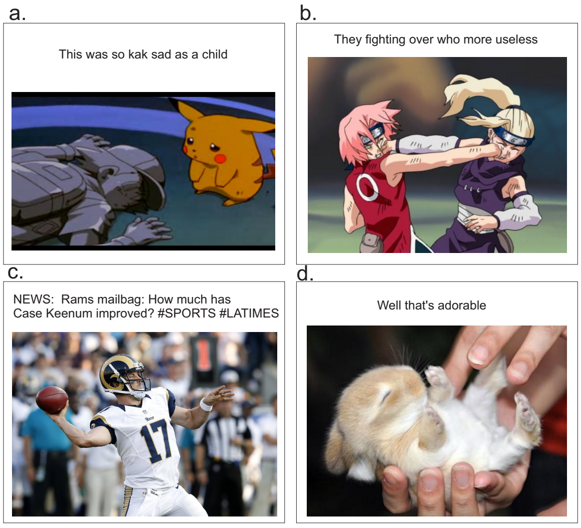

a.
This was so kak sad as a child
b.
They fighting over who more useless
c.
NEWS: Rams mailbag: How much has Case Keenum improved? #SPORTS #LATIMES
d.
Well that's adorable

Figure 1: Motivation Figure: Our model treats language as a rich label space and generates subjective descriptions of images. Examples of samples used in the experimental evaluation. Each sample consists of a pair made of an image and the subjective text associated to it. Each sample has a sentiment associated to it: a., b. samples convey a negative sentiment; c. sample conveys a neutral sentiment; d. sample conveys a positive sentiment.

sentence represents the output. We employ a Kernel Ridge Regression for the task of mapping images to text. We considered two types of image features: *i)* RGB pixel-values features, and *ii)* features extracted with a deep learning approach. We used a bag-of-words model to construct the text features. In addition, we consider several sentiment categories associated to each image-text sample, and analyze this mapping in the context of these sentiment categories.

We investigate data from Twitter. These data contain images and text associated to each image. The text is a subjective description or impression of the image, written by a user. Data from social networks, and especially Twitter, is usually associated to a sentiment, which could be a positive, neutral or negative sentiment. We designed a system that automatically associates an image to a set of words from a dictionary, these words being not only descriptors of the content of the image, but also subjective impressions and opinions of the image.

One of the interesting findings of our work is that there is a difference in performance for different sentiment categories, in particular the mapping performs better for the positive sentiment category in comparison with the neutral and negative categories. Furthermore, the experimental results show that the more complex deep learning features perform better than the RGB pixel-value features for all sentiment categories and for larger training sets.

The paper is organized as follows. Section 2 discusses related works. Section 3 describes a Kernel Ridge Regression model for image to text mapping. Section 4 shows the experimental evaluation performed on a real-world data set. Section 5 finishes with conclusions and directions for future research.

2 Related Work

Image captioning. The research presented in this paper is in the direction of image captioning, but goes further to map images to text. The texts that we consider are not only descriptions of the images, which is the task of image captioning, but they contain subjective statements related to the images. Mapping images to text is an extension of the image captioning task, and this mapping allows us to build some dictionaries of words and select from these dictionaries the words which are the most relevant to an image. The learning setting that we investigate in this paper is different to the image captioning setting, because our system automatically associates an image to a set of words from a dictionary, these words being not only descriptors of the content of the image, but also subjective opinions of the image. Image captioning has been actively studied in last years, a recent survey on image captioning is given in (Bai and An, 2018). Several approaches for image captioning are making use of the deep learning techniques (Bai and An, 2018; P. Singam, 2018).

Image description. Several approaches that address the challenge of generating image descriptions have been proposed (Kulkarni et al., 2013; Karpathy and Fei-Fei, 2015; Park et al., 2017; Ling and Fidler, 2017). However, these models only rely on objective image descriptors, and do not take into account the subjectivity which appears when describing an image on social networks.

Sentiment analysis. We investigate mapping images to text in the context of sentiment analysis. Most of the previous research in sentiment analysis is performed on text data. Recent works focus on sentiment analysis in images and videos (Yu et al., 2016; You et al., 2015; Wang et al., 2016). The research on visual sentiment analysis proceeds along two dimensions: *i)* based on hand-crafted features and *ii)* based on features generated automatically. Deep Learning techniques are capable of automatically learning robust features from a large number of images (Jindal and Singh, 2015). An interesting direction for sentiment analysis is related to word representations and capsule networks for NLP applications (Xing et al., 2019; Zhao et al., 2019).

3 Kernel Ridge Regression for Mapping Images to Text

Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ be the set of inputs and outputs, respectively, and n represents the number of observations. And let $F_X \in \mathbb{R}^{d_X \times n}$ and $F_Y \in \mathbb{R}^{d_Y \times n}$ denote the input and output feature matrices, where d_X, d_Y represent the dimensions of the input and output features respectively. The inputs represent the images, and the input features can be either simple RGB pixel-values or something more complex, such as features extracted automatically using convolutional neural networks (O'Shea and Nash, 2015). The outputs represent the texts associated to the images and the output features can be extracted using Word2Vec (Ma and Zhang, 2015).

A mapping between the inputs and the outputs can be formulated as a multi-linear regression problem (Cortes et al., 2005, 2007). Combined with Tikhonov regularization, this is also known as Kernel Ridge Regression (KRR). The KRR method is a regularized least squares method that is used for classification and regression tasks. It has the following objective function:

$$\arg_W \min\left(\frac{1}{2}||WF_X - F_Y^T||_{\mathcal{F}}^2 + \alpha\frac{1}{2}||W||_{\mathcal{F}}^2\right) \quad (1)$$

where $||\cdot||_{\mathcal{F}}$ is the Frobenius norm, α is a regularization term and the superscript ^{T} signifies the transpose of the matrix.

The solution of the optimization problem from Equation 1 involves the Moore-Penrose pseudo-inverse and has the following closed-form expression:

$$W = F_Y F_X^T (F_X F_X^T + \alpha I_{d_X})^{-1} \in \mathbb{R}^{d_Y \times d_X} \quad (2)$$

which for low-dimensional feature spaces $(d_X, d_Y \leq n)$ can be calculated explicitly (the I_{d_X} in Equation 2 represents the identity matrix of dimension d_X).

For high-dimensional data, an explicit computation of W as presented in Equation 2 without prior dimensionality reduction is computationally expensive. Fortunately, Equation 2 can be rewritten as:

$$\begin{aligned} W = & \; F_Y F_X^{\mathrm{T}} (F_X F_X^{\mathrm{T}} + \alpha \mathrm{I}_{d_x})^{-1} \\ = & \; F_Y (F_X^{\mathrm{T}} F_X + \alpha \mathrm{I}_n)^{-1} F_X^{\mathrm{T}} \end{aligned} \quad (3)$$

Making use of the kernel trick, the inputs x_i are implicitly mapped to a high-dimensional Reproducing Kernel Hilbert space (Berlinet and Thomas-Agnan, 2011):

$$\Phi = [\phi(x_1), \; \ldots, \; \phi(x_n)]. \quad (4)$$

When predicting a target y_{new} from a new observation x_{new}, explicit access to Φ is never actually needed:

$$\begin{aligned} y_{\text{new}} = & \; F_Y (\Phi^{\mathrm{T}}\Phi + \alpha \mathrm{I}_n)^{-1} \Phi^{\mathrm{T}} \phi(x_{\text{new}}) \\ = & \; F_Y (K + \alpha \mathrm{I}_n)^{-1} \kappa(x_{\text{new}}) \end{aligned} \quad (5)$$

With $K_{ij} = \phi(x_i)^{\mathrm{T}} \phi(x_j)$ and $\kappa(x_{\text{new}})_i = \phi(x_i)^{\mathrm{T}} \phi(x_{\text{new}})$, the prediction can be described entirely in terms of inner products in the higher-dimensional space. Not only does this approach work on the original data sets without the need of dimensionality reduction, but it also opens up ways to introduce non-linear mappings into the regression by considering different types of kernels, such as a Gaussian or a polynomial kernel.

4 Experimental Evaluation

4.1 Dataset

We used a data set with images and text that was introduced in (Vadicamo et al., 2017). The data have been collected from Twitter posts over a period of 6 months, and using an LSTM-SVM architecture, the tweets have been divided into three sentiment categories: positive, neutral, and negative. For image labelling the authors have selected data with the most confident textual sentiment predictions and they used these predictions to automatically assign sentiment labels to the corresponding images. In our experimental evaluation we selected 10000 images and the corresponding 10000 tweets from each of the three sentiment categories. Figure 1 shows examples of image and text data used in the experimental evaluation.

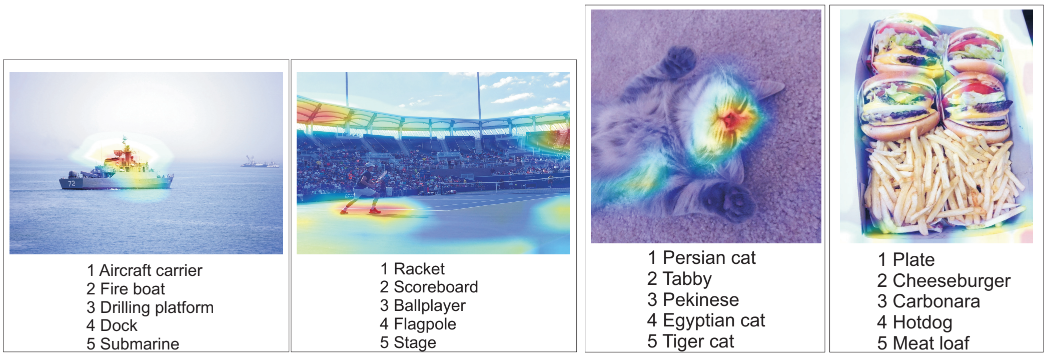


Figure 2: Visualizing heatmaps of class activation in an image.

4.2 Image and Text Features

4.2.1 Image Features

The research on feature extraction from images proceeds along two directions: i) traditional, handcrafted features, and ii) automatically generated features. With the increasing number of images and videos on the web, traditional methods have a hard time handling the scalability and generalization problem. In contrast, automated generated feature-based techniques are capable to automatically learn robust features from a large number of images (Jindal and Singh, 2015). We discuss below how these two directions for extracting features from images apply in our case, in particular, we use RGB pixel-values features for the first direction and Deep Learning based features for the second direction.

RGB pixel-values. In this approach for extracting features from images, we simply convert the images into arrays. Each image was sliced to get the RGB data. The 3-channels RGB image format was preferred instead of using 1-channel image format since we wanted to use all the available information related to an image. Using this approach, each image was described by a 2352 (28 x 28 x 3)-dimensional feature vector.

Deep Learning based features. Deep Learning models use a cascade of layers to discover feature representations from data. Each layer of a convolutional network produces an activation for the given input. Earlier layers capture low-level features of the image like blobs, edges, and colors. This primitive features are abstracted by the high-level layers. Studies from the literature suggest that while using pre-trained networks for feature extraction, the features should be extracted from the layer right before the classification layer (Rajaraman et al., 2018). For this reason, we extracted the features from the last layer before the final classification, so the entire convolutional base was used for this. The features were extracted using the pre-trained convolutional base VGG16 network (Simonyan and Zisserman, 2014). For computational reasons, the images were resampled to a 3232 pixel resolution. The model was initialized by the ImageNet weights. For understanding what part of an image was used to extract the features, visualizing heatmaps of class activation technique was employed. This is a technique which illustrates how intensely the input image activates different channels, how important each channel is with regard to the class and how intensely the input image activates the class. Figure 2 illustrates the heatmaps of class activation for some random images using VGG16 as a pre-trained convolutional base. The VGG16 model makes the final classification decision based on the highlighted parts from each image, and furthermore each image is associated with the five most representative captions.

4.2.2 Text Features

We used a Bag-of-Words (BoW) model (Harris, 1954) for extracting the features from the text samples. The first step in building the BoW model consists of pre-processing the text: removing nonletter characters, removing the html tag from the Twitter posts, converting words to lower cases, removing stop-words and making the split. A vocabulary is built from the words that appear in the text samples. The input of the BoW model is a list of strings and the output is a sparse matrix with the dimension: number of samples x number of words in the vocabulary, having 1 if a given word from the vocabulary is contained in that particular text sample. We initialized the BoW model with a maximum of 5000 features. We extracted a vocabulary for each sentiment category, and the corresponding 0-1 feature vector for each text sample.

4.3 Experimental Results

Evaluation Measure

Each output of our model represents a very large vector of probabilities, with the dimension equal to the number of words in the dictionary (approximately 5000 components). Each component of the output vector represents the probability of the corresponding word from the vocabulary as being a descriptor of that image. Given this particular form of the output, the evaluation measure was computed using the following algorithm:

1. we sorted in descending order the absolute values of the predicted output vector;
2. we created a new vector containing the first 50 words from the predicted output vector;
3. we computed the Euclidean distance between the predicted output vector values and the actual output vector.

The actual output vector is a sparse vector, a component in this vector is 1 if the corresponding word from the vocabulary is contained in that particular description of the image.

The values computed in step 3) described above were averaged over the entire test data set and the average value obtained was considered as the error.

Experimental Protocol

We designed an experimental protocol, that would help us answer the following questions:

1. Could our proposed Kernel Ridge Regression model map images to natural language descriptors?
2. What is the difference between the two types of image features that we considered? In particular, we are interested whether the more complex deep learning features give a better performance in comparison to the simple RGB pixel-values features.
3. Is there a difference in performance based on the sentiment associated to each image-text sample?

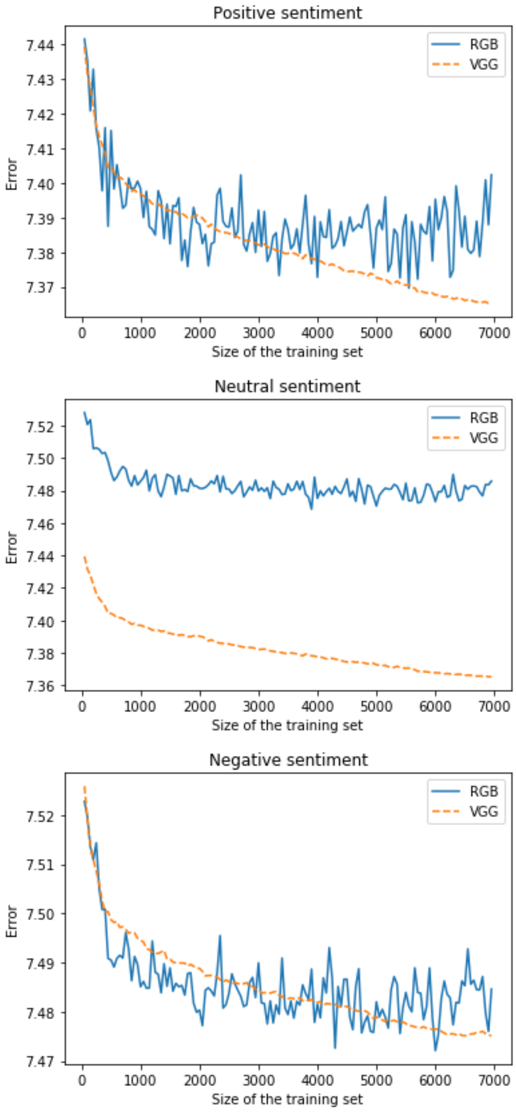


Figure 3: The plots show mean errors and standard deviation for different sizes of the training set. Comparison between RGB pixel-values features and the more complex VGG16 features. The different rows correspond to different sentiment categories: top row - positive sentiment category, middle row - neutral sentiment category, bottom row - negative sentiment category.

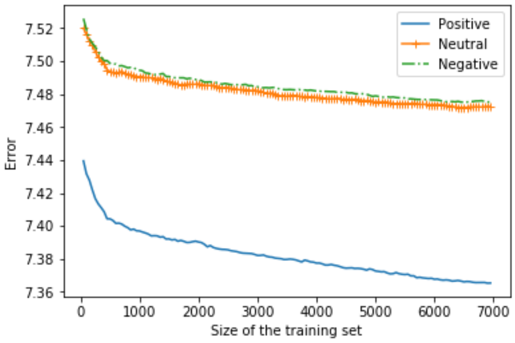


Figure 4: Comparison of the learning performance based on the type of sentiment using the VGG16 image features.

We designed the following experimental protocol. For each of the three sentiment categories, we randomly split the data 5 times into training and testing, taking 70% for training and the rest for testing. For training the model, we considered different sizes of the training set: from 50 to 7000 observations with a step size of 50. For a correct evaluation, the models built on these different training sets, were evaluated on the same test set. The error was averaged over the 5 random splits of the data into training and testing.

Results

The first two questions raised above can be answered by analyzing the experimental results shown in Figure 3. The plots show the learning curve (mean errors and standard deviations) for different sizes of the training set and for different sentiment categories. Since the error decreases as the training size increases, we can say that there is a learning involved, thus our proposed model can map images to natural language descriptors.

The plots from Figure 3 also show the comparison between the RGB pixel-values and VGG16 features for the three categories of sentiments considered. Overall, the more complex deep learning features give a better performance in comparison to the simple RGB pixel-values features.

To answer the third question, we analyzed the experimental results shown in Figure 4. There is a significant difference in learning performance for the positive sentiment category in comparison with the other two categories, both using RGB pixel-values features and VGG16 features. The positive category is simpler to be learned because of the subjective part from images: a positive feeling can be interpreted as positive for the majority of the people, but a neutral or a negative sentiment can be interpreted as having a different meaning depending on the people.

Furthermore, analyzing again Figure 3, we see that the neutral sentiment category has a different behaviour in comparison with the positive and negative sentiment categories, with respect to the image features used. In the case of neutral sentiment, the more complex VGG16 features appear to have a better performance than the simpler RGB pixel-values features as the size of the data increases. For positive and negative sentiment categories the simpler RGB pixel-values features lead to an error which varies a lot, while using the VGG16 features, the error is more stable.

5 Conclusions and Future Work

In this work, we investigated a method for image to text mapping in the context of sentiment analysis. The mapping from images to text was performed using a Kernel Ridge Regression model. We considered two types of image features: *i)* the simple RGB pixel-values features, and *ii)* a more complex set of image features extracted with a deep learning approach. Furthermore, in this paper we took a step forward form the image captioning task, which allows us to build some dictionaries of words and select from these dictionaries the words which are the most relevant to an image. We performed the experimental evaluation on a Twitter data set containing both text and images and the sentiment associated with these. We found that there is a difference in performance for different sentiment categories, in particular the mapping performs better for the positive sentiment category in comparison with the neutral and negative ones for both features extraction techniques.

We plan to further extend our approach by investigating the input-output kernel regression type of learning (Brouard et al., 2016). The output kernel would allow us to take into account the structure in the output space and benefit from the use of kernels. We also plan to integrate in our model textual captions of images obtained using a pre-trained network (Simonyan and Zisserman, 2014). The textual captions could be used as a new type of features and can be compared and integrated with the other two types of image features considered.

References

Bai, S. and An, S. (2018). A survey on automatic image caption generation. *Neurocomputing*, 311.

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science Business Media.

Brouard, C., Szafranski, M., and D'Alché-Buc, F. (2016). Input output kernel regression: supervised and semi-supervised structured output prediction with operator-valued kernels. *The Journal of Machine Learning Research*, 17(1):6105–6152.

Cortes, C., Mohri, M., and Weston, J. (2005). A general regression technique for learning transductions. In *Proceedings of ICML 2005*, pages 153–160. ACM Press.

Cortes, C., Mohri, M., and Weston, J. (2007). A general regression framework for learning stringtostring mappings. In *Predicting Structured Data*. MIT Press.

Harris, Z. (1954). Distributional structure. pages 146–62.

Jindal, S. and Singh, S. (2015). Image sentiment analysis using deep convolutional neural networks with domain specific fine tuning. In *2015 International Conference on Information Processing (ICIP)*, pages 447–451. IEEE.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2013). Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2891–2903.

Ling, H. and Fidler, S. (2017). Teaching machines to describe images via natural language feedback. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5075–5085. Curran Associates Inc.

Ma, L. and Zhang, Y. (2015). Using word2vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2895–2897. IEEE.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *ArXiv e-prints*.

P. Singam, Ashvini Bhandarkar, B. M. C. T. K. K. (2018). Automated image captioning using convnets and recurrent neural network. *International Journal for Research in Applied Science and Engineering Technology*.

Park, C., Kim, B., and Kim, G. (2017). Attend to you: Personalized image captioning with context sequence memory networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 895–903.

Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Jaeger, S., and Thoma, G. R. (2018). Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Vadicamo, L., Carrara, F., Cimino, A., Cresci, S., Dell'Orletta, F., Falchi, F., and Tesconi, M. (2017). Cross-media learning for image sentiment analysis in the wild. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Wang, J., Fu, J., Xu, Y., and Mei, T. (2016). Beyond object recognition: Visual sentiment analysis with deep coupled adjective and noun neural networks. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3484–3490. AAAI Press.

Xing, F. Z., Pallucchini, F., and Cambria, E. (2019). Cognitive-inspired domain adaptation of sentiment lexicons. *Information Processing & Management*, 56(3):554–564.

You, Q., Luo, J., Jin, H., and Yang, J. (2015). Joint visual-textual sentiment analysis with deep neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 1071–1074, New York, NY, USA. ACM.

Yu, Y., Lin, H., Meng, J., and Zhao, Z. (2016). Visual and textual sentiment analysis of a microblog using deep convolutional neural networks. *Algorithms*, 9(2):41.

Zhao, W., Peng, H., Eger, S., Cambria, E., and Yang, M. (2019). Towards scalable and reliable capsule networks for challenging nlp applications. *arXiv preprint arXiv:1906.02829*.

Building a Morphological Analyser for Laz

Esra Önal
Cognitive Science,
Boğaziçi University
Istanbul, Turkey
esra.onal@boun.edu.tr

Francis M. Tyers
Department of Linguistics
Indiana University
Bloomington, United States
ftyers@iu.edu

Abstract

This study is an attempt to contribute to documentation and revitalization efforts of endangered Laz language, a member of South Caucasian language family mainly spoken on northeastern coastline of Turkey. It constitutes the first steps to create a general computational model for word form recognition and production for Laz by building a rule-based morphological analyser using Helsinki Finite-State Toolkit (HFST). The evaluation results show that the analyser has a 64.9% coverage over a corpus collected for this study with 111,365 tokens. We have also performed an error analysis on randomly selected 100 tokens from the corpus which are not covered by the analyser, and these results show that the errors mostly result from Turkish words in the corpus and missing stems in our lexicon.

1 Introduction

The Laz language, which is mainly spoken on the northeastern coastline of Turkey and also in some parts of Georgia has been recorded as a 'definitely endangered[1]' language in UNESCO Atlas of the World's Languages in Danger. It belongs to South Caucasian language family[2] with the number of speakers estimated to be between 130,000 and 150,000 according to UNESCO 2001 records and between 250,000 and 500,000 according to more recent studies (Haznedar, 2018). Until the 1920s it was a spoken language with only some written collection of Laz grammar and folklore studies. Later, İskender Tzitaşi became the pioneer in developing a writing system for Laz based on Latin alphabet and later 'Lazuri Alboni' (Laz Alphabet) and only after 1990s, the written texts started to come out as several associations were founded for the preservation of Laz language and culture. Now with all these efforts, Laz has been thought in public schools in Turkey as an elective language course since 2013 (Kavaklı, 2015; Haznedar, 2018).

There is not much research on lexicon and syntax of Laz and the first academic level research studies began by the end of 20th century. In 1999, the first dictionary for Laz (Turkish—Laz) was prepared and published by İsmail Bucaklişi and Hasan Uzunhasanoğlu. The following years, Bucaklişi also published the first Laz grammar book (Kavaklı, 2015) and has begun teaching Laz at Boğaziçi University in İstanbul as an elective course since 2011. The foundation of the *Lazika Publishing Collective* in 2011 has given rise to the publication of more than 70 books on Laz language and literature (Kavaklı, 2015).

Laz language is only one of many that faces the danger of extinction. By the end of this century, many will not survive with the decreasing number of the native speakers of such languages (Riza, 2008). This has alarmed not only native speakers of these languages but also research community to direct their attention for language documentation as well as preservation and revitalization studies for these languages (Ćavar et al., 2016; Gerstenberger et al., 2017). Bird (2009) calls out for a 'new kind of computational linguistics' in his paper that would protect this endangered invaluable cultural heritage by helping to accelerate these studies, and he ends his paper with these words 'Who knows, we may even postpone the day when these languages utter their last words.' which emphasizes the importance of each and every attempt to keep these languages alive.

Riza (2008) gives accounts on language diversity on the Internet by pointing to the fact that many en-

[1] UNESCO defines the degree of definitely endangered as the situation in which "children no longer learn the language as mother tongue in the home".

[2] The Southwest Caucasian language family consists of four languages: Svans, Mingrelians, Georgian and Laz.

dangered languages lacks access to Information and Communication Technology and the representation of these languages on digital environment is rather low. Considering Laz resources online, there are some online dictionaries and only a couple of web sites that give information about the Laz language and culture, and many of them are mostly in Turkish or English. He suggests that regarding 'digital language divide' such small regional languages must be represented more by creating and using resources in digital format.

One of the drawbacks while working with these languages is clearly the small amount of data to begin with (written or spoken, annotated or non-annotated) (Riza, 2008). Current dominant computational methods and tools are mostly used on languages with large corpora, following a statistical approach to train their systems according to a relevant task. However, with little data at hand these methods may not present a good solution. Therefore, Gerstenberger et al. (2017) suggests a rule-based morpho-syntactic modelling for annotating small language data. On their study of Komi language, his results show by-far significant advantages of rule-based approaches for endangered languages by providing much more precise results in tagging as well as 'full- fledged grammatical description based on broad empirical evidence' and a future development for computer-assisted language learning systems.

In this study, the aim is to create a morphological analyzer using the Helsinki Finite State Toolkit (HFST) that will help to overcome manual annotation of a potential Laz corpus. Additionally, as Gerstenberger et al. (2017) suggest, these may later help developing programs to be able to facilitate learning of Laz, considering the increase of interest in Laz courses not only in secondary schools but also in universities such as Boğaziçi University and İstanbul Bilgi University (Haznedar, 2018). From spelling and grammar-checkers to machine translation systems and language learning materials, this small study will hopefully lead to further developments on Laz language in the field of Computational Linguistics.

The remainder of the paper is laid out as follows: in Section 2, the grammatical structure of Laz is discussed and later, in Section 3.1 and 3.2 the process of preparing a lexicon and corpus for Laz is described. The Section 4 gives and explains the details of the morphological analyzer and the usefulness of Flag diacritics for representing Laz verbal complex and for generating complex verbal word forms.

2 Laz

There are eight dialects of the Laz language, none of which is considered to be normative or 'standard'. Even though underlyingly the structure of these dialects is the same, they show lexical and morphological, as well as phonological differences. There are two main groups. The Western dialects (Gyulva), such as Pazar (Atina), Çamlıhemşin (Furthunaşi gamayona), Ardeşen (Arthaşeni) and the Eastern dialects (Yulva) as Fındıklı (Viʒe), Arhavi (Arkabi), Hopa (Xopa), Borçka-İçkale (Çxala).[3]

For the purposes of this initial study we have chosen to base the analyser on the Pazar dialect. The reasons for this are twofold: Firstly the Pazar dialect is less irregular in terms of verbal inflection and secondly a separate and well-documented grammar of Laz written in English is based only on this dialect.[4] Unfortunately, there is no study yet that would provide an analysis of Laz grammar to be treated as 'standard' (Haznedar, 2018).

2.1 Verbs

In terms of morphosyntactic alignment, Laz is an ergative–absolutive language. It marks the subject of unergative predicates and transitives with agentive/causer subjects with ergative case while the subject of unaccusative predicates and the direct object of transitive and ditransitive verbs are inflected with nominative case. These patterns are marked differently on the verbal complex, depending on their case markings which also indicate their argument types. The verb encodes person information both preverbally and postverbally as seen in Table 1 and Table 2 and 3.[5] While we can observe verbs agreeing with agent-like arguments and sole argu-

[3] We exclude the Sapanca dialect as the region in which it is spoken is further away from the other dialects. Speakers of the Sapanca dialect are considered to be migrated from Batum, Georgia to Sapanca, Turkey (Bucaklişi and Kojima, 2003)

[4] The main grammar book used for this study is Pazar Laz written by Öztürk and Pöchtrager (2011) which is based on courses given by İsmail Bucaklişi in Boğaziçi University and it is the most recent and only complete study on a dialect of Laz written in English which would enable us to define grammatical rules for the morphological analyser. The grammar book by René Lacroix (2009) was also referred to several times but since it is mostly based on Arhavi (Arkabi) dialect and written in French, we used it when we needed to look for more examples for certain structures and specific details, especially valency-related vowels on the verbal complex and verb classes.

[5] It should be noted that post-verbal person markers encode tense information as well.

ments in both positions at the same time, we can only observe theme-patient (of mono-transitive and ditransitive verbs) and dative marked recipient-goal or applied non-core argument agreement in preverbal position. Additionally, Laz applies a hierarchical selection rule among arguments while marking person in -2 pre-verbal position seen in Table 1. This can be represented as in (1), where D represents the dative-marked arguments in the structure and P represents theme/ patient argument type while A means agent-like argument type.[6]

(1) D1/2 > P1/2 > A1 > D3=P3=A2/3

The reason why D1/2 arguments comes first but not D3 is that D3 is unmarked; therefore, overt A/S1 markings fills the position if they are available in the structure. Only when we have A2/3 type argument, the position remains empty. We will also discuss this topic later in Section 4.1.

(2) *Bere-k Lazuri d-i-gur-am-s.*
child-ERG Laz.NOM PV-VAL-learn-TS-PRS.3.*a*.SG
'The child is learning Laz.'

(3) *Bere-s Lazuri dv-a-gur-e-n.*
child-DAT Laz.NOM PV-APPL-learn-TS-PRS.3.S.SG
'The child is able to learn Laz.'

The case markings of arguments are apt to change despite their argument type when the general construction of the predicate changes, which in turn changes the verbal complex as in present perfect constructions and in expressing ability and involuntary actions.[7] They lead to the backgrounding of agent-like arguments; therefore, we can only observe 3.SG in post-verbal person marking position unless NOM marking objects are emphasized. Emphasizing such arguments allow the verbal complex to bear their marking in post-verbal position, as in (2) and (3) from Öztürk and Pöchtrager (2011).

As seen in the example, the verb not only takes *a-* valency vowel and ability related *-e(r)* TS suffix but also changes the person marking as well. There are also valency-changing operations such as applicativization, causativization and reflexivization which introduces non-core dative marked arguments, nominative and dative-marked arguments, and verbal reflexivization through a theme argument or a non-core argument[8] respectively. All these operations commonly mark the verb with different valency-related vowel in the same preverbal position. Therefore, under conditions where the verb is needed to be inflected both applicativization and causativization at the same time, APPL vowel `i/u-` suppresses CAUS vowel `o-`. This is important for us since when we mark the verb with APPL, the structure should allow CAUS construction as well. We will discuss such intersecting constructions and how we deal with them in our lexicon file in Section 4.1 in detail. An example of this from Öztürk and Pöchtrager (2011) is given in (4).

(4) *Him Ayşe-s bere*
S/he Ayşe.DAT child-NOM
u-bgar-ap-ap-u-n
APPL-cry-CAUS-CAUS.PERF-TS-PRS.3.A.SG
'S/he has made Ayşe make the baby cry.'

Table 1 shows the pre-verbal complex of Pazar Laz and Table 2 and 3 show post-verbal complex which we have based our main FST continuation classes on.

2.2 Substantives

Adjectives, adverbs and nouns together constitute substantive category in the language since they behave similarly within a sentence depending on the suffixes they carry as well as their position. An adverb can take dative suffix *-s* and an adjective can be used as a noun by taking case or plural marker. The differentiation between these categories are not very clear.

As mentioned partially above, Laz marks nouns with case markings such as ergative *-k*, nominative (unmarked), dative *-s*, allative *-şe* (showing the direction of an event), ablative *-şe(n)*[9] (indicating the source of an event), genitive *-şi* and instrumental -te. Other than these case markings, nouns are marked plural marking *-pe* and only some nouns ending with *a* take *-lepe* as plural marker. Since there is no phonological rule for this alternation, we need to categorise each noun in our lexicon manually for our morphological analyser.

2.3 Orthography

Orthographically, we have adopted the Laz alphabet given below which is an extended version of Turk-

[6] Intransitive verbs only has S 'sole' argument which has the same marking pattern with A arguments.

[7] These three constructions are called *inversion* constructions which require certain type of predicates, specifically those including agent-like arguments, and ergative case is never available for these constructions.

[8] This additionally assumes the function of applicativization.

[9] Final *n* only occurs with post-position *doni*; therefore, we will mark every noun with both forms.

-4	-3	-2	-1	0
Affirmative preverb	Spatial preverb	Person marker	Valency-related vowel	**Root**
o-, ko-, do-, menda-	ama-, ce-, cela-, çeǩo-, çeşǩa-, do-, dolo-, e-, eǩo-, ela-, eşǩa-, eǯo-, eyo-, gama-, go-, gola-, goyo-, ǩoǩo-, ǩoşǩa-, me-, mela-, menda-, meşǩa-, meyo-, mo-, mola-, moǩo-, moşǩa-, moǯo-, moyo-, oǩo-, exo-, ǩoǯo-, oxo-, gela-, ǩoǯa-	*S-A.1: v-,p-, p̌-, b, (f-) **P-D.1: m- P-D.2: g-, k-, ǩ *'S-A' = Sole or agent-like arg **'P-D' = Patient/theme arg or Dative marked arg	i/u-, i-, a-, o-	-t̆ax- 'break'

Table 1: Pre-verbal complex the numbers in the header refers to the pre-verbal position relative to the verbal root. The spatial preverb is a prefix that indicates the direction or manner of an event. The different forms of person markers are realised based on the laryngeal properties of the following consonant. This will be later discussed in Section 4.2.

0	1	2	3	4	5	6
Root	Augment. stem formant	Causative suffix for intransitives	Causative suffix for transitives	Cusative suffix for present perfect cons.	Thematic suffix	Imperfect stem formant
-t̆ax- 'break'	-am	-in	-ap	-ap	-am,-um,-er,-ur	-t́

Table 2: Post-verbal complex-1

7	8	9	10	11
Subjunctive marker	Person suffixes	Conditional marker	Plurality	Auxiliaries
-a	S-A.1.PST: -i S-A.2.PST: -i S-A.3.PST: -u S-A.1.PRS: ∅	-ǩo	-t; -es, an (3rd)	-(e)re, -(e)rt́u

Table 3: Post-verbal complex-2

ish alphabet based on Latin letters. Across texts, they have been encoded with different characters but these forms will be the standard for our study.

3 Resources

3.1 Lexicon

The lexicon composed for this study comes from the *Büyük Lazca Sözlük* (Didi Lazuri Nenapuna). It is the most extensive dictionary available for Laz prepared by Hasan Uzunhasanoğlu , İsmail Bucaklişi and İrfan Çağatay Aleksiva in 2007 in Laz and Turkish.

The verbs were extracted from the dictionary automatically whereas other word classes were extracted semi-automatically. The words are taken as entries with their dialect labels[10] and if available, dialect-specific forms as seen in (5).[11]

(5) doinu [Atn., Viw, dorinu Gyl., Ark., Xop., doǩunapa Sap.]

We have prepared verb word lists for each dialect separately as well as a complete word list for all. Considering the possibility that dialects may borrow words from one another, we decided to build a lexicon based on not only the Pazar dialect but all dialects of Laz. This is an important strategy to form a 'common source lexicon' (Beesley and Karttunen, 2003). However, for the sake of simplicity, we have excluded nominal and verbal compounds from our lexicon.

The challenging part in preparing the lexicon has been the stemming process for verbs since the verbs in the dictionary are in their infinitival form and some of them also include preverbs. Even though the preverbs have been easily separated, the infinitive suffixes were harder to process. For example, there are verbs ending with *-alu* and while some of these verbs include *-al* suffix in their bare form, some do not. It means that they are lexically determined.

Even though noun declension was easy to define, extracting substantives from the dictionary and carefully separating them into nouns, adverbs, and adjectives as well as categorizing other syntactic elements like interjections, conjunctions and pronominals were among the hardest tasks for this study. We needed to separate these word classes semi-automatically because there were words that should be put in more than one category such as in both noun and adjective or adjective and adverb (determined only by sentential position) or noun and adverb. Therefore, it could not be possible for us to include words (except verbs) that belong to other dialects other than Pazar for this study.

3.2 Corpus

We have collected different type of written texts for our Laz corpus. However, differences in terms of dialects have forced us to divide texts into their corresponding dialects for this study since we have decided on working Pazar Laz first the reasons of which are discussed in Section 2. Unfortunately, Pazar Laz has almost no written text known in the literature. The only resource we have is an 800 page document consisting of 111,365 tokens collected by İsmail Bucaklişi, a native speaker of Pazar Laz, by himself which contains daily conversations and stories shared in his immediate circle. It should be noted that it also contains Turkish words and sentences given as translations throughout the document the effects of which on the results can be seen in Section 5.2.

4 Methodology

The purpose of this project is to develop a computational model for morphological analysis for Laz by using the Helsinki Finite-State Toolkit (HFST; (Linden et al., 2011)) which is popular in this field of research. A finite-state transducer associates a morphological analysis with the corresponding phonological representation. Xerox `lexc` and `twolc` formalism supported by HFST are used to create lexicon files and a two-level grammar file respectively (Beesley and Karttunen, 2003).

4.1 Lexicon Files

The `lexc` (**Lex**icon **C**ompiler) formalism is used to define lexicons which contain grammatical labels and morphotactic rules for the morphemes in the language (Beesley and Karttunen, 2003).

4.1.1 `lexc` File for Substantives

The substantive `lexc` file has 27 tags for morphemes indicating person, number and case information and 18 continuation classes for morphotactics or word-formation rules together with the

[10] The following dialect codes were found in the dictionary: *'Yul' (Eastern dialects), 'Gyl' (Western dialects), 'Viw' (Viʒe), 'Xop' (Xopa), 'Ark' (Arkabi), 'Çxl' (Çxala), 'Atn' (Atina/Pazar), 'Fur' (Furthunaşi gamayona), 'Arş' (Arthaşeni), 'Sap' (Sapanci).*

[11] (5) shows that *doinu* 'to give birth' belongs to *Atn.* and *Viw.* dialects, and it takes the form of *dorinu* in *Gyl., Ark.* and *Xop.* dialects, and *doǩunapa* in *Sap.* dialect.

a	*b*	*c*	*ç*	*ç̆*	*d*	*e*	*f*	*g*	*ğ*	*h*	*x*	*i*	*j*	*k*	*ǩ*	*l*
[a]	[b]	[dʒ]	[t͡ʃ^{h}]	[tʃ’]	[d]	[e]	[f]	[g]	[ɣ]	[h]	[x]	[i]	[ʒ]	[k^{h}]	[k’]	[l]
m	*n*	*o*	*p*	*p̌*	*r*	*s*	*ş*	*t*	*ť*	*u*	*v*	*y*	*z*	*ž*	*ʒ*	*ʒ̆*
[m]	[n]/[ŋ]	[p]	[p^{h}]	[p’]	[r]	[s]	[ʃ]	[t^{h}]	[t’]	[u]	[v]/[w]	[j]	[z]	[dz]	[ts^{h}]	[t͡s’]

Table 4: The 34 letters of the Laz alphabet. The letters ǩ p̌ ť ç̆ ʒ̆ ʒ and ž represent ejective consonants.

lexemes. We specified pronouns as personal pronouns, possessive pronouns, demonstratives, reflexives, interrogative pronouns, indefinite pronouns, quantifiers as well as numerals in different continuation classes. We have continuation classes for case and plural markers to show nominal inflection. There are two forms of plural markings and ablative markings each, whose differentiation is lexical, not phonological. Therefore, we have encoded this information in our lexicon by using flag diacratics that will be explained in Section 4.1.3.

4.1.2 `lexc` File for Verbs

The `lexc` file for Laz verbal complex has 53 tags for the morphemes encoding preverb, valency-related, mood, tense, person and number information and 19 continuation classes which correspond to the affixes in the verbal complex as defined in Öztürk and Pöchtrager (2011) also seen in Table 1, 2 and 3 with three additions — additive position for suffix *-ti*, — question for *-i*, — participle for *-eri*.

We have mostly followed the description in Öztürk and Pöchtrager (2011) when naming the tags and classes.

The final combined `lexc` file also includes interjection, conjunction, negation, post-position and pre-position lexicons with 5 more tags.

4.1.3 Flag Diacritics

Laz verb complex has required substantial use of *flag diacritics*[12] to solve problems like dependent person marking, and causativisation or applicativisation processes, which require preverbal valency-related vowel marking as well as postverbal causative markers at the same time. The hierarchical selection rule for person marking position preverbally among the arguments of the verb is easily applied using flag diacritics. We have allowed structures with 3rd person prefixes to only occur in structures with 3rd person suffixes by disallowing paths including 1st and 2nd person prefixes. This is done by setting a flag, @P.D-P.3@ which means **P**ositive setting of the D-P (dative–patient) type argument bearing 3rd person information and later in person suffix continuation class we reject/**D**isallow those paths with positively setting of 3rd person information by setting @D.D-P.3@ for the 1st and 2nd person suffixation. Additionally, we can also reject paths that include combinations of 1st and 2nd person D-P with 1st and 2nd person A (agent) respectively since the language does not allow such constructions. We use the same patterns as above for these rules.

We have also found flag diacritics useful for overwriting of valency-related vowels. In such constructions, we engage in two separate operations/constructions at the same time such as causativisation and present perfect construction both of which mark the verb with their specific valency-related vowel in the -1 position. However, Laz allows overwriting causative *o-* to be overwritten by applicative *i-* while keeping post verbal causative markers *-in* or *-ap*. We have managed to form these constructions by also allowing applicative *i-* (as well as causative *o-*) to have flag @P.CAUS.PRS@ which will let them through paths defined with @R.CAUS.PRS@ (**R**equire the causative feature to be present). These paths are naturally those causative suffixes which do not allow structures with related valency vowel otherwise.[13]

Other flag diacritics include N which sets the related feature as **N**egative. In our study, we use them for subjunctive suffix and its special construction with thematic suffixes. They do not normally occur together but we can see that they do in constructions with the imperfective suffix in between in (6) from Öztürk and Pöchtrager (2011).

[12] Flag diacritics are used for feature-setting and feature-unification operations. They represent long-distance constraints for dependencies within a word (Beesley and Karttunen, 2003). As a member of the same language family, Georgian also shows these kinds of long-distance dependencies in verbal complex which are effectively treated again with this device in order to build a computational grammar for Georgian (Meurer, 2009).

[13] Additionally, Laz allows only intransitive bases to take the *-in* causative suffix, so we have also tried to use flag diacritics to differentiate between transitive and intransitive bases by encoding the information onto verb itself. However, since we have automatically extracted verbs from the dictionary, we could not label all of them (2240 verb roots) as transitive or intansitive for this study, we ignore this differentiation and allow all verb roots to be able to bear both *-in* and *-ap* suffixes.

(6) *m-i-t́ax-ap-ur-t́-a-s*
D1-APPL-break-CAUS.PERF-TS-IMPRF-SUBJ-PRS.3.S.SG
'Let say that I have broken it.'

If the subjunctive follows imperfective (sets thematic suffix information as @N.THM.PRS@), the path allows subjunctive (normally disallows thematic suffixes as @D.THM.PRS@) to follow thematic suffix after imperfective; therefore, we need to get rid of @P.THM.PRS@ setting by re-setting the same feature to N as @N.THM.PRS@ which will allow the structure to go through the path by taking non-past person markers that are set as @D.TNS.PST@ disallowing past tense constructions.

The substantive `lexc` file includes only one flag diacratic case which is to label nouns that take *-lepe* plural marker instead of *-pe*. We label the noun root with @P.LEPE.PRS@ in order for it to be able to take the path with @R.LEPE.PRS@ label.

4.2 `twol` File

The `twolc` (**Two-l**evel **C**ompiler) formalism is used to define phonological and morphophonological alternations. The `twol` file mostly includes person marking elements differing based on the following consonant's laryngeal property for verbal inflection. We define rules/environments to account for morphophonological changes in the structure with archiphonemes[14] as given in Figure 1.

Laz also exhibits a phonological change in noun stems starting with *n* sound when preceded by ejective *p̌*-, the person prefix for 1.SG. The two consonants are combined and becomes *m*. We represent this as ejective *p̌* turning to *m* and dropping the initial *n* of the stem. Additionally, the final *i* sound of noun stems becomes *e* when the stem is inflected with plural marker.

```
"Assimilation of person prefix to p-"
{V}:p <=> _ >: Voiceless: ;

"Assimilation of person prefix to b-"
{V}:ṕ <=> _ >: Ejectives: ;
```

Figure 1: Two two-level phonological rules for assimilation. The underspecified prefix archiphoneme {V} is restricted to surface either as *p* before voiceless consonants or *ṕ* before ejective consonants.

We also observe a morphologically-conditioned phonological alternation for valency-related vowels. The alternation for valency-related vowels *i/u-* depends on the preverbal person information, *i-* for 1st and 2nd person, and *-u* for 3rd person.

The preverbs show a great amount of morphophonological alternations in their final vowels, such as *a, e* and *o*. When they combine with overt person prefixes (consonants) together with valency related vowels, final *o* and *a* become *e* or *o* and the change is not always predictable. They can also turn into *v* or can be dropped. Even though they may end with the same vowel, the alternations can be different when followed by the same sound; therefore; we need to define different archiphonemes for the same vowel. For example, the final *o* sound in *exo-* drops when it attaches to a verbal complex starting with *a* sound but not the one in *moyo-*.

5 Results

We have evaluated the morphological analyser by calculating the naïve coverage and doing error analysis on randomly selected 100 tokens from the corpus.

5.1 Coverage

The coverage is measured by calculating the number of the tokens that receive at least one morphological analysis by the analyser. It should also be noted that the tokens may have other analysis that is correct but not provided by the analyser even though they get at least one analysis.[15] We have collected a corpus for Pazar Laz which consists of 111,365 tokens mentioned before in Section 3.2. The final morphological analyser has 64.9% coverage over this corpus.

Corpus	Tokens	Coverage
Pazar Laz	111,365	64.9%

Table 5: Naïve coverage of the analyser

5.2 Error Analysis

We have looked at the tokens that are not covered by the morphological analyser. Randomly selected 100 tokens has been examined and separated according to their error type seen in Table 7. It should also be

[14] An archiphoneme is used as a placeholder to be later replaced with the appropriate sound determined by morphophonological rules written in `twol` file. They are given inside curly brackets.

[15] Unfortunately since there is no annotated corpus which can be used as the 'gold standard', we were unable to calculate *precision* and *recall* that could show the average accuracy of the analysis provided by the transducer.

Category	Number of Stems
Noun	9417
Verb	2240
Adjective	745
Adverb	215
Pronoun	92
Numeral	46
Interjection	31
Postposition	29
Conjunction	8
Preposition	3
Negation	4
Total	12,830

Table 6: Number of lexicon entries by part of speech / lexical category.

Error Type	Frequency	Percentage
Missing lexeme	41	37.9%
Turkish word	35	32.4%
Missing or erroneous morphotactic rule	13	12.0%
Typing errors	7	6.4%
Loanwords	7	6.4%
Missing or erroneous Phonological rule	5	4.6%
Total	108	

Table 7: Error analysis for randomly selected 100 tokens

noted that some of them may go into more than one category.

The highest percentage of unrecognized tokens belongs to the category of ‘Missing lexeme’. This is partly because of the fact that our lexicon for substantives was not large enough to account for the phonological and lexical differences for stems in different dialects. For example, *açkvaneri* ‘next time’ appearing in our corpus belongs to Xopa dialect but not to Pazar dialect according the the dictionary. It also includes lexemes which do not appear in the dictionary and consequently not in our lexicon as well as those which are simply missed out during the automatic extraction of words from the dictionary.

We still have certain morphotactic rules to work on to be able to cover inflectional morphology of Laz such as verb inflection for adverbial clauses such as the *-şa* suffix meaning ‘while’ (related to the allative suffix normally attached to nouns).

6 Future Work

Since we still have problems with verb stemming and separating substantives into nouns, adjectives and adverbs as well as determining other word classes and their inflectional morphology, our current lexicon can be manually checked and extended accordingly. We definitely need to improve the coverage of the morphological analyser not only for Pazar Laz but also for other dialects of Laz for the future studies. This requires both defining morphotactics for other dialects and carefully separating and including lexemes from the dictionary. It is also equally important to prepare a gold standard corpus for Laz to be able to evaluate the accuracy of the analyser. Additionally, investigating borrowed words from specifically Turkish and how they are adapted and used in Laz will also improve the results. Derivational morphology is also nontrivial to look into to expand the lexicon.

7 Concluding Remarks

We have presented the first ever morphological analyser for Laz, a language in the Caucasian language family spoken in Turkey. The analyser currently covers the Pazar dialect.

This study will hopefully lead to further studies for language documentation and revitalization efforts for Laz in a larger context.

All the up-to-date project files have been uploaded on `Github`[16] and licensed under the CreativeCommons BY-NC-SA 3.0.

8 Acknowledgements

We would like to express our sincere gratitude to İsmail Bucaklişi for his enormous contribution to the project as our native speaker informant as well as his help in finding language resources.

[16] https://github.iu.edu/esraonal/laz-morphological-analyser-fst

References

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite state morphology*. CSLI Publications.

Steven Bird. 2009. Natural language processing and linguistic fieldwork. *Computational Linguistics* 35(3):469–474.

İsmail Bucaklişi and Goichi Kojima. 2003. *Laz Grammar (Lazuri Grameri)*. Chiviyazilari.

Ciprian Gerstenberger, Niko Partanen, and Michael Rießler. 2017. Instant annotations in ELAN corpora of spoken and written Komi, an endangered language of the Barents Sea region. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*. Association for Computational Linguistics, pages 57–66.

Belma Haznedar. 2018. The living Laz project: The current status of the Laz language and Laz-speaking communities in Turkey.

Nurdan Kavaklı. 2015. Novus Ortus: The awakening of Laz language in Turkey. *İdil Journal of Art and Language* 4(16):133–146.

René Lacroix. 2009. *Description du dialecte laze d'Arhavi (caucasique du sud, Turquie) Grammaire et textes*. Ph.D. thesis, Université Lumière Lyon.

Krister Linden, Miikka Silfverberg, Erik Axelson, Sam Hardwick, and Tommi Pirinen. 2011. *HFST–Framework for Compiling and Applying Morphologies*, volume 100 of *Communications in Computer and Information Science*, pages 67–85.

Paul Meurer. 2009. A computational grammar for georgian. In Peter Bosch, David Gabelaia, and Jérôme Lang, editors, *Logic, Language, and Computation*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 1–15.

Hammam Riza. 2008. Indigenous languages of Indonesia: Creating language resources for language preservation. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Balkız Öztürk and Markus A. Pöchtrager. 2011. *Pazar Laz*. LINCOM.

Malgorzata Ćavar, Damir Ćavar, and Hilaria Cruz. 2016. Endangered language documentation: Bootstrapping a Chatino speech corpus, forced aligner, asr. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Term Based Semantic Clusters for Very Short Text Classification

Jasper Paalman
Jheronimus Academy of Data Science
j.v.paalman@tilburguniversity.edu

Shantanu Mullick
School of Industrial Engineering
Eindhoven University of Technology
s.mullick@tue.nl

Kalliopi Zervanou
School of Industrial Engineering
Eindhoven University of Technology
k.zervanou@tue.nl

Yingqian Zhang
School of Industrial Engineering
Eindhoven University of Technology
yqzhang@tue.nl

Abstract

Very short texts, such as tweets and invoices, present challenges in classification. Although term occurrences are strong indicators of content, in very short texts, the sparsity of these texts makes it difficult to capture important semantic relationships. A solution calls for a method that not only considers term occurrence, but also handles sparseness well. In this work, we introduce such an approach, the **T**erm **B**ased **Se**mantic **C**lusters (TBSeC) that employs terms to create distinctive semantic concept clusters. These clusters are ranked using a semantic similarity function which in turn defines a semantic feature space that can be used for text classification. Our method is evaluated in an invoice classification task. Compared to well-known content representation methods the proposed method performs competitively.

1 Introduction

Bag-of-words approaches (Harris, 1954) to text classification rely on measures of term occurrence, or co-occurrence, such as $tf \cdot idf$ (Salton and Buckley, 1988), log-likelihood (Dunning, 1993), and mutual information (Church and Hanks, 1990). Such methods, despite their enduring popularity, are well known for their shortcomings in dealing with numerous natural language issues, such as morphological, semantic, and other types of variation and ambiguity (Augenstein et al., 2017). These issues become more critical in very short texts, such as microblogs, chat logs, reviews and invoices, because of the lack of data and context that could provide more reliable measures and a source for semantic disambiguation.

Distributional semantic models (Baroni and Lenci, 2010) such as pre-trained word embeddings (Pennington et al., 2014; Grave et al., 2018) encode words as fixed sized real-valued vectors. Embeddings may address the issues of data sparseness and lack of extensive context, by providing a semantic representation model that is not as sensitive to literal word occurrence in the data, thus providing semantic information coverage of out-of-vocabulary words (Bojanowski et al., 2017). This is because embeddings may allow for any given word to be mapped to a real-valued vector (e.g. by using character n-grams), even if it hasn't been observed during training. Additionally, embeddings implicitly capture semantic, syntactic and lexical properties, thereby representing implicit relationships. In this way, embeddings provide a rich representation that is otherwise difficult to attain. In a short text scenario, despite early findings that a small corpus size may not be sufficient for representing the relationships between words and documents (Song et al., 2014), availability of pre-trained embeddings makes this issue less of a concern. Despite these advantages and growing popularity, embeddings trained on general language data usually do not perform well in (i) specialised domains (Liu et al., 2018; Kameswara Sarma, 2018) and in (ii) languages with richer morphological variation than English (Zervanou et al., 2014). In this work, we attempt to address these issues while exploiting the advantages of pre-trained word embeddings in a text classification task for very short, domain specific texts in a morphologically rich language, i.e., invoices in Dutch.

Our aim is to classify short invoice descriptions, in such a way that each class reflects a different group of products or services, as illustrated in the examples in Table 1. When considering such texts, the augmented information offered by embeddings is crucial, because such texts abound in ellipsis, grammatical errors, misspellings, and

Proceedings of Recent Advances in Natural Language Processing, pages 878–887,
Varna, Bulgaria, Sep 2–4 2019.

Text	Class
Vervoer Almere-Lille *Transport Almere-Lille*	Travel expenses
600GB SAS interne harde schijf *600GB SAS internal hard drive*	Automation hardware

Table 1: Example documents and classes

semantic variation. The inherent advantage of embeddings in dealing with out-of-vocabulary words presents, at the same time, the disadvantage of providing a text representation that does not focus on the importance of individual terms for the classification. Conversely, measures of term occurrence focus heavily on individual term importance but are very sensitive to variation. In very short text and domain-specific applications, where occurring terms are both strong indicators of the respective text class, as well as abound in variation, the preferred solution would combine embeddings to extracted terms. For example, for invoices, each occurring term in an invoice description is highly informative of the respective invoice text class. Hence a method is required that not only focuses on such terms, but also leverages the flexibility of embeddings. Our proposed method **T**erm **B**ased **Se**mantic **C**lusters (TBSeC) attempts to provide such a solution. The contribution of this paper lies in (i) combining the advantages of word embeddings with conventional term extraction techniques (ii) apply our method in an application domain not previously investigated, namely invoice text, which is characterised by specialised terminology and very short, elliptical and/or ungrammatical text, in a language that is morphologically richer than English and therefore posing an additional challenge in statistical approaches.

TBSeC proposes a two-stage methodology. In the first stage we use class-specific textual information to build semantic concept clusters. Concept clusters are vector representations of strongly related terms that are distinctive for a certain class. In the second stage, we compute cluster similarity scores on generated concept clusters for a given description. This serves as a ranking function that can be used in both unsupervised and supervised learning tasks.

The remainder of this paper is organized as follows: section 2 discusses related work; section 3 elaborates on the TBSeC method; section 4 outlines the experimental setup and section 5 discusses the results. We conclude with our main observations and suggestions for further work.

2 Related work

Text or document classification is defined as the assignment of text sections or entire documents to a predefined set of categories (Feldman and Sanger, 2007). For this purpose, algorithms process various types of text representations which are used as features for describing content. To our knowledge, invoice text classification has not been investigated previously[1]. Work related to text classification of short texts has been applied for microblogs (Singh et al., 2016; Ren et al., 2016; Missier et al., 2016), email subject classification (Alsmadi and Alhami, 2015) and spam detection (Bahgat et al., 2016).

Initial approaches to document content representation used counts of term frequency and inverse document frequency, $tf \cdot idf$ (Salton and Buckley, 1988), whereby frequently occurring terms are assumed to represent document content and inverse document term frequency scores select the most distinctive terms for a given document within a collection. Various subsequent approaches use variants of term occurrence measures with probabilities, such as $\chi 2$-test, log likelihood (Dunning, 1993) and mutual information (Church and Hanks, 1990), or attempt to combine statistical measures with various types of linguistic and stop-word filters, so as to refine the keyword results. Considerations regarding term ambiguity and variation also led to rule-based approaches (Jacquemin, 2001) and resource-based approaches exploiting existing thesauri and lexica, such as UMLS (Hliaoutakis et al., 2009), or WordNet (Aggarwal et al., 2018). Knowledge poor statistical approaches, such as Latent Semantic Analysis (Deerwester et al., 1990) and Latent Dirichlet Allocation (Blei et al., 2003) attempt to detect document content in an unsupervised manner while reducing the dimensionality of the feature space of other bag-of-word approaches, but are also sensitive to sparse data and variation in short texts.

The advent of large semantic resources in the form of pre-trained word embeddings (Pennington et al., 2014; Grave et al., 2018) gave rise to a new line of approaches employing word embeddings for document content representation, such as Word2Vec (Mikolov et al., 2013), FastText (Bojanowski et al., 2017), GloVe (Pennington et al.,

[1] An approach reported by Bartoli et al. (2010) focuses on image features of scanned invoices rather than the invoice text.

2014), and ELMo (Peters et al., 2018). Within this line of approaches, methods have also been developed using word embeddings specifically for document classification, such as task-oriented word embeddings (Liu et al., 2018) and word-sense based embeddings (Jin et al., 2016). Embedding models encoding words in documents or document sections have also been developed, such as Doc2vec (Le and Mikolov, 2014), Infersent (Conneau et al., 2017), Skip-thought (Kiros et al., 2015) and FastSent (Hill et al., 2016). Such type of embeddings can be employed to calculate the semantic similarity between texts, but the risk is that intricate word-specific semantic relationships are lost. Methods originating from text similarity research using word rather than document embeddings, such as Kusner et al. (2015); Kenter and De Rijke (2015); De Boom et al. (2016) attempt to address this issue. Embeddings have been also used for keyphrase extraction via supervised (Mahata et al., 2018) and unsupervised (Bennani-Smires et al., 2018) approaches.

Finally, the problem of variation and data sparsity with very short texts has been addressed in the past with query expansion approaches (Vechtomova, 2009). For text similarity purposes query expansion techniques have been used for document term augmentation, exploiting relevant search results and a matrix representation (Sahami and Heilman, 2006; Abhishek and Hosanagar, 2007) or a combination of search results page count difference and lexico-syntactic patterns derived from text snippets (Bollegala et al., 2007).

3 The TBSeC methodology

Our proposed method, TBSeC, consists of two stages: In the first stage we use class-specific textual information to build semantic concept clusters. Concept clusters are vector representations of strongly related terms that are distinctive for a certain class. In the second stage, we compute cluster similarity scores on generated concept clusters for a given description, thereby forming a semantic feature space. This serves as a ranking function that can be used in both unsupervised and supervised learning tasks.

3.1 Concept clustering

Concept clustering starts by extracting distinctive terms, particular to a class[2]. Distinctive terms are extracted based on normalized term frequency. For a given class, word embeddings belonging to found terms are used to form numerous clusters. Hence, for each class multiple clusters are created and each cluster can be seen as a group of terms that are closely related. Each cluster is transformed into a concept cluster vector by taking the respective word embeddings of terms and computing the mean over each dimension. This process is illustrated in Figure 1, with actual examples included and word embeddings indicated as vectors with dots.

Specifically, the method works as shown in Algorithm 1 for n distinct classes, k most frequent terms to be incorporated as cluster and normalized term frequency (i.e., Bag-of-words, B_{ij}) threshold t. A more detailed description is given underneath the algorithm. Algorithm line numbers refer to steps as denoted in Figure 1.

Terms that provide a clear distinctive value to a class are retrieved using term frequency and L1 normalization over rows and columns. Terms not appearing in the vocabulary of the embedding model, or having a score below the normalized threshold t are filtered out. Word embeddings for these selected terms are employed to create concept clusters for each class using the DBSCAN clustering model (Ester et al., 1996). Terms can either be included in a multi-term cluster through DBSCAN clustering or can be added as a single-term cluster when occurring frequently enough based on k. Ultimately, for each class i, concept clusters are created as a single vector equal to the averaged embedding of included terms.

3.2 Semantic cluster similarity

We adapt the similarity measure by Kenter and De Rijke (2015), which is based on the BM25 framework (Robertson et al., 2009), to propose our semantic cluster similarity measure. We combine idf scoring with word by word cosine similarity to calculate a weighted similarity score.

The Kenter and De Rijke (2015) function for calculating semantic text similarity between two sentences s_l and s_s is as defined as follows:

[2] In the case of invoice classification, each class refers to a specific expense type.

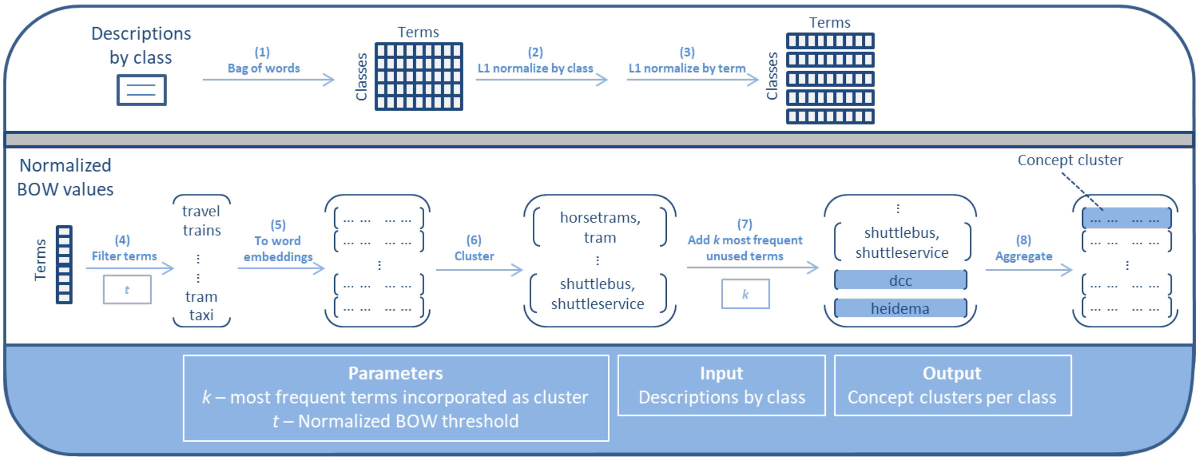


Figure 1: Concept clustering process diagram. *Upper panel*: Preparing normalized bag of words using descriptions by class. *Lower panel*: Employing found values to create concept clusters for a given class

$$f_{sts}(s_l, s_s) = \sum_{w \in s_l} IDF(w) \cdot \frac{sem(w, s_s) \cdot (k_1 + 1)}{sem(w, s_s) + k_1 \cdot (1 - b + b \cdot \frac{|s_s|}{avgsl})} \tag{1}$$

Parameters k_1 and b are inherited from the BM25 framework and serve as smoothing parameters. Parameter k_1 influences what semantic text similarity value is approached asymptotically, thereby limiting the influence that semantic term similarity can have. Parameter b provides a degree of importance to sentence length ratio $\frac{|s_s|}{avgsl}$, comparing sentence length to the average sentence length $avgsl$ of sentences being ranked. The function sem returns the semantic term similarity of term w with respect to text s, as follows:

$$sem(w, s) = \max_{w' \in s} f_{sem}(w, w') \tag{2}$$

where f_{sem} is a vector similarity function that is typically computed as cosine similarity.

In TBSeC, the Kenter and De Rijke (2015) function is adapted to compare each sentence to all composed concept clusters. In our implementation, parameter b is redundant, because each concept cluster is represented as a single embedding and the measure is computed by a single cosine similarity score. Moreover, we normalize our similarity score with the number of terms appearing in the sentence to allow for use in a supervised learning task. Finally, we remove term-specific weighting, for four reasons: First, idf scoring imposes a hefty constraint on the terms that can be used because of the predefined vocabulary. Second, we argue that the limited amount of terms appearing in a description justifies the exclusion of term-specific weighting. Each term in the description holds an important piece of information and differentiating is not essential. Third, terms that don't hold considerable semantic importance are not likely to steer the score towards an incorrect class. Each concept cluster is created to serve as a distinctive concept, thus making it unrealistic that unimportant terms will relate to it well. Fourth, terms in descriptions are subject to frequent misspellings and personal abbreviations, making idf scores inherently unreliable in this setting.

Based on the changes to Equation 1 discussed above, our function for calculating semantic cluster similarity f_{scs} between text s and concept cluster c is defined as follows:

$$f_{scs}(s, c) = \frac{1}{|s|} \cdot \sum_{w \in s} \frac{(sem(w, c) \cdot t(w)) \cdot (k_1 + 1)}{(sem(w, c) \cdot t(w)) + k_1} \tag{3}$$

where $t(w)$ is the term frequency of term w in the text. The score is normalized by the number of terms $|s|$ in text s. In addition to smoothing parameter k_1, two other hyper parameters sem_{th} and sem_{sq} are added to influence scoring. These hyper parameters affect the result of sem as follows.

$$sem(w, c) = \begin{cases} f_{sem}(w, c) & \text{if } f_{sem}(w, c) \geq sem_{th} \\ & \text{and } sem_{sq} = \text{false} \\ f_{sem}(w, c)^2 & \text{if } f_{sem}(w, c) \geq sem_{th} \\ & \text{and } sem_{sq} = \text{true} \\ 0 & \text{if } f_{sem}(w, c) < sem_{th} \end{cases} \tag{4}$$

Algorithm 1 Concept Clustering

Parameters:
k - No. of most frequent terms to be incorporated as cluster
t - Normalized bag of words threshold
Input:
D_i , $i \in \{1, \dots , n\}$ - Merged descriptions for each class
Output:
C_i , $i \in \{1, \dots , n\}$ - Concept clusters for each class

```
① $B_{ij} \leftarrow$ Calculate bag of words using $D_i$,
  with found vocabulary set $V$,
  $i \in \{1, \dots , n\}, j \in \{1, \dots |V|\}$
  $C_i \leftarrow$ Instantiate empty concept cluster array,
  with $i \in \{1, \dots , n\}$

  for $i \in \{1, \dots , n\}$ do
②     $B_{i*} \leftarrow$ L1 normalize $B_{i*}$          ▷ Normalize by class
  end for
  for $j \in \{1, \dots |V|\}$ do
③     $B_{*j} \leftarrow$ L1 normalize $B_{*j}$          ▷ Normalize by term
  end for
  for $i \in \{1, \dots , n\}$ do                    ▷ For each class
④     Retrieve terms in vocabulary with a score $> t$
      and occur more than once
      $terms \leftarrow$ Array of found terms that appear in
      word embedding vocabulary
⑤     $emb \leftarrow$ Respective word embeddings of $terms$
      $C_i \leftarrow$ CREATE CLUSTERS($emb, k, terms$)
  end for
  return $C$

  function CREATE CLUSTERS($emb, k, terms$)
      Instantiate $DBSCAN$ clustering model with
      $eps$, $min\_samples$ and
      $metric = cosine\ similarity$
      Fit $emb$ on $DBSCAN$ model
⑥     $clusters \leftarrow$ formed clusters as collections of
      word embeddings
      for $term$ in $k$ most frequent $terms$ do
          if $term$ not used in $clusters$ then
              $e \leftarrow$ word embedding of $term$
⑦             $cluster$.append([$e$])
          end if
      end for
      $concepts_c \leftarrow$ empty concept cluster array
      with $c \in \{1, \dots , |clusters|\}$
      for $c \in \{1, \dots , |clusters|\}$ do
⑧         $concepts_c \leftarrow$ coordinate mean over each
          dimension
      end for
      return $concepts$
  end function
```

The semantic threshold sem_{th} serves as a way to add a semantic similarity bound above which it will be presumed to hold importance. Parameter sem_{th} achieves that when $f_{sem}(w, c)$ is under the set threshold value, that $sem(w, c)$ equals 0. Squaring $f_{sem}(w, c)$ through sem_{sq} increases term importance of terms that are a near match and lowers importance of terms that match to a lesser extend. Squaring $f_{sem}(w, c)$ therefore promotes the divergence of semantic similarity scores.

Semantic cluster similarity f_{scs} produces features for use in supervised learning applications, but the initial performance of TBSeC is measured without a predictive model. For this reason, a similarity score for each class is required in order to rank the classes. This is calculated as scs for each class i, by extracting the maximum score over all concept clusters $c \in C_i$ (see C_i in Algorithm 1):

$$scs(s, C_i) = \max_{c \in C_i} f_{scs}(s, c) \tag{5}$$

4 Experimental setup

This section covers data description, data processing and the experimental set-up for our method.

4.1 Data

Our invoice data originate from an auditing company. In the data, as illustrated in the examples in Table 1, each class refers to a particular type of expenses. Available data is accessed from a data directory, where each file is specific to a client. For our purposes, only the *invoice description* and *class assignment* are relevant. The volume of the entire data directory amounts to approximately 1.5 million instances. There is a total of 111 unique classes to which assignments are made. The five classes that are least represented have 24, 106, 178, 418 and 452 entries respectively. Invoice descriptions on average contain 2.80 terms, with a standard deviation of 1.55.

4.2 Word embeddings

Pre-trained Dutch FastText word embeddings[3] are used for sentence embedding construction and for use in semantic similarity computations (Bojanowski et al., 2017). The FastText embedding model was trained on Dutch Wikipedia.

[3] https://github.com/facebookresearch/fastText/blob/master/docs/pretrained-vectors.md

4.3 Data processing

Descriptions are processed using a procedure similar to the one used in training the FastText model. Special characters are replaced with a whitespace, stopwords in both the English and Dutch language are dropped, digits are removed and finally terms are retrieved by splitting on any sequence of unicode whitespace characters. When creating validation sets special care is taken to remove duplicates and to include data from all individual clients and all classes in a randomized manner. As a result, largely balanced validation sets are formed with data from various sources.

4.4 Learning algorithm

During supervised learning a Support Vector Machine[4] is used with regularization parameter $C = 0.1$ and a linear kernel. This classifier performed best when compared to other feasible classifiers (e.g. random forest), given a local working memory bounded set-up, and allows for the use of sparse matrices. Regularization parameter C regulates the importance of focusing on correctly classifying training samples in favor of realizing a hyperplane with a large minimum margin. A high C can lead to overfitting, a low C can lead to the inability to learn meaningful decision boundaries. We set C to 0.1 since it appears to offer a good balance on the basis of the main validation set in terms of limited running time and general performance.

4.5 Parameter tuning

Prior to including our framework in a supervised learning task, we optimize the parameters (see section 5.1 for results). We construct an initial set of concept clusters using preset values $k = 5$ and $t = 0.8$. Parameters are set such that the model offers a well-performing baseline with low chances of overfitting. Initial concept clusters are used to tune semantic cluster similarity parameters. This order of parameter tuning is chosen, because it is relatively straightforward to pick sensible values for k and t, as opposed to f_{scs} hyperparameters. Table 2 lists the attempted combinations of parameter settings for f_{scs}.

Parameter	Distinct values
k_1	[1.2, 1.6, 2.0]
sem_{sq}	[true, false]
sem_{th}	[0, 0.1, 0.2, 0.3, 0.4]

Table 2: f_{scs} parameter settings

After parameter tuning for semantic cluster similarity, employed concept clusters are reconsidered. Values in the range from 5 to 50 with a step size of 5 are attempted for the k most frequent terms incorporated as cluster.

Validation is performed on a dataset with approximately 5,000 entries. Performance differences for f_{scs} are evaluated using three performance measures: (1) accuracy, (2) ranking loss and (3) standardized score. Accuracy is calculated by picking the class with the highest semantic similarity score as predicted class and finding the percentage of correctly classified instances. Ranking loss is calculated by obtaining the rank of the true class. The standardized score is calculated by standardizing all scores for a given instance and retrieving the score of the true class. By standardizing it can be observed how the score for the true class is positioned against all other scores. Ultimately, the objective is to maximize the accuracy and standardized score and to minimize ranking loss. Afterwards, we investigate the influence of parameter k for concept cluster construction on the basis of accuracy and dimension size. Accuracy is calculated as an unsupervised score as well as a 5-fold cross validated supervised score. Both scoring methods use best values for k_1, sem_{th} and sem_{sq} which are found in the previous step. Results are compared to determine an appropriate value for parameter k for use of features in a predictive model.

4.6 Invoice classification

We use the proposed semantic cluster similarity matching method, to measure performance in a classification task. We compare the performance to existing methods and we test whether combining methods leads to an increase in performance.

The parameters for generation of semantic cluster similarity scores are set in accordance with values obtained during parameter tuning.

For validation, a dataset containing approximately 20,000 entries is used. The data is used to perform 5-fold cross validation to determine overall performance. For all tests, we chose to use accuracy as performance quality measurement. Only one out of 111 classes shows significant imbalance, which for testing overall method performance is deemed negligible. Consequently, no balancing is performed and no alternative quality measure, such

[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Method	Parameters	Dimension size
tf · *idf*	-	117,766
tf	-	117,766
LSA	# components: 100	100
LDA	# topics: 200	200
FastText	-	300
P-mean	P-values: $\{-\infty, 1, 2, 3, \infty\}$	1500

Table 3: Benchmark methods

k_1	sem_{sq}	sem_{th}	acc	loss	std score
2.0	true	0.0	**21.46**	**28.08**	**1.5773**
2.0	true	0.1	**21.46**	28.09	1.5759
1.6	true	0.0	21.26	28.12	1.5412
1.6	true	0.1	21.24	28.13	1.5397
2.0	true	0.2	21.24	28.35	1.5127
1.6	true	0.2	21.26	28.40	1.4739

Table 4: Best f_{scs} configurations

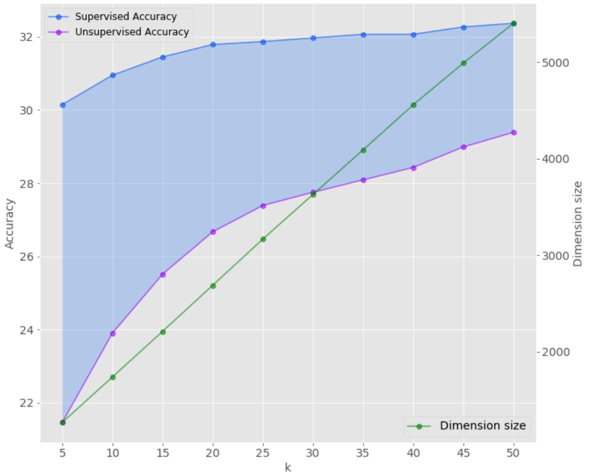


Figure 2: Concept clustering parameter tuning

as macro-averaged recall or F1-score is used.

Our benchmarks consist of other state-of-the-art content feature generation methods: term frequency (tf), *tf* · *idf*, Latent Semantic Analysis (*LSA*, Deerwester et al. (1990)), Latent Dirichtlet Allocation (*LDA*, Blei et al. (2003)), concatenated power mean word embeddings (*P-mean*, Rücklé et al. (2018)), and FastText sentence embeddings (Bojanowski et al., 2017). All benchmark methods, parameter settings and dimension sizes are shown in Table 3.

Feature transformation models (*tf*, *tf*·*idf*, LSA, LDA) are trained on the entire training directory. Additionally, when using such transformation models, words are stemmed to reduce inflectional word forms to their word stem.

5 Results

This section discusses our results from parameter tuning and the subsequent supervised classification task.

5.1 Parameter tuning

5.1.1 Semantic cluster similarity

Parameter combinations have been attempted and results have been retrieved for the used dataset. For each quality measure, the 3 best scores have been retrieved, returning all instances that conform to that score. The results are shown in Table 4.

Table 4 tells us that squaring the semantic similarity score works well. High accuracy scores are retrieved, accounting for the fact that evaluations are made based on maximum similarity scoring instead of a predictive model. As we are able to retrieve six configurations, we note that quality measures tend to score parameter settings similarly. The configuration with $k_1 = 2.0$, $sem_{sq} = true$ and $sem_{th} = 0.0$ across quality measures is the best performing.

5.1.2 Concept clustering

Next, we study the influence of parameter k on concept cluster construction. For each setting, we present the accuracy scores and dimension sizes in Figure 2.

With increasing k, more concept clusters of unused single terms are added. As a result, the dimension size steadily increases. We can see that accuracy also shows an increasing trend with the value of k. When more unused single terms are added, we run the risk of overfitting. The ratio of single terms to broader concept clusters increases with k, thereby relatively shifting the focus from broader concepts to frequently occurring distinctive individual terms. This behaviour is also reflected in the graph, in the relative performance difference between unsupervised and supervised scoring. Although supervised accuracy is undoubtedly higher with lower number of concept clusters, both accuracy scores converge with increasing k. By adding unused single terms, the unsupervised ranking method is able to capture an increasing number of edge cases, leading to a convergence in performance. This behaviour is an indication of overfitting.

After carefully considering the points above, we proceed with value $k = 20$. This choice appears to offer a good compromise between generalizing ability and direct performance gains. The primary

Methods	test acc	test sd	train acc	train sd
Benchmark				
tf · *idf*	43.93	0.89	82.79	0.07
tf	**44.85**	0.67	84.84	0.12
LSA	7.89	0.35	9.16	0.20
LDA	11.78	0.37	14.15	0.15
TBSeC	36.03	0.60	46.96	0.26
FastText	32.62	0.79	41.26	0.24
P-mean	39.78	0.63	90.33	0.1
Feature combinations				
tf & TBSeC	47.83	0.54	86.17	0.13
tf & FastText	47.07	0.48	85.81	0.14
tf & TBSeC & FastText	**47.86**	0.42	86.38	0.10

Table 5: Invoice classification results

goal of TBSeC is relating input to broader concepts, which is why a relatively moderate value for k is preferred.

5.2 Invoice classification

In this section, we discuss the results of our method against state-of-the art benchmark methods in a supervised invoice classification task. The results, consisting of cross validated accuracy scores with standard deviation (sd), are shown in Table 5 under header 'Benchmark'. The fact that term frequency has comparable performance to $tf \cdot idf$ reinforces the notion that all terms within an invoice description are important to take into account and that term-specific weighting has limited value. Moreover, techniques that are concerned with dimensionality reduction (LSA, LDA) perform worse, arguably because they truncate a large amount of information, most of which should have been retained. Sentence embeddings and our method TBSeC perform relatively well, with accuracy scores nearing performance levels of $tf \cdot idf$ and term frequency. Furthermore, a large feature space (1500D) that is achieved with P-mean embeddings appears to have a positive influence on the amount of information that is contained. It is also found that methods $tf \cdot idf$, term frequency and P-mean have a tendency to overfit on the data, having cross-validation training accuracy scores of over 80%. In comparison, TBSeC and FastText have training accuracy scores closer to test accuracy scores.

Combinations of techniques are attempted next to improve performance. The feature combinations are formed by concatenating the feature spaces of each method. The most successful combinations are highlighted in Table 5 under header 'Feature combinations'. Combining feature generation techniques leads to surprising results. P-mean performs well as sole feature, but doesn't yield better results when combined with other techniques. In contrast, FastText does pair well with other techniques and improves performance levels. Moreover, when TBSeC is used performance is even better. Term frequency in combination with both TBSeC and FastText does also improve performance, although slightly. TBSeC and FastText are more likely to be complementary as soon as TBSeC encompasses lower number of concept clusters, but this doesn't guarantee better overall performance. In the current configuration it manages to perform better, but the difference is not major. The tendency to overfit for $tf \cdot idf$, term frequency and P-mean is a likely cause for some of the other unsuccessful feature combinations.

6 Conclusion

A new feature generation framework TBSeC was presented that is suited to the prediction of well defined classes on the basis of very short texts (2.8 words on average). Generated features were proven to be able to function well independently and jointly with traditional feature generation techniques. Performance and reliability was improved by pairing multiple disjoint feature generation techniques, including TBSeC. A combination of highly specific features with more flexible ones was found to lead to the best results. Combinations of features were found to reach a bound in effectiveness, highlighting that methods ultimately start impeding each other. Businesses can use our method to derive actionable insights from online user generated content such as firm-specific tweets, online reviews and customer chats logs. Future work could test TBSeC on larger texts, since it offers more room to differentiate from sentence embeddings.

References

Vibhanshu Abhishek and Kartik Hosanagar. 2007. Keyword generation for search engine advertising using semantic similarity between terms. In *Proceedings of the ninth international conference on Electronic commerce*. ACM, pages 89–94.

Ayush Aggarwal, Chhavi Sharma, Minni Jain, and Amita Jain. 2018. Semi supervised graph based keyword extraction using lexical chains and centrality measures. *Computación y Sistemas* 22(4).

Izzat Alsmadi and Ikdam Alhami. 2015. Clustering and classification of email contents. *Journal of King*

Saud University-Computer and Information Sciences 27(1):46–57.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, pages 546–555.

Eman M Bahgat, Sherine Rady, and Walaa Gad. 2016. An e-mail filtering approach using classification techniques. In *The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Beni Suef, Egypt*. Springer, pages 321–331.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Alberto Bartoli, Giorgio Davanzo, Eric Medvet, and Enrico Sorio. 2010. Improving features extraction for supervised invoice classification. In *Artificial Intelligence and Applications*. MH Hamza.

Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Brussels, Belgium, pages 221–229. https://www.aclweb.org/anthology/K18-1022.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. *www* 7:757–766.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics* 16(1):22–29.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters* 80:150–156.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391–407.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19(1):61–74.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*. volume 96, pages 226–231.

Ronen Feldman and James Sanger. 2007. *The text mining handbook : advanced approaches in analyzing unstructured data*. Cambridge University Press.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483* .

Angelos Hliaoutakis, Kalliopi Zervanou, and Euripides G. M. Petrakis. 2009. The AMTEx approach in the medical document indexing and retrieval application. *Data and Knowledge Engineering* 68(3):380–392.

Christian Jacquemin. 2001. *Spotting and Discovering Terms Through Natural Language Processing*. MIT Press.

Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for text classification. In *IJCAI*. volume 16, pages 2824–2830.

Prathusha Kameswara Sarma. 2018. Learning word embeddings for data sparse and sentiment rich data sets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. Association for Computational Linguistics, New Orleans, Louisiana, USA, pages 46–53.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, pages 1411–1420.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*. pages 957–966.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. pages 1188–1196.

Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian, and Luyang Liu. 2018. Task-oriented word embedding for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 2023–2032.

Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. 2018. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. pages 634–639.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Paolo Missier, Alexander Romanovsky, Tudor Miu, Atinder Pal, Michael Daniilakis, Alessandro Garcia, Diego Cedrim, and Leonardo da Silva Sousa. 2016. Tracking dengue epidemics using twitter content classification and topic modelling. In *International Conference on Web Engineering*. Springer, pages 80–92.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Yafeng Ren, Ruimin Wang, and Donghong Ji. 2016. A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences* 369:188–198.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4):333–389.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated p-mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400* .

Mehran Sahami and Timothy D Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*. AcM, pages 377–386.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

Monika Singh, Divya Bansal, and Sanjeev Sofat. 2016. Behavioral analysis and classification of spammers distributing pornographic content in social media. *Social Network Analysis and Mining* 6(1):41.

Ge Song, Yunming Ye, Xiaolin Du, Xiaohui Huang, and Shifu Bie. 2014. Short text classification: A survey. *Journal of multimedia* 9(5):635–644.

Olga Vechtomova. 2009. *Query Expansion for Information Retrieval*, Springer US, Boston, MA, pages 2254–2257.

Kalliopi Zervanou, Elias Iosif, and Alexandros Potamianos. 2014. Word semantic similarity for morphologically rich languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*. pages 1642–1648.

Quotation Detection and Classification with a Corpus-Agnostic Model

Sean Papay & Sebastian Padó

Institute for Natural Language Processing
University of Stuttgart, Germany
{sean.papay,pado}@ims.uni-stuttgart.de

Abstract

The detection of quotations (i.e., reported speech, thought, and writing) has established itself as an NLP analysis task. However, state-of-the-art models have been developed on the basis of specific corpora and incorporate a high degree of corpus-specific assumptions and knowledge, which leads to fragmentation. In the spirit of task-agnostic modeling, we present a corpus-agnostic neural model for quotation detection and evaluate it on three corpora that vary in language, text genre, and structural assumptions. The model (a) approaches the state-of-the-art on the corpora when using established feature sets and (b) shows reasonable performance even when using solely word forms, which makes it applicable for non-standard (i.e., historical) corpora.

1 Introduction

Quotation is a general notion that covers different kinds of direct and indirect speech, thought, and writing in text (Semino and Short, 2004). Quotations are a prominent linguistic device used to express claims, assessments, or attitudes attributed to speakers. Consequently, the analysis of quotations is gaining traction in computational linguistics and digital humanities, providing evidence for speaker relationships (Elson et al., 2010; Agarwal et al., 2012), inter-speaker sentiment (Nalisnick and Baird, 2013), politeness (Faruqui and Pado, 2012), and narrative structure (Jannidis et al., 2018).

As is often the case with semantic phenomena, manual annotation of quotations has shown to be slow and resource-intensive, in particular when undertaken in conjunction with the annotation of speakers and information quality (Brunner, 2013; Pareti, 2015). This provides the rationale for automatic *quotation recognition* methods. After a first round of rule-based methods (Pouliquen et al., 2007; Brunner, 2013), recent supervised models use mostly sequence classifiers (Pareti et al., 2013; Almeida et al., 2014; Scheible et al., 2016).

Not surprisingly, these corpora differ substantially across a number of relevant dimensions. including text genre, annotation scheme, and theoretical assumptions. For example, Pareti et al. (2013) focus exclusively on newspaper text and focus on developing a *uniform* annotation schema that captures the shared properties of all kinds of annotations. Thus, even though this corpus contains direct, indirect, and mixed quotations, these not marked as instances of their specific subtypes. In addition, each quote is assumed to be introduced by a *cue* (markables are shown surrounded by red square brackets):

(1) Hillary Clinton on Saturday [acknowledged]$_{\text{CUE}}$ [the state of the economy is good]$_{\text{QUOTE}}$.

This assumption is generally true for newspaper text, and simplifies the task of quotation detection.

The situation is rather different in the literary texts considered by Semino and Short (2004). Cues are much more varied, and are sometimes omitted entirely, such as in this exchange from Dickens' *Christmas Carol*:

(2) ["Much!"]$_{\text{QUOTE}}$ – Marley's voice, no doubt about it.
["Who are you?"]$_{\text{QUOTE}}$
["Ask me who I was."]$_{\text{QUOTE}}$

The study follows a generally more *differentiating* approach. It develop and annotate a rich typology of different subtypes of quotations to distinguish, e.g., direct from indirect quotations, and speech from thoughts from writing.

Not surprisingly, therefore, all models for quotation detection were developed for one specific corpus. This leads to two problems:

1. The models inherit the corpora's *structural and theoretical assumptions*, such as the presence of a cue assumed by models for the Pareti et al. (2013) corpus.

2. The models typically include *domain-specific* features and knowledge sources that happened to be available from the corpus, such as lists of likely cue verbs or syntactic realizations of quotations.

This corpus dependence amounts to conceptual overfitting: while it leads to better fit for the original corpus, models are not transferable to new domains and analysis schemes. In other words, it leads to serious *fragmentation*.

In this paper, we develop and evaluate a *corpus-agnostic* neural model architecture for automatic quotation recognition that makes as few assumptions as possible about the corpus to be modelled but is still expressive enough to deal with the challenge of recognizing quotation spans of essentially arbitrary length (Scheible et al., 2016). In this respect, we see our study as a step towards transfer learning (Pan and Yang, 2009) and task-agnostic learning (Hashimoto et al., 2017). We find that our model can perform reasonably well across corpora differing in genre, language, and structure.

2 Related Work: Datasets and Models

We now review the state of the art in automatic quotation annotation, describing the three major quotation corpora for English and German and the corresponding models. We exclude corpora that focus on one specific quotation subtype such as the Columbia Speech Attribution corpus (Elson and McKeown, 2010) which only covers direct speech.

2.1 PARC Dataset

Dataset. The Penn Attribution Relation Corpus (Pareti, 2015), version 3 (PARC3) is a subset of the Penn Treebank, annotated with quotations and attribution relations. It consists of English newswire text from the Wall Street Journal. Each attribution relation consists of a cue, optionally a source (speaker), and content (quotation span), all marked as text spans. As part of the Penn Treebank, PARC3 provides manually annotated tokenization, POS tags, lemmas, and constituency parses.

Quotation spans are not labeled with more specific types, but PARC3 distinguishes informally (based on the surface form) between direct quotations (starting and ending with quotation marks), indirect quotations (without any quotation marks), and mixed quotations (everything else).

Pareti model. Pareti (2015), an extension of Pareti et al. (2013), presents a pipeline architecture for quotation annotation. It first applies a k-NN classifier to identify quotation cues within the corpus. Then, a linear-chain conditional random field (CRF) is used to identify quotation spans in the vicinity of each cue. The Pareti model builds on corpus-specific knowledge, including lists of cue verbs, and handcrafted features sensitive to punctuation conventions in English newswire text.

Scheible model. Scheible et al. (2016) retain the pipeline architecture of Pareti (2015) and its feature set, but replace the components. Cue annotation is performed with an averaged perceptron. More importantly, they replace quotation annotation proper with a sampling-based procedure: a perceptron samples tokens as likely span boundaries, which are then combined into complete quotation spans, using a semi-Markov model.

2.2 STOP Dataset

Semino and Short (2004) presents a corpus-based ontology of quotations in English text. It introduces two dimensions: (a), speech vs. thought vs. writing; and (b), direct vs. indirect vs. free indirect vs. reported, yielding a Cartesian product of twelve quotation subclasses. These are used to annotate the Speech, Thought, and Writing Presentation corpus (STOP). It comprises 120 sections, split evenly across three genres (fiction, newspaper, and biographies), of about 2,000 words each (Total size: 250,000 tokens; 8,000 quotations). The corpus has no linguistic annotation: the only features available are words' surface forms. To our knowledge, there are no models for this dataset.

2.3 *Redewiedergabe* Dataset

Dataset The Redewiedergabe ('reported speech') corpus (RWG) (Brunner, 2013) is a corpus of German narrative text, comprising thirteen public-domain German narratives (1787–1913). The quotation annotations in RWG adopt the scheme by Semino and Short (2004) and distinguish direct, indirect, free indirect, and reported variants of speech, thought, and writing. The total size of the corpus is 57.000 tokens, and 17.000 quotation spans.

Unlike STOP, RWG contains some linguistic information, namely POS tags, lemmas, and mor-

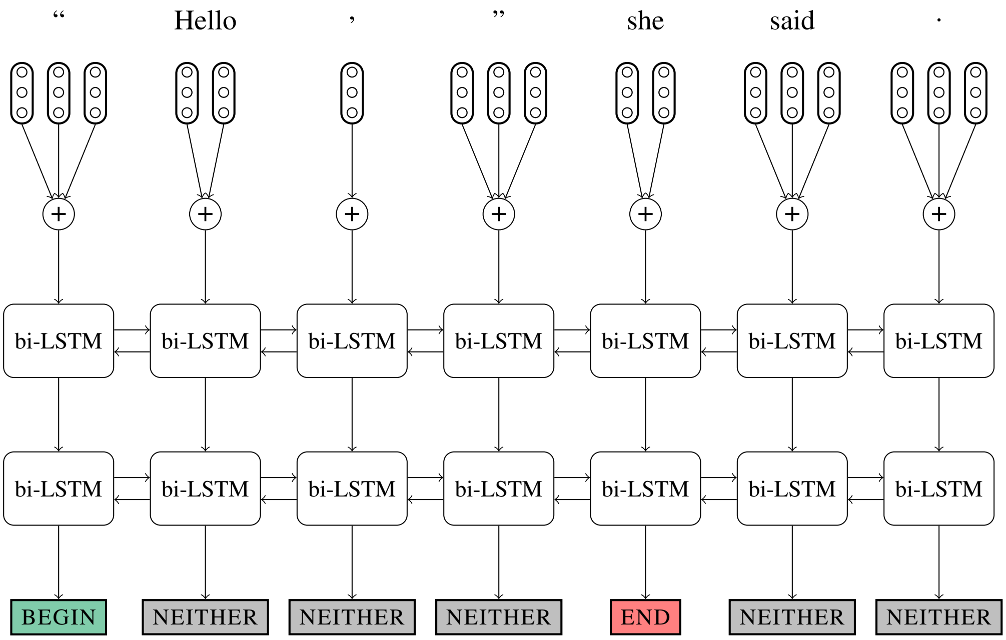


Figure 1: The NQD architecture. Tokens are represented as a bag of feature embeddings, and each token sequence is processed by a 2-layer bi-LSTM network, before a max-entropy classifier labels each token.

phological features (case, number, gender). This information is obtained automatically, though.

Models. Brunner (2013) proposes two models for quotation annotation on RWG. Both models work at the sentence level and predict only the presence of absence of quotations, not their spans (even though this information is annotated). The first model is rule-based (**Brunner RB**). It uses a set of handcrafted rules to identify direct, indirect, reported, and free indirect quotations. The second model (**Brunner ML**) is a simple classification model based on random forests.

3 Neural Quotation Detection (NQD)

We now define a neural architecture, NQD, with the goal of modeling the quotations in all three corpora described in Section 2. We design our model to leverage the commonalities across datasets, while not depending on the features of any dataset in particular. As all datasets involve long quotation spans with long-distance dependencies, an LSTM-based approach was natural, given such models' ability to capture very long-distance dependencies of up to 200 tokens (Khandelwal et al., 2018). Conversely, given the structural differences between corpora, we decided against a pipeline model like those employed by Pareti (2015) and Scheible et al. (2016) which predict cues first and then quotation spans. NQD predicts quotations directly without explicitly identifying cues.

NQD frames quotation prediction as token classification, classifying each token as either beginning a quotation (BEGIN), ending a quotation (END), or neither (NEITHER). Quotation spans then consist of all tokens starting with a BEGIN tag, up to (but not including) the next END or BEGIN tag, or the end of sequence. This model is not limited to the sentence level: it is able to make predictions for a whole document and in this manner can capture very long quotation spans (Scheible et al., 2016). Concretely, the sequence-to-sequence architecture comprises a 2-layer bi-LSTM network, with the outputs of the second bi-LSTM feeding into a 3-class softmax classifier. Thus, the model takes token sequences as input and produces a sequence of token labels. Figure 1 shows a schematic diagram of the NQD architecture. For datasets with multiple quotation types, NQD uses a separate sequence-to-sequence model for each span type, connecting them by weight sharing. All NQD code is available from https://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/index.en.html.

In the input token sequences, each token is a bag of features. Each feature value is represented as an n-dimensional continuous vector, and each token is represented as the sum of these vectors.

This approach to feature representation allows our model to work with corpora with arbitrary types of token-level features. In the simplest case, when only raw text is present in the corpus, each token is given a single feature for that token's word. If other token-level features are present, such as POS-tags, lemmas, or even parse tree information, these can be incorporated as additional feature vectors, without requiring any changes to the model architecture. Feature vectors can also be initialized to pre-trained representations (e.g. word embeddings) when these are available, or initialized randomly and learned when they are not. Section 2 describes in detail which features are used for the corpora we experiment with.

4 Experimental Evaluation

We now train and test NQD on the three corpora and compare against the state-of-the-art.

4.1 Experimental Setup

PARC3. For PARC3, we train a single classifier on the quote content spans and ignore the cue and source spans. As features, we use token surface forms, lemmas, POS tags, as well as, for each token, the bags of constituents that start with, end with, and contain it. These features are a subset of the features used by Scheible et al. (2016) and Pareti (2015), and like these studies, we use gold standard annotation. We initialize the features for word surface forms with the default GloVe Wikipedia word embeddings (Pennington et al., 2014). Our model makes predictions on entire documents at a time. We use performance on the corpus's development set to guide early stopping during training, and we evaluate on the corpus's test set.

STOP. For STOP, we train four classifiers for the four quote types (direct, indirect, free indirect, reported). We train and evaluate our model on a per-document basis as for PARC3. We use word surface forms (and their GloVe embeddings) as features, we used no other features in this model. As the corpus contains no held-out development or test sets, we used 10-fold cross validation to evaluate our model, using 8 folds for training, 1 for development, as 1 as for testing in each iteration.

RWG. For RWG, we adopt the same four-classifier setting as for STOP, using the word, lemma, POS, and morphological features available. For the sake of comparability with (Brunner, 2013), we train and evaluate on individual sentences, as opposed to entire documents. We use 10-fold cross validation again, randomly partitioning all corpus sentences into 10 subsets. We use GloVe embeddings pre-trained on the German Wikipedia.[1]

Evaluation. Previous studies on PARC3 adopted an exact span match setting, i.e., only those predicted spans that exactly match a gold standard span count as true positives. We report precision, recall, and F_1 in this setting for PARC3 and STOP. For RWG, we report the sentence-level accuracy used by Brunner (2013). In this mode, we train and predict with our model as before, but for evaluation we just record whether the model predicts the presence of a quotation type in a sentence.

4.2 Results

PARC3. The results in Table 1 show that NQD cannot beat the performance of Scheible et al. (2016), but does almost as well as Pareti et al. (2013). Given that our model is substantially simpler than either of these two (both include a special cue classifier, dictionaries, etc.), we see this as a success. Our model is competitive with the Scheible et al. model with regard to recall, but shows subpar precision for all quotation types, indicating a remaining weakness in the input encoding: for direct quotations, quote characters should provide strong indicators for quotation boundaries.

Note that these results, as well as the earlier studies (Pareti et al., 2013; Scheible et al., 2016), use unrealistic gold standard features. Therefore, we ran a second version of NQD using only word features, but no tags or structural information. The model is clearly worse, but still surprisingly good at 61% F_1. Not surprisingly, we see the highest drop for indirect quotations, which are most sensitive to syntactic structure. This indicates that NQD does a reasonable job in a setting that is realistic in general, and particularly so for non-standard corpus varieties (e.g., historical and literary corpora) that are often used in Digital Humanities.

STOP. To our knowledge, the results in Table 2 are the first modeling results on STOP. In comparison to PARC3, the results are noticeably lower. It is still the case that direct quotations are easiest to find, but their F_1 is somewhat lower than in PARC3. Indirect quotations are much more difficult, and free indirect quotations essentially impossible. This involves multiple factors: (a) STOP is

[1] Available from deepset at `https://deepset.ai/german-word-embeddings`

Model	Features	Overall			Direct			Indirect			Mixed		
		Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1
Pareti et al. (2013)	word, syn, domain	63	**80**	71	88	**94**	91	56	**78**	65	60	67	63
Scheible et al. (2016)	word, syn, domain	**71**	79	**75**	93	**94**	**94**	**64**	73	**69**	68	**81**	**74**
NQD	word, syn	**71**	67	69	**94**	82	88	**64**	64	64	**70**	59	64
NQD	word	61	61	61	90	84	87	53	56	54	60	54	57

Table 1: Results on PARC3 (exact span match evaluation)

Model	Features	Overall			Direct			Indirect			Free Indirect			Reported		
		Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1
NQD	word	51	66	57	78	83	80	33	49	40	01	04	01	46	58	51

Table 2: Results on STOP (exact span match evaluation)

Model	Features	Overall			Direct			Indirect			Free Indirect			Reported		
		Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1	Rec	Prec	F_1
Brunner (2013) RB	word, syn	**71**	67	**69**	**87**	81	84	**62**	**81**	**71**	**44**	24	31	**64**	51	**57**
Brunner (2013) ML	word, syn	63	77	**69**	85	**88**	**87**	47	62	53	29	63	40	45	**56**	50
NQD	word, syn	60	**78**	68	77	86	82	52	69	60	31	**68**	**42**	34	**56**	43
NQD	word	59	73	65	77	83	80	40	69	50	14	62	23	41	50	45

Table 3: Results on RWG (sentence-level accuracy evaluation)

significantly smaller, but more varied, than PARC3, providing sparser training data; (b) STOP covers a wider variety of quotation types, some of these are intrinsically difficult to model – in particular free indirect quotations (McHale, 2009).

RWG. The results in Table 3 show a picture that is overall similar to PARC3:[2] NQD does not outperform the state-of-the-art, but approximates it closely despite the lack of corpus-specific tuning. As in STOP, we see the lowest results for free indirect quotations, showing that this class is generally hard to classify. In general, even though this resource's size and annotation are similar to STOP, we see significantly higher numbers. This is mostly due to the different evaluation we use for RWG to compare to previous work: detecting the presence of quotes is easier than identifying their spans.

On RWG, we also run a basic NQD with only word form information. With this corpus and evaluation, this results in a drop of merely 3 points F_1 – due to losses on the indirect and free indirect categories – which bolsters the potential of this configuration.

5 Error Analysis

To gain some insights into the failure modes of NQD, we perform a brief qualitative analysis of the cases where our model gave false predictions.

These errors can broadly be divided into three categories: cases where the model predicts the presence of extraneous quotations (false positives), cases where the model fails to identify existing quotations (false negatives), and cases where the model correctly identified the presence of a quote, but did not correctly determine its boundaries (boundary mismatch, leading both false positives and false negatives in our exact span evaluation). We focus our error analysis on PARC, the previously best explored of our three corpora. In the examples, gold-standard quotations are marked with red square brackets, as above, and model-predicted quotations are marked with blue parentheses.

5.1 False Positives

Among the false positives produced by our model was a surprising number of quotations that are correct according to PARC's guidelines, but which are not annotated in the corpus. As an example, our model correctly identifies the presence of an

[2] Brunner (2013) does not report overall results. We compute them as micro-averages over reported per-type results.

unannotated quotation in the following sentence:

(3) (Britain's retail price index rose 0.7% in September from August and was up 7.6% for the year), the Central Statistical Office said.

Outside of these cases, proper false positives seem to be rare. Many of the false positives we found were boundary mismatches, discussed separately below.

5.2 False Negatives

Among the false negatives we analyzed, we found that the model is most likely to miss "tricky" quotations that are unusual in their grammatical structure. In particular, it tends to miss a class of quotations that are expressed as short noun phrases or adjectival phrases embedded within a non-quotation sentence such as

(4) Mandela, considered [the most prominent leader of the ANC] remains in prison. But [his release within the next few months] is widely expected.

According to the PARC guidelines, these are cases of quotations since they are attributable statements, but they are difficult for the model to retrieve since they are hard to distinguish from "non-quotation" nominal phrases – in particular in cases like this one, where there are not even overly realized speakers. In STOP and RWG, these cases might arguably not even be annotated as quotations.

5.3 Boundary Mismatches

A large proportion of the errors of NQD are boundary errors, where the model identifies the presence of a quotation, but fails to identify its exact boundaries. This can happen when our model correctly predicts one quotation boundary, but not the other.

For example, in the following sentence, our classifier identified the first quotation's beginning, but not its end (it also failed to identify the second quotation entirely – a false negative):

(5) He reiterated ([his opposition to such funding], but expressed [hope of a compromise].

This type of error occurs both for noun phrases and verb phrases and embedded sentences, but for different reasons: noun phrases are difficult to recognize since they are not marked by punctuation as are almost all other cases of quotation spans; verb phrases, on the other hand, can become arbitrarily complex. In the case above, the segmentation problems are exacerbated by the fact that the noun phrase quotation span occurs in a complex syntactic environment involving coordination.

6 Conclusion

In this paper, we have argued that existing models of automatic quotation annotation suffer from the tight relation between corpus annotation and model properties in particular in terms of model reusability. As an alternative, we have presented a general neural architecture, NQD, that can be trained "as is" on various corpora that differ in terms of genre, structure, and language. While the model does not reach the state of the art on any particular corpus, it performs close to it on all of them. Notably, the model is also able to deal relatively graciously with the absence of linguistic information. We will release an implementation with pre-trained models.

As NQD makes independent predictions for each token, it cannot model correlations and mutual exclusions between labels, and there is no guarantee for well-formed output class sequences. We investigated a number of extensions, including linear-chain CRF layers that are effective for Named Entity Recognition (Lample et al., 2016), but did not obtain improvements. We believe this is due to the unbounded length of quotation spans which is challenging for CRFs (Scheible et al., 2016).

The overall greatest challenge that NQD faces is data scarcity — all existing corpora with manual annotation are small, and our results show consistently bad performance for infrequent quotation types. In this situation, transfer learning seems like a natural proposition, and our model makes it possible for the first time to apply straightforward transfer learning to quotation annotation. In future work, we will explore this direction.

References

Apoorv Agarwal, Augusto Corvalan, Jacob Jensen, and Owen Rambow. 2012. Social network analysis of Alice in Wonderland. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 88–96, Montréal, Canada. Association for Computational Linguistics.

Mariana S. C. Almeida, Miguel B. Almeida, and André F. T. Martins. 2014. A joint model for quotation attribution and coreference resolution. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 39–48. Association for Computational Linguistics.

Annelen Brunner. 2013. Automatic recognition of speech, thought, and writing representation in German narrative texts. *Literary and Linguistic Computing*, 28(4):563–575.

David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden.

David Elson and Kathleen McKeown. 2010. Automatic attribution of quoted speech in literary narrative. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Manaal Faruqui and Sebastian Pado. 2012. Towards a model of formal and informal address in English. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–633, Avignon, France.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933, Copenhagen, Denmark.

Fotis Jannidis, Albin Zehe, Leonard Konle, Andreas Hotho, and Markus Krug. 2018. Analysing direct speech in German novels. In *Proceedings of DhD*, Cologne, Germany.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT*, pages 260–270, San Diego, CA.

Brian McHale. 2009. Speech representation. In Peter Hühn, John Pier, Wolf Schmid, and Jörg Schönert, editors, *Handbook of Narratology*. De Gruyter.

Eric T. Nalisnick and Henry S. Baird. 2013. Character-to-character sentiment analysis in Shakespeare's plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–483, Sofia, Bulgaria. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Silvia Pareti. 2015. *Attribution: A Computational Approach*. Ph.D. thesis, University of Edinburgh.

Silvia Pareti, Tim O'Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 989–999, Seattle, WA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Bruno Pouliquen, Ralf Steinberger, and Clive Best. 2007. Automatic detection of quotations in multilingual news. In *Proceedings of Recent Advances in Natural Language Processing*, pages 487–492, Borovets, Bulgaria.

Christian Scheible, Roman Klinger, and Sebastian Padó. 2016. Model architectures for quotation detection. In *Proceedings of ACL*, pages 1736–1745, Berlin, Germany.

Elena Semino and Michael Short. 2004. *Corpus Stylistics: Speech, Writing And Thought Presentation In A Corpus Of English Writing*. Routledge Advances In Corpus Linguistics. Routledge, London.

Validation of Facts Against Textual Sources

Vamsi Krishna Pendyala Simran sinha Satya Prakash Shriya Reddy Anupam Jamatia
Department of Computer Science and Engineering
National Institute of Technology
Agartala, Tripura, India
{krishnapendiala, simsinha32}@gmail.com
{satyaprakash30497, shriyavipul90, anupamjamatia}@gmail.com

Abstract

In today's world, the spreading of fake news has become facile through social media which diffuses rapidly and can be believed easily. Fact Checkers or Fact Verifiers are the need of the hour. In this paper, we propose a system which would verify a claim(fact) against a textual source provided and classify the claim to be true, false, out-of-context or inappropriate with respect to that source. This would help us to verify a fact as well as know about the source of our knowledge base against which the fact is being verified. We used a two-step approach to achieve our goal. First step is about retrieving the evidence related to the claims from the textual source. Next step is the classification of the claim as true, false, inappropriate and out of context with respect to the evidence using a modified version of textual entailment module. The accuracy of the best performing system is 64.95%.

1 Introduction

Fact Checking is one of the biggest buzzwords of this era. Many a time, the news transmitted through social media can be moulded and altered when so many people share information and it is hard to discern between fact and fiction. So there is a need to verify every piece of information we observe in our day-to-day life to be true or false. A solution to this problem is, we check each claim or fact manually against a reliable source and then label the claim or fact to be either true or false which is time consuming for large data. Vlachos and Riedel (2014) discussed the fact verification process as an ordinal text classification task, where they created a data-set using the manually annotated data present on sites like PolitiFact, FactCheck, FullFact. The FakeNews Challenge[1] by Riedel et al. (2017) addresses fact-checking as a simple instance detection problem, which mainly checks whether the given instance is in accordance with the article or not. Recently Thorne et al. (2018a) published a huge data set to deeply understand the process of large scale fact-checking. All of the above approaches rely completely upon the source against which data is to be verified, but in some cases, the source text might contain limited or no amount of information about the claim and a claim might be *Out-of-Context* of this source text. This leads to a problem of classifying a claim to be within the scope/context of the source text or not. Concretely, we do need a system which would not only classify a claim to be true or false but also check whether the source text is sufficient enough to classify the claim. To address this issue, in this paper, we come with an approach to verify facts or claims against a reliable source and classify them into 4 different classes.

In our approach, a claim or fact is classified as *True* (if a proper supporting evidence is available from the source text), *False* (if a contradicting evidence is available from the source text), *Inappropriate* (nothing can be concluded about the claim based upon evidence retrieved) or *Out-of-Context* (out of the scope of the source text). Prior to the classification, the evidence is retrieved from the

[1] www.fakenewschallenge.org/

textual source related to a given claim. If the textual source has no knowledge about the claim, it would be labelled as '*Out-of-Context*' for example if we have a source text about sports and we need to verify a claim related to politics then the claim is *Out-of-Context*. Although if it has some information about the claim it would further be classified as *True*, *False* or *Inappropriate*. The result would be '*True*' if the source supports the claim, it would be '*False*' if the source opposes the claim and it would be labelled as '*Inappropriate*' if the evidence is not sufficient to conclude the classification of the claim, e.g., knowing that Michael Jackson was a singer, we cannot infer whether his children will be singers. This is something inappropriate to the information provided. Hence the claim 'Michael Jackson's children would be singers' is an inappropriate claim for the evidence 'Michael Jackson was a Singer'. This labelling would not only help in knowing about the claim but also helps us to know about our textual source as well. The labels *Inappropriate* and *Out-of-Context* claim tell us whether the textual source has enough information to conclude about the claim or not, if a claim is *Out-of-Context* then the domain of our source text can be expanded to answer the claim. *Inappropriate* means that we do have required knowledge about the claim and also cannot conclude anything based upon this available knowledge. The best performing version of our system is seen to be giving 63.06% accuracy.

In Section 2 we have mentioned various works done by different authors related to fact extraction and verification. Section 3 explains the data-set collection and the system architecture to classify claims against a textual source. Section 4 describes the stages of this experiment which are evidence retrieval, similarity measures used to obtain the relation between claim and evidence, the first level of classification of the claim into in-context and out-context, then the further classification of in-context claims to *True*, *False* and *Inappropriate* claim. In Section 5 we discussed error analysis of our system along with the results obtained on using different classification algorithms on our data for classification purpose and comparative measure between different models. In Section 6 we conclude about our model with its practical usage in the real world.

2 Related Work

There has been a substantial amount of work done in the field of fact verification. Vlachos and Riedel (2014) provided the first dataset related to fact verification containing 211 labelled claims in the political domain with evidence hyperlinks. An alternative is Wang (2017) which released a dataset called LIAR dataset for detecting fake news, which contains 12.8K claims labelled manually using POLTIFACT.COM[2] on different context. Alhindi et al. (2018) extended the LIAR dataset and labelled a claim using the speaker related metadata without using the evidence. Basically, they used the justification given by the humans at the end of the article in the summary. Modelling the extracted justification along with the claim yielded better results rather than using a machine learning model for binary classification and a six way classification. Ferreira and Vlachos (2016) later presented a new modified dataset known as Emergent, where they had 300 claims and 2,595 related articles and they came to the conclusion that fact verification can also be treated as Natural Language Inference Task, as they used textual entailment to predict whether the article supports the claim or not. The latest large scale dataset is prepared Thorne et al. (2018a) which was annotated manually and used for verification against textual sources. It contains 185,441 claims generated from Wikipedia. These claims were classified Supported, Refuted and Not Enough Info by annotators. In this, Recognizing Textual Entailment (RTE) component was proceeded by an evidence retrieval module. The accuracy measured found to be 31.87% if evidence was taken into consideration and 50.91% if evidence is ignored. The only drawback of this system is its restriction to the Wikipedia domain. Thorne and Vlachos (2018) conducted a survey on automated fact checking research stemming using natural language processing and other related fields. According to this survey, the inputs for verification system play a vital role. Evidence retrieval plays a vital role in solving the fact verification problem. Fact checking requires the apt evidence against which sentences can be predicted to be true or false. Chen et al. (2017a) provides a framework for open domain question answering upon Wikipedia and SQuAD data set. This involved machine reading along with the document

[2] www.politifact.com

retrieval and then identifying the answers. We deal with a similar retrieval problem like open domain Question Answering, which would be succeeded by verification using textual entailment. Natural Language Inference is basically a task to find out whether a hypothesis entails, contradicts or is neutral about the claim. There have been recent developments in these fields like the SNLI dataset for learning natural language inference built by Bowman et al. (2015). Different neural NLI models (Nie and Bansal (2017); Parikh et al. (2016); Chen et al. (2017b); Gong et al. (2018)) that achieve promising performance. Parikh et al. (2016) has the highest accuracy on the Stanford Natural Language Inference task. We used the similar approach as the FEVER (Thorne et al., 2018c) but, instead of a three class classification we have extended it to a four class classification namely *True*, *False*, *Inappropriate* and *Out-of-Context*. These two modules combined together help us in validation of a fact. In 2018, a shared task known as the FEVER Shared Task (Thorne et al., 2018b) was held which dealt with the fact verification problem. The FEVER shared task is closely related to our work as it uses the same two modules. The following are some of the systems that participated in the FEVER shared task: Nie et al. (2018) have scored the maximum of 64% accuracy in the FEVER shared task, in which they used the neural semantic matching networks. For both the evidence retrieval and RTE model they enhanced the working using the neural networks. Hidey and Diab (2018) known as the Team Sweepers, made the evidence retrieval system better using lexical tagging and syntactic similarity. They used multitask learning and trained both the components together and set the parameters in a way using reinforcement learning so that it can first find sentences related to the claim and then find their relation with the claim. DeFactoNLP (Reddy et al., 2018) aimed at retrieving the evidence for the valuation of the claim from Wikipedia. The retrieval of documents which is considered as evidence is done by TF-IDF vectors of the claim and the sentences in the documents followed by inputting them to a textual entailment recognition module. Then the Random forest classifier is used for the classification of the claim. Lee et al. (2018) have introduced a method by developing a neural ranker using decomposable attention model and lexical tagging instead of TF-IDF for evidence retrieval part. Lexical tagging is done by using two lexical tags name such as Parts-of-Speech and Named Entity Recognition to enhance the performance.

3 System Architecture

In this section, we discuss the overview of our system and our approach for classifying a claim based upon a particular source text. Our approach is divided into two stages: Evidence Retrieval and Classification of claim as *True*, *False*, *Inappropriate* or *Out-of-Context* using a textual entailment module. The reason for using textual entailment is because it precisely gives us a relationship between an evidence and a claim. In the first stage i.e., evidence retrieval, given a claim, we find its TF-IDF vectors corresponding to the source text against which it is being verified.

Later we find out the cosine similarity between the TF-IDF vector of a claim to each of the sentences present in the source text. Then, we filter out the top four sentences which have the highest cosine similarity values and consider this as the evidence for that particular claim as discussed in section 3.2. The reason for considering top 4 sentences is that they closely correspond to the nearest sentences to the claim. In the next stage, the extracted evidence and the present claim are passed into a Textual Entailment module which returns the probabilities of two texts entailing, contradicting or neutral towards each other. These probabilities along with other variables discussed later are used as a feature vector for our classification model. The entire claim classification process is explained in 3.3. Section 3.1 describes the process of preparation of the dataset.

3.1 Dataset

Due to the uniqueness of our classification, we were supposed to either prepare our own dataset or modify an existing standard dataset for serving our purpose. Here, we have done both, we modified a dataset known as the SICK dataset and prepared a new dataset called the NITA dataset.

3.1.1 SICK Dataset

The main target of our experiment was to classify a claim based upon its evidence, so we required a dataset consisting of sentence pairs and a correlation between these two sentences. Hence we used a publicly available dataset known as the SICK dataset. The SICK-2014 dataset (Marelli et al., 2014) was introduced as Task 1 of the SemEval

ID	Sentence1	Sentence2	Entailment	Score
23	A group of kids is playing in a yard and an old man is standing in the background	A group of boys in a yard is playing and a man is standing in the background	yes	4.5
14	A brown dog is attacking another animal in front of the man in pants.	Two dogs are fighting.	unknown	3.5
13	Two dogs are wrestling and hugging.	There is no dog wrestling and hugging.	no	3.3

Table 1: Sample SICK dataset with entailment labels and relatedness scores.

Source Text	Claim/Fact	Label
The Lion King	Mufasa is Father of Simba	True
The Lion King	Ram killed Ravan	Out of Context
The Lion King	Cats hate Lions	Inappropriate Claim.

Table 2: Sample NITA dataset.

2014 conference and in contrast to SNLI (Bowman et al., 2015), it is geared at specifical benchmarking semantic compositional methods, aiming to capture only similarities on purely language and common knowledge level, without relying on domain knowledge, and there are no named entities or multi-word idioms. It consists of total 10,000 pairs of sentences.

We modified the SICK dataset as per our classification by adding two more columns to it. We manually labelled a claim and an evidence pair to be in-context or out-context based upon their relatedness score as given in the dataset, which indicates the semantic similarity of these pair of sentenced. Firstly, we labelled all the pairs with relatedness score less than 3 as *Out-of-Context* and other claims as *True*, *False* or *Inappropriate* based upon their textual entailment labels provided by the SICK dataset.

3.1.2 NITA Dataset

After considering SICK dataset we even wanted to develop our own dataset consisting of source texts and claims along with their labels as follows:

- **Source Text Collection**: We collected some short stories and articles related to *sports, movies, mythology, moral stories, Wikipedia articles* in English language and considered them as source texts. The total number of source texts collected in this way turned out to be 53 .

- **Claim Generation**: Corresponding to these 53 stories/articles/textual content, we prepared a total of 928 claims. The purpose was to generate claims about a single fact which could be arbitrarily complex and allowed for a variety of expressions for the entities. The claims were generated based upon every source text. For example, consider "The rabbit tortoise race" as the source text, one of the claims related to this source text can be "Rabbit won the race".

- **Claim Labelling**: Classifying whether a claim is *True*, *False*, *Inappropriate* or *Out-of-Context* based on the evidence from source text was done at this stage. We checked every claim manually with respect to its source text and labelled the claim accordingly. The labelling is done as per the meaning of each label which was discussed in the introduction section.

- **Dataset Validation**: Considering the complexity of labelling of claims, we considered validation of the data set generated by us. For this purpose we tried to analyse the labels we gave to each claim, where labels generated by one person were analysed by other to establish an inter annotator agreement. We considered around 30% that is 240 claims for this validation process and calculated the Fleiss k score (Fleiss, 1971) to be 0.876.

LABEL	No. of Claims
TRUE	170
FALSE	170
OUT-OF-CONTEXT	420
INAPPROPRIATE	168
Total No. of Claims	**928**

Table 3: NITA Dataset Splitting based upon Labels

3.2 Evidence Retrieval

We used the concept of Document Retrieval from the DrQA system (Chen et al., 2017a). Firstly, we find out the Term Frequency-Inverse Document Frequency (TFIDF) vectors (Hiemstra, 2000) for a claim and the sentences of the source text. We then calculate the cosine similarity between the claim and each sentence. Thereafter, we pick the top four similar sentences based on the cosine similarity between the bigram TF-IDF vectors of the sentences and the claim. These sentences are finally chosen as possible sources of *evidence*. Now, we are left with claim and evidence pairs.

3.3 Classification

In this final stage, we classify all the claims to be *True*, *False*, *Inappropriate* or *Out-of-Context* using machine learning classification models. The features for this classification are obtained by passing a claim and an evidence to a textual entailment module in order to obtain probabilities of entailment, contradiction and neutrality between claim and evidence. The RTE is the process of determining whether a text fragment (Hypothesis H) can be inferred from another fragment (Text T) (Sammons et al., 2012). The RTE module receives the claim and the set of possible evidences from the previous stages. Let there be 'n' possible sources of evidence for verifying a claim. For the ith possible evidence, let $\mathrm{s_i}$ denote the probability of it entailing the claim, let $\mathrm{r_i}$ denote the probability of it contradicting the claim, and let $\mathrm{u_i}$ be the probability of it being uninformative. The RTE module calculates each of these probabilities. The SNLI corpus (Bowman et al., 2015) is used for training the RTE model. This corpus is composed of sentence pairs T, H where T corresponds to the literal description of an image and H is a manually created sentence. If H can be inferred from T, the "Entailment" label is assigned to the pair. If H contradicts the information in T, the pair is labelled as "Contradiction". Otherwise, the label 'Neutral' is assigned. We chose to employ the state-of-the-art RTE by (Parikh et al., 2016). We selected this because at the time of development of this work, it was one of the best performing systems on the task with publicly available code.

For a particular claim *c* and an evidence *e* let s_i denote the probability of it entailing the claim, let r_i denote the probability of it contradicting the claim, and let u_i be the probability of it being uninformative returned by textual entailment. Below are some variables we considered for our convenience:

$$cs_i = \begin{cases} 1 & \text{if } s_i \geq r_i \text{ and } s_i \geq u_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$cr_i = \begin{cases} 1 & \text{if } r_i \geq s_i \text{ and } r_i \geq u_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$cu_i = \begin{cases} 1 & \text{if } u_i \geq s_i \text{ and } u_i \geq r_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$CosineSimilarity = \Big\{ cos(\theta) = \frac{C.E}{||C||||E||} \tag{4}$$

The similarity variable used here is cosine similarity between claim and evidence. The value C and E denote the vector notation of claim *c* and evidence *e* based upon their word frequency. Consider the cosine similarity between claim and evidence *i* to be S_i. Using above variables we form a feature vector for each claim and evidence pair for the classification model *i* as:

$$\textit{feature vector} =< s_i, r_i, u_i, cs_i, cr_i, cu_i, S_i >$$

The above *feature vector* give us an understanding of how closely two statements are related i.e., a relationship between claim and evidence. Some statements which are a negation to each other may have high cosine similarity but then their contradiction probability would be high which would help the learning algorithm to classify claims accurately. We used both the datasets i.e., the SICK dataset and the NITA dataset, along with the above mentioned feature vector for training and testing purpose of various machine learning classification models like Naïve Bayes, Logistic Regression, Support Vector Machine, Random Forest and Multi-Level Perceptron. These models were used as they are widely used in the industry for practical applications.

4 Experiments

As mentioned in section 3, our model of fact verification consists of two stages:

1. *Retrieving evidence related to the claim from the source text.*

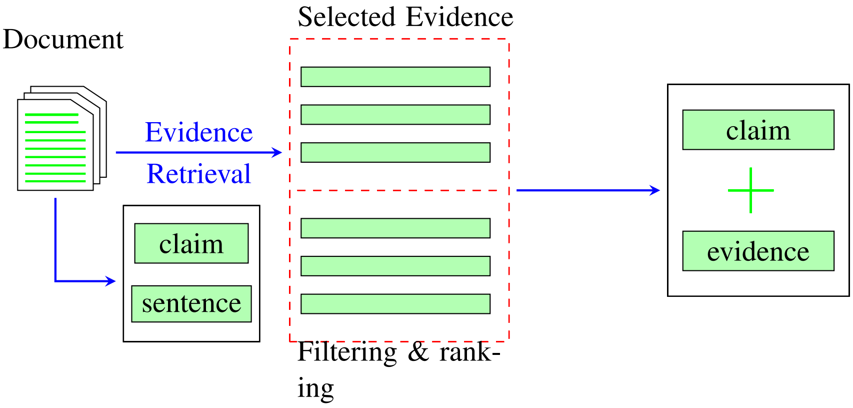


Figure 1: System Overview: Document Retrieval, Sentence Selection, and Claim Verification.

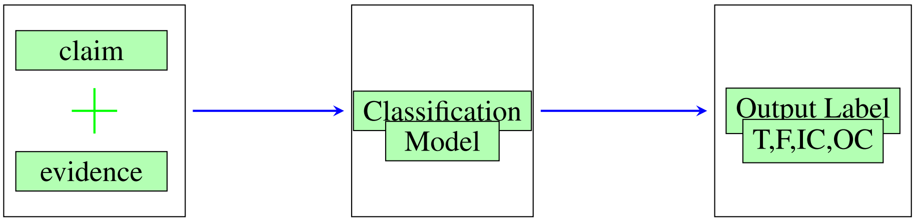


Figure 2: System Overview: Claim Classification Process.

Relatedness		**Entailment**	
[1-2) range	923 (10%)	NEUTRAL	5595 (57%)
[2-3) range	1373 (14%)	CONTRADICTION	1424 (14%)
[3-4) range	3872 (39%)	ENTAILMENT	2821 (29%)
[4-5) range	3672 (37%)		

Table 4: Distribution of SICK sentence pairs for each gold relatedness level and entailment label.

2. *Classifying a claim to be True, False, Inappropriate and Out-of-Context with respect to the source text.*

4.1 Evidence Retrieval

We used the uni-gram TF-IDF vector of sentences of the source text and the claim and computed the cosine similarity between various sentences from source text and claim. Based upon their cosine similarities, we selected a concatenation of top four sentences as evidence for the claim from the source text. After the evidence is retrieved, we are now left with a claim and an evidence upon which classification of claim is to be carried out in the next stage. In NITA dataset, we have a column describing the name of source text from which we wish to derive evidence from the claim.

4.2 Classification of Claims

In this stage we classify the claims using textual entailment module. For accomplishing the textual entailment task, we used the decomposable attention model developed by Parikh et al. (2016). This model was trained and tested upon the Stanford Natural Language Inference (SNLI) Corpus and has a test accuracy of 86.8% . We used this model to obtain probabilities of entailment, contradiction, and neutrality between a claim and evidence and further developed more variables as discussed in Section 3. The feature vector along with modified SICK data was passed into various classification models. We used the same approach for the NITA dataset. The test and train dataset split for both the above experiments was 60% training, 20% cross validation and 20% testing. The results of experiments on both the datasets are as in Table 5 which consists of the weighted average of all

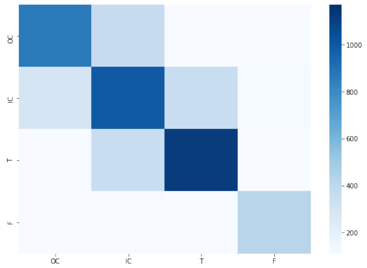


Figure 3: Random Forest Confusion Matrix for SICK Dataset

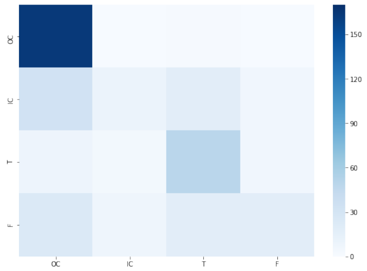


Figure 4: Random Forest Confusion Matrix for NITA Dataset

the results.

SICK dataset consists of pair of sentences and their relationship, hence evidence retrieval part for this dataset is skipped. The highest accuracy model observed with both the datasets is the "Random Forests" model with 100 trees and a maximum depth of 5.

5 Error Analysis

On observing the results it was found that the false claims were the most error-prone and have the least correct results. The SICK dataset gave better results in comparison to the NITA dataset prepared by us which is depicted by the random forest confusion matrix in Figure 3 and 4. The reason behind this could be that the evidence retrieved in the evidence retrieval part was not appropriate consisting of punctuation symbols and the other unwanted context in its text. On observing the results it was found that the false claims were the most error-prone and have the least correct results. The possible reasons for this could be the low probability rate of contradiction returned by the textual entailment module and also high cosine similarity, because in some instances the claim and evidence pair can be a negation of each other and hence have high word similarity. Next, upon observing results produced by the classification model, we saw that most inappropriate claims were classified as true and some true claims were classified as inappropriate. This ambiguity is mainly due to evidence supporting a claim partially, the probability of entailment for this would be high but due to variance in cosine similarity between claim and evidence there can arise an ambiguity. The overall system performance is at par with other existing fact verification systems as mentioned in section in terms of accuracy. Further modifications to improve the performance of the system are discussed in the next section .

6 Conclusion and Future Scope

The uniqueness in our approach is classifying a fact into four classes, this not only gives information about the fact whether it is true or false but also gives us an insight whether the source text we are using is limited. The *Out-of-Context* label particularly tries to validate the scope of our source text whether the source is enough to classify a particular fact/claim.

Here we discussed about the modification of the SICK dataset as per our requirement along with our approach of carrying out the process of classifying our claims. Compared to the existing fact verification systems such as FEVER systems which classifies a claim only into 3 classes, our model classifies a claim into 4 classes giving additional information. Our system can have many practical applications like subjective paper correction, fake news identifier, social media fact checking, etc. We believe that our system will provide a stimulating challenge for claim/fact extraction and verification systems and be effective for knowing about the scope of the source.

In future, we wish to tackle the problem of restricting our system for a particular source text, by enabling the system to extract evidence from a larger source like the internet itself by using various APIs provided by prominent search engines such as Google API to get appropriate evidence. Next thing we wish to implement further as a modification in our system is come up with a larger dataset comprising of our 4 class classification labels to train our system for better accuracy.

Data-set	Model	Precision	Recall	F1-score	Accuracy
SICK	Naive Bayes	0.53	0.53	0.50	0.535
	SVM	0.58	0.58	0.57	0.582
	Random Forest	0.63	0.63	0.63	0.630
	Logistic Regression	0.57	0.58	0.57	0.577
	MLP	0.63	0.62	0.63	0.624
NITA	Naive Bayes	0.61	0.63	0.59	0.631
	SVM	0.62	0.65	0.61	0.649
	Random Forest	0.61	0.65	0.61	0.649
	Logistic Regression	0.61	0.65	0.60	0.649
	MLP	0.61	0.64	0.61	0.644

Table 5: Classification result using various classification models

References

Tariq Alhindi, Savvas Petridis, and Smaranda Muresan. 2018. Where is your evidence: Improving fact-checking by justification modeling. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. pages 85–90.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1870–1879.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1657–1668.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1163–1168.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *International Conference on Learning Representations*.

Christopher Hidey and Mona Diab. 2018. Team SWEEPer: Joint sentence extraction and fact checking with pointer networks. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Brussels, Belgium, pages 150–155.

Djoerd Hiemstra. 2000. A probabilistic justification for using tfxidf term weighting in information retrieval. *International Journal on Digital Libraries* 3(2):131–139.

Nayeon Lee, Chien-Sheng Wu, and Pascale Fung. 2018. Improving large-scale fact-checking using decomposable attention models and lexical tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 1133–1138.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, Dublin, Ireland.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*. Association for Computational Linguistics, Copenhagen, Denmark, pages 41–45.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2018. Combining fact extraction and verification with neural semantic matching networks. *arXiv preprint arXiv:1811.07039* .

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2249–2255.

Aniketh Janardhan Reddy, Gil Rocha, and Diego Esteves. 2018. Defactonlp: Fact verification using entity recognition, tfidf vector comparison and decomposable attention. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*. Association for Computational Linguistics, pages 132–137.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264* .

Mark Sammons, Vinod Vydiswaran, and Dan Roth. 2012. Recognizing textual entailment. *Multilingual Natural Language Applications: From Theory to Practice* pages 209–258.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 3346–3359.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 809–819.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The Fact Extraction and VERification (FEVER) shared task. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018c. Proceedings of the first workshop on fact extraction and verification (fever). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. pages 18–22.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 422–426.

A Neural Network Component for Knowledge-Based Semantic Representations of Text

Alejandro Piad-Morffis[1], **Rafael Muñoz**[2], **Yudivián Almeida-Cruz**[1],
Yoan Gutiérrez[3], **Suilan Estevez-Velarde**[1], and **Andrés Montoyo**[2]

[1]School of Math and Computer Science, University of Havana, Cuba
{sestevez,yudy,apiad}@matcom.uh.cu
[2]Department of Languages and Computing Systems, University of Alicante, Spain
[3]U.I. for Computer Research (IUII), University of Alicante, Spain
{montoyo,ygutierrez,rafael}@dlsi.ua.es

Abstract

This paper presents Semantic Neural Networks (SNNs), a knowledge-aware component based on deep learning. SNNs can be trained to encode explicit semantic knowledge from an arbitrary knowledge base, and can subsequently be combined with other deep learning architectures. At prediction time, SNNs provide a semantic encoding extracted from the input data, which can be exploited by other neural network components to build extended representation models that can face alternative problems. The SNN architecture is defined in terms of the concepts and relations present in a knowledge base. Based on this architecture, a training procedure is developed. Finally, an experimental setup is presented to illustrate the behaviour and performance of a SNN for a specific NLP problem, in this case, opinion mining for the classification of movie reviews.

1 Introduction

In recent years, the increase in the volume of available data has provided both new techniques and new challenges for discovering relevant knowledge. The huge amount of information produced daily makes it impossible for humans to manually build organized representations of all this information. On the other hand, the surplus of information enables the design of statistical learning techniques which scale better with large data. Deep learning approaches have successfully obtained state-of-the-art results for many learning problems, from image recognition to natural language parsing. Instead of handcrafted representations designed by experts, deep learning automatically builds representations from raw data that are suitable for a given machine learning problem.

When deep learning is used, we can often identify structures inside a neural network, which can be explained as high-level concepts that are automatically discovered during the learning process (Bengio, 2012). For instance, in the image recognition domain, often internal neurons or groups of neurons can be identified to recognize common visual features such as textures or patterns or even specific objects (Le, 2013). Hence, the neural network is able to build high-level concepts from low-level input (i.e., pixels). This ability to discover higher-level features is one of the main strengths of deep learning and representational learning in general (Bengio, 2012). However, one of the main challenges of deep learning is interpreting the internal representations of a neural network in terms that can be understood by humans and mapped to clearly defined domain concepts (Montavon et al., 2017).

To address this challenge, our proposal is to“persuade” a neural network to build representations that can be interpreted in terms of a formal conceptualization (such as a knowledge base). This intuitive approach, as opposed to hand-crafting features, would still allow the neural network to learn the best representation of the input, while enabling a better interpretation and explanation of the whole learning process.

Our approach involves designing specific neural network components —Semantic Neural Networks (SNNs)— that are trained to learn how to map raw input to specific domain concepts that are extracted from a convenient knowledge base. These components could then be included in larger deep learning architectures, providing a semantic representation of the input that the rest of the network could learn to use for solving a specific problem.

In this work, SNNs have been designed to represent the relevant concepts of a knowledge base as part of the artificial network architecture. Hence, during the whole process, the structures that map to specific concepts and relations are clearly identifiable inside the network. This process attaches a semantic meaning to the network architecture, which is useful for debugging and understanding the learning dynamics. Furthermore, the SNN is trained to learn the specific instances in the knowledge base and their relations. This way, the SNN not only encodes abstract concepts, but also true facts about instances of those concepts. Preliminary source code is available online for the research community.[1]

When used as a component of a larger deep learning architecture, a SNN that is trained for a specific knowledge domain can be seen as a semantic representational component. Its input consists of a low-level representation of data, (for example, words or entities), and its output consists of an implicit representation of this data expressed in terms of the learned domain. This representation can be seen as a type of embedding that maps raw input to a semantic space defined by the concepts and relations of the learned knowledge base. The SNN is used as a representational layer in a larger neural network, for a natural language processing problem in which the semantic representation induced by the learned knowledge base is expected to be a good representation.

The paper is organized as follows. Section 2 presents a review of related works and relevant concepts in the domain of representational learning, with an emphasis on different semantic representations of natural language. In section 3 the architecture and training procedure for the SNN is formalized and explained. Section 4 presents a brief experimental setup designed to illustrate the behavior and performance of a SNN in a specific natural language problem. Finally, section 5 discusses the main contributions of this proposal, whereas section 6 presents the final conclusions of the research and highlights possible future lines of development.

2 Related Works

Building semantic representations of raw input data, specifically for natural language text, is a common task for many machine learning problems. In this section we present different approaches to the design of semantic representations for natural text.

Network-Based Approaches for semantic representations usually consist of defining some similarity metric based on the relations of terms in some knowledge base, interpreted as a graph. It is based in the assumption that words which are connected by short paths in a knowledge base should have similar semantics. WordNet is commonly used as a knowledge base where different semantic relations among words can be exploited for defining similarity metrics. Using WordNet (Miller, 1995), several semantic similarity metrics are defined by exploring the graph structure of the knowledge base, mostly depending on the graph structure of words, such as *Hirst-St-Onge* (Hirst and St-Onge, 1998) and *Leacock-Chodorow* (Leacock and Chodorow, 1998). In this direction, other researchers include information content formulae to measure appearances of terms in a corpus, such as the *Resnik* metric (Resnik, 1999).

Corpus-Based Similarity Metrics are defined by some measure of the co-occurrence of terms in a corpus of natural text. The intuitive idea is that words which co-occur within the same context must have similar semantics. One such metric is PMI-IR (*point mutual information - information retrieval*) (Turney, 2001), which considers the information content of each pair of words (w_i, w_j), measured as the relative number of co-occurrences of w_i and w_j in a document, with respect to the individual count of occurrences of each word. Another corpus-based similarity metric based is ESA (*explicit semantic analysis*) (Gabrilovich and Markovitch, 2007), which considers Wikipedia as a corpus for building a co-occurrence matrix of words. A similar approach is HAL (*hyperspace analogue to language*) (Lund and Burgess, 1996), which also builds a co-occurrence matrix, but only considering words within a small window.

Dimensionality Reduction Techniques such as PCA (*principal component analysis*) (Martinsson et al., 2011) can be interpreted as a projection from the BOW (bag of words) or TF-IDF (term frequency - inverse document frequency) space to a semantic space. An interesting recent approach, that mixes ideas from the previous tech-

[1] `https://github.com/knowledge-learning/snn`

niques, is the family of word embedding algorithms (Mikolov et al., 2013). In word embeddings, similar to PCA, each word is mapped to a vector which encodes the semantics of the word. The embedding is chosen such that a word's vector contains an implicit representation of the probabilistic distribution of the word's context in a given corpus. To achieve this, a neural network is trained to predict, given a word w_i's embedding, the probability that some other word w_j appears in a small window centered around w_i. In this sense, word embeddings can be seen as a generalization of corpus-based metrics, whereby the best representation is learned from the data, rather than handcrafted. Even though word embeddings don't explicitly model specific semantic relations (such as hypernymy, or synonymy), it has been shown that several interesting semantic relations get encoded in specific directions in the embedding space, enabling the solution of analogue inference queries (Schnabel et al., 2015).

Entity Embeddings are a specific type of embedding technique that encodes the context of entities in a knowledge base. Several metrics can be used to define the notion of "context similarity" when using a knowledge base for entity embedding. For example, embeddings can be designed such that a particular direction d_r is associated with each particular relation of the knowledge base, such that $e_i + d_r \approx e_j$ whenever e_i and e_j are related by r. These formulations allow a semantic meaning to be attached to a particular algebraic operation and properties, and enable a whole new field of study that finds the "meaning" of, say, other directions d which are orthogonal to or linearly dependent on a specific relation. Entity embeddings have been extended to encode also the hierarchical structure of knowledge bases (Hu et al., 2015) and mixed with word embeddings for tasks such as entity disambiguation (Yamada et al., 2016).

Word and entity embeddings in general are promising approaches to deal with learning semantic representations of data. Moreover, recent research deals with finding ways to exploit the structure of these representations to explain why a specific answer is output by a neural network. Being able to explain neural networks is a first step towards designing accountable machine learning systems that humans can trust for solving the most crucial problems (e.g.medical diagnosis or legal advice). By carefully designing the learning criteria and structure of embeddings, it is conceivable that a semantic representation can be interpreted in terms of a formal conceptualization defined *a priori*.

3 Semantic Neural Networks

We define a Semantic Neural Network (SNN) as an artificial neural network architecture that encodes knowledge. Two main semantic elements are encoded from the Knowledge Base. First, the graph structure of the Knowledge Base (i.e. entity classes and their relations) is directly represented in the architecture of a SNN. Second from a Knowledge Base KB, the information about which instances belong to which entity classes and their specific relations are encoded into the weights of the SNN. By design, the architecture of the SNN is built to represent each specific entity class and relation in a clearly recognizable structure (a set of neurons with a pre-designed connection pattern). This allows a semantic meaning to be attached to an activation of the SNN in terms of the classes and relations defined in the Knowledge Base.

The purpose of the SNN is to provide a semantic representation of the input data that can be used as component inside a larger deep learning architecture, to solve a related learning problem L. If the knowledge represented in KB is useful for solving the problem L, then the representation provided by the SNN should be richer than plain bag-of-words or general purpose embeddings. With the same computational power (same number of parameters), a deep learning architecture using a SNN is expected to achieve equal or better performance than using other representations not specifically designed to exploit the knowledge in KB. Furthermore, the SNN architecture provides a semantic explanation for the model's predictions.

A SNN is built based on a specific Knowledge Base KB which is of interest for solving a related learning problem L. Figure 1 shows a schematic representation of the process for constructing, training and using an SNN in an arbitrary learning problem L. First, given the problem L to solve, a relevant Knowledge Base KB is chosen. The entity classes and relations of KB are used to define the architecture of the SNN (Section 3.1). Then, training instances are

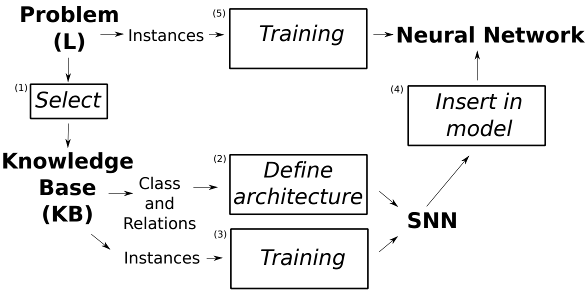


Figure 1: The process for constructing and using a SNN for a specific learning problem (L) and a suitable Knowledge Base (KB). The SNN is first defined and pre-trained based on KB, and then used in a larger neural network trained specifically for L.

extracted from KB and used to pre-train the SNN weights (Section 3.2.1). Afterwards, the SNN is included in a deep learning model (which can contain other components such as extra layers). Finally, this larger model is trained on instances from L using standard optimization techniques and loss functions suitable for the problem L (Section 3.2.2).

Different Semantic Neural Networks can be trained on different Knowledge Bases ahead of time and reused for many related problems. In this sense, SNNs are similar to pre-trained word embeddings, since a SNN trained for a commonly used Knowledge Base (i.e., DBPedia or WordNet) could be useful in solving different problems. However, pre-trained SNNs can (and should) be fine-tuned on a specific problem L after deciding a convenient deep learning architecture for this problem.

3.1 Architecture of the Semantic Neural Network

The architecture of a Semantic Neural Network (SNN) is composed of several instances of two simple structures: **entity blocks** and **relation blocks**. For each class of the knowledge base, an entity block is created, and for each relation, a corresponding relation block.

An **entity block** is a computational graph with a single input variable and a single output variable. The input dimension and shape is determined by the specific application, a sensible default consists of a single one-hot encoding layer when using a bag-of-words representation, but an alternative input could be a general-purpose word embedding representation. The output dimension is a fixed size dense vector of small dimension (e.g., 10). The input and output are connected by a linear matrix operator. Additionally, a one-dimensional indicator neuron is connected to the output layer through a dot product operator with a sigmoid activation function. The purpose of the indicator is to signal when the activation of the output is large in absolute value. This is interpreted as the importance of the corresponding concept in a particular input text.

A **relation block** is a similar computational graph, but with an input variable whose size is twice the entity output size. Thus, the relation input shape corresponds with the output shape of the two entity blocks that will be connected. The outputs from each incoming entity block are concatenated, forming a single vector, which is then connected through a linear matrix operator to the output variable. An identical indicator neuron is connected to the output variable.

The overall architecture of an SNN consists of several entity blocks, one for each class in the knowledge base, and several relation blocks, one for each relation defined. The entity blocks are all connected to a single input (e.g., a bag of words representation). Every relation block is connected to the respective outputs of the entity blocks that represent the classes for which the relation is defined. Figure 2 shows a schematic representation of an example SNN built from a knowledge base in the cinematic domain (specifically the Internet Movie Database, IMDB).

The input size of the SNN is the size of some vocabulary chosen before hand. This vocabulary should include the common terms in the knowledge domain(s) of interest. An additional input dimension can be added to account for unknown words (those not present in the vocabulary at the moment of training).

3.2 Training the Semantic Neural Network

The training procedure for a SNN is divided in two phases, each of which solves a different learning problem. In the first phase, which we call “structured pre-training”, the parameters of the entity and relation blocks are adjusted. The learning objective for this phase consists of predicting to which instance of the knowledge base the sample of natural text refers. In the second phase, in order to deal with the original natural language process-

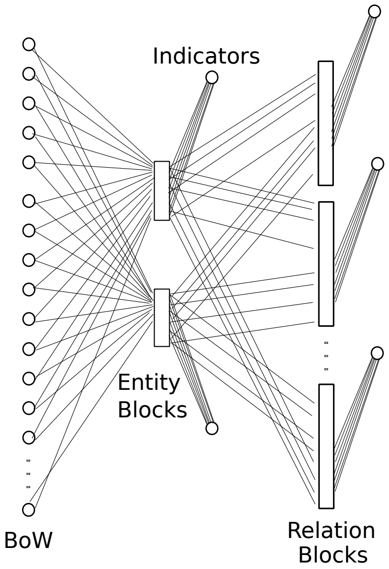


Figure 2: Schematic representation of a Semantic Neural Network, for an example knowledge domain in the film industry. The input layer consists of a bag of words (BOW) representation. The middle layer contains all the entity blocks (Movie and Person in this example) and the final layer contains all the relation blocks. The indicator outputs are one-dimensional sigmoid neurons.

ing problem, a standard training is performed. In this phase, the parameters of the rest of the network are adjusted. The internal parameters of the SNN can either be frozen, or optionally, adjusted together with the rest of the network in a post-optimization phase.

3.2.1 Structured Pre-Training

During the structured pre-training step, a knowledge base is used to extract random instances and adjust the SNN parameters. This knowledge base is the same as that chosen for designing the architecture, i.e., to choose the entities and relations which are represented in the SNN blocks. A random instance of this knowledge base is a tuple (e_i, r, e_j) which asserts the relation r between entities e_i and e_j. In this instance, e_i and e_j are assumed to have a natural text label associated (e.g., a name, description, tag, etc.).

The SNN learns to represent natural text in a manner which resembles the entity extraction process. Note that no natural text is associated to the label of the relation. Hence, in order for the SNN to accurately predict that relation $r(e_1, e_2)$ is true for a specific pair of entities e_1, e_2, but false for another pair of entities, the "list" of pairs of entities that hold in each relation must be encoded in the W_r weights of the corresponding relation block. Furthermore, in order to differentiate the relations, the SNN needs to learn to differentiate the entities implicitly because there is no requisite to define the specific learning objective, namely representing different entities and their corresponding encodings.

3.2.2 Unstructured Training

After the pre-training phase is completed, standard training proceeds. In this phase, given the characteristics of the learning problem, the training is performed. For example, if the problem consists of text classification (i.e., opinion mining, topic detection, etc.), then the training examples consist of pairs of single-hot encoded text and the corresponding class label. The exact parameters of the training depend on the problem characteristics, and are thus left unspecified in this proposal. During this training phase, the parameters of the SNN are frozen, i.e., they remain unchanged throughout the training procedure. The reason for this is that the parameters of the SNN have already been adjusted, and the rest of the parameters are randomly initialized. Therefore, allowing the training algorithm to change the SNN parameters would destroy the learned mappings.

After the standard training, a final phase of finetunning can be optionally performed. In this final phase, all the parameters, both those of the SNN and those of the rest of the network, are allowed to change. In this phase, only small parameter updates are expected to happen, since the network should have converged in the previous phase.

4 Experimental Analysis

To analyze the behavior of the SNN, we selected a classic NLP problem and a suitable knowledge base. The selected problem is opinion mining in movie reviews, using the dataset from Pang and Lee (2004). The corpus contains 1000 positive and 1000 negative movie reviews written in English. For building the knowledge base, raw data from IMDB[2] was processed obtaining a graph representation with 2 classes (*Person* and *Movie*), 11 relations and a total 27,044,985 tuples. Table 1 summarizes the statistics of the knowledge base.

The SNN obtained from the IMDB knowledge base has the structure shown in figure 3. The input

[2] https://datasets.imdbws.com

Classes	Instances
Person	2 854 359
Movie	2 361 769
Relations	
actor	6 531 498
actress	4 561 176
archive_footage	160 467
archive_sound	1 643
cinematographer	1 016 444
composer	1 030 405
director	2 871 640
editor	946 097
producer	1 646 903
production_designer	221 315
self	4 739 853
writer	3 256 270

Table 1: Summary statistics of the knowledge base built from IMDB data.

size is 267,178 (number of unique words in a standard English dictionary), the output size of each entity block is 10, and of each relation block is 20. The total number of indicator outputs is 14 (total number of concepts present in the IMDB knowledge base). Hence, the total number of trainable parameters in the SNN is 5,350,873.

The pre-training phase is executed for 10 epochs, each one with 100 batches, and each batch comprising 100 training examples. This cycle was repeated three times, first only with entities, then only with relations, and finally with both entities and relations. Hence, a total of 300,000 training examples were used in the pre-training phase. The final validation accuracy was 0.976 for the entities cycle, 1.00 for the relations cycle and 0.987 for the combined cycle. Since there are far fewer different relation types than actual entities, convergence is expected earlier when only training with relations.

4.1 Evaluating in the Opinion Mining Problem

The performance of the pre-trained SNN was evaluated by including it as an internal component in a larger neural network. This neural network was applied to the original problem of classifying movie reviews. The architecture of this extended neural network consists of a single-hot encoding input which is connected to the SNN. The output of the SNN is connected to three sequential dense layers with ReLU activation. A final dense layer with sigmoid layer was used, since the movie reviews problem is a binary classification problem. This network had an additional 141,361 trainable parameters.

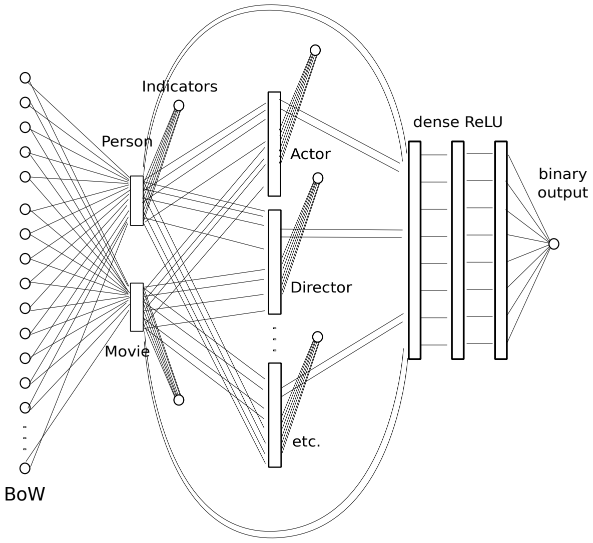


Figure 3: Schematic representation of the Semantic Neural Network trained in IMDB. Not all relation blocks are shown, given space constraints.

Training was performed only on these new parameters for 10 epochs, using 100 batches each one with 100 samples per epoch. After this process, the SNN parameters were unfrozen, and a fine-tune training was performed with all 5,491,981 parameters.

This final training was performed for 50 epochs, until convergence was achieved. The validation accuracy obtained at this point was 0.702. Finally, an independent test set was used to measure test accuracy, obtaining 67.82% precision in the movie reviews problem.

For comparison purposes, a fully-connected feed forward neural network, with 4 dense ReLU layers and a total of 5,494,681 trainable parameters was also trained from scratch in the movie review corpus. This network achieved a test accuracy of 64.47% in the same test set used for testing the SNN. Hence, with roughly the same number of trainable parameters the SNN obtains a small, but statistically significant advantage ($p \approx 10^{-39}$). Results are summarized in Table 2.

5 Discussion

The experimental results obtained show that using a SNN provides some benefits over a stan-

Model	Parameters	Accuracy
SNN + 3 ReLU	5,491,981	**67.82**
4 Dense ReLU	5,494,681	64.47

Table 2: Comparison of performance in the opinion mining problem between different SNN-based architectures.

dard architecture. No comparison has been performed with more advanced architectures, such as Long Short Term Memory (LSTM) or Convolutional (CNN) networks. However, the purpose of SNNs is not to compete with, but rather to complement existing architectures with a new type of component that is knowledge-aware. Hence, the fact that the SNN performs effectively when combined with more traditional architectures encourages its use with other state-of-the-art deep learning components.

The most significant advantage of using a SNN is that it provides a semantic interpretation of the network's behavior. The one-dimensional indicators used for guiding the SNN training could potentially act as a signaller for the high-level concepts that are being activated in any given example. By looking at the activation patterns inside the SNN, it may be possible to obtain an explanation of the network's prediction for a given example, in terms of the concepts in the knowledge base learned. Figure 4 presents an illustrative example in the context of the NLP problem presented in Section 4.

Another advantage of SNNs is that once pretrained with a given knowledge base, they can be reused in many different architectures. In a sense, SNNs can be seen as feature extractors or representational components that are associated with a given knowledge domain. It is also conceivable to use multiple SNNs trained in different knowledge domains in the same neural network. This provides a strategy for knowledge integration, when different and possible incompatible knowledge domains are considered relevant for a particular problem. This opens the door to new strategies for transfer learning, from a more semantic perspective, where the transferred or reused knowledge can be tied to a particular domain.

In our experimental setup we used a small natural language dataset, since our purpose was not to obtain state-of-the-art results in the opinion mining problem, but rather to illustrate the design of the SNN and explain its main architectural characteristics. We demonstrated that the SNN model provides effective support to learning approaches for resolving NLP problems.

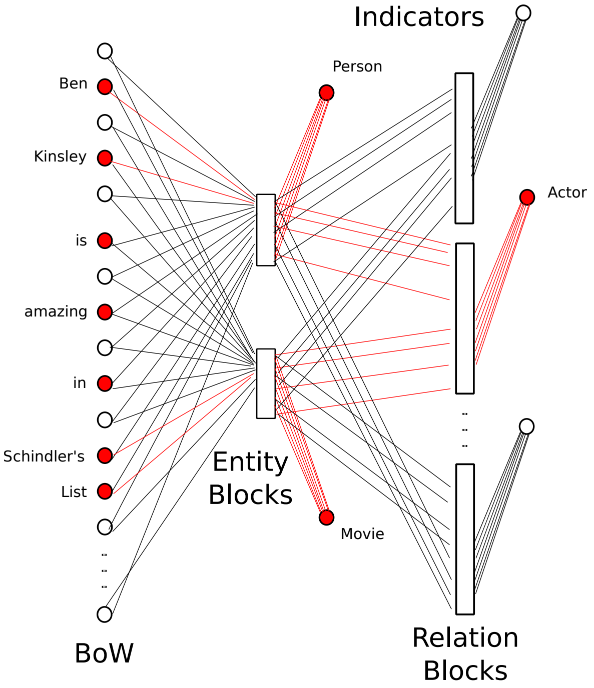


Figure 4: Illustrative example of a Semantic Neural Network activation in response to a particular input.

6 Conclusions and Future Work

In this paper, we present Semantic Neural Networks (SNNs), which allow the inclusion of explicit semantic knowledge in traditional neural networks without affecting performance. This knowledge is extracted from available knowledge bases, built by domain experts. We considered that the SNN is a first step towards raising the abstraction level in neural networks and building semantically-aware architectures that can be self-explained. A possible line of future research consists of combining multiple SNNs for multiple knowledge domains in a single problem, and studying how the different representations output by each network can be merged. Further, in the current design, the core of the SNN is a simple linear operation. In future works, we will explore other, more complex operations, which will potentially improve accuracy. It is also necessary to compare the performance of SNN-powered architectures with other state-of-the-art deep learning architectures, such as LSTMs, CNNs and different embedding strategies. Finally, the SNN internal structure has a semantic meaning attached to each neuron block. This opens the door for the design of interpretation models that can automatically output an explanation of a neural networks response in terms of human-defined concepts.

Acknowledgments

This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, it has also been partially funded by both aforementioned universities, the Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVING-LANG (RTI2018-094653-B-C22) and INTEGER (RTI2018-094649-B-I00).

References

Yoshua Bengio. 2012. Deep Learning of Representations for Unsupervised and Transfer Learning. In *JMLR: Workshop and Conference Proceedings*. volume 27, page 17–37.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI International Joint Conference on Artificial Intelligence*. https://doi.org/10.1145/2063576.2063865.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database* 305(April):305–332. https://doi.org/citeulike-article-id:4893262.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity hierarchy embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Stroudsburg, PA, USA, volume 1, pages 1292–1300. https://doi.org/10.3115/v1/P15-1125.

Quoc V Le. 2013. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 8595–8598.

Claudia Leacock and Martin Chodorow. 1998. Combining Local Context and WordNet Similarity for Word Sense Identification. *In: WordNet: An electronic lexical database.* (January 1998):265–283. https://doi.org/citeulike-article-id:1259480.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers* https://doi.org/10.3354/ame01683.

Per Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* https://doi.org/10.1016/j.acha.2010.02.003.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)* pages 1–12. https://doi.org/10.1162/153244303322533223.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* .

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research* 11:95–130. https://doi.org/10.1613/jair.514.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 298–307.

Peter D Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *Lecture Notes in Computer Science* https://doi.org/10.1007/3-540-44795-4_42.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation http://arxiv.org/abs/1601.01343.

Toponym Detection in the Bio-Medical Domain: A Hybrid Approach with Deep Learning

Alistair Plum, Tharindu Ranasinghe, Constantin Orăsan
Research Group in Computational Linguistics, University of Wolverhampton, UK
{a.j.plum, t.d.ranasinghehettiarachchige, c.orasan}@wlv.ac.uk

Abstract

This paper compares how different machine learning classifiers can be used together with simple string matching and named entity recognition to detect locations in texts. We compare five different state-of-the-art machine learning classifiers in order to predict whether a sentence contains a location or not. Following this classification task, we use a string matching algorithm with a gazetteer to identify the exact index of a toponym within the sentence. We evaluate different approaches in terms of machine learning classifiers, text pre-processing and location extraction on the SemEval-2019 Task 12 dataset, compiled for toponym resolution in the bio-medical domain. Finally, we compare the results with our system that was previously submitted to the SemEval-2019 task evaluation.

1 Introduction

The task of toponym resolution (TR) is a topic in natural language processing (NLP) which is aimed at extracting locations from texts. TR includes the sub-tasks of detecting, disambiguating and finally resolving and assigning coordinates to the proper location in the text. The first step is to detect a location in a text, which can be carried out by more simple means of gazetteer lookup or named entity recognition (NER) (Piskorski and Yangarber, 2013). Next, each detected location that is ambiguous needs to be distinguished and the correct location chosen (Leidner et al., 2004). The final step involves assigning coordinates or applying some other meta-information to clearly distinguish each proposed location (Smith and Crane, 2001; Leidner et al., 2004).

TR can involve many challenges, caused by cases that are not trivial to resolve. This includes, for example, locations contained in phrases such as *London Bus Company*, which contains *London* but refers to an organisation and should therefore not be marked as a location. Furthermore, locations can be contained in other types of phrases, such as genome sequences mentioned in bio-medical texts, as seen in the SemEval-2019 data (Weissenbacher et al., 2019). For example, the location *Henan* in the string *A/Tree sparrow/Henan/4/04* could be omitted from the results, as it is part of the name of a genome. Similar problems can occur where a location might be detected in a character sequence denoting a chemical.

The increasing availability of online resources has been beneficial to TR. On the one hand large-scale geographical databases, such as GeoNames[1], make information about many different locations easily and freely available. On the other hand, readily available mapping services, such as Google Maps[2], allow users to visualise these locations. Although some of these services, such as GeoNames, have been used since the beginnings of TR (Smith and Crane, 2001; Leidner et al., 2004), they are more complete today. A lack of training and evaluation data also existed for some time, mainly reflected by the fact that no standard corpora existed up until a certain period (Leidner, 2004) and that advanced learning techniques could not be used (Smith and Crane, 2001).

Geographic information contained in texts is highly useful and, therefore, the areas where TR is applied are diverse. In the past, TR has been used to catalogue digital libraries (Larson, 1996; Hill et al., 1999) and to make information retrieval techniques spatially aware (Clough,

[1] http://www.geonames.org/

[2] http://www.google.com/maps

2005). A more specialised area included the automatic tracking of biological specimens from different places around the world (Beaman and Conn, 2003). Today, the possible uses range from analysing social media texts (Ireson and Ciravegna, 2010) to news streams (Lieberman and Samet, 2012), where locations of large-scale activity and problematic regions are mapped, respectively. A further example is the bio-medical domain, where the spreading of viruses can be tracked by analysing texts that mention locations (Weissenbacher et al., 2019). However, as the areas of application are varied, the effectiveness of toponym resolvers is also said to vary among different types of text (Gritta et al., 2018).

This paper presents further research into TR, or to be more precise the detection of toponyms. Moreover, this research was carried out in the context of SemEval-2019 Task 12: Toponym Resolution (Weissenbacher et al., 2019) and aimed specifically at subtask 2, which deals only with the detection of toponyms. To the best of our knowledge, prior work on exploring how a machine learning classifier can be used together with relatively simple string matching to detect locations in texts has been limited. Previously, we explored the use of machine learning classifiers to predict a location within short word windows in the context of SemEval-2019 Task 12 (Plum et al., 2019).We employ our system submitted to the SemEval-2019 task as a baseline and make use of the same dataset, consisting of texts from the bio-medical domain Plum et al. (2019). While the system serving as a baseline was reasonably competitive in terms of precision, it did not achieve a high recall. The neural network architectures that are used for this research are novel to this type of task even though some of the architectures have been used for sentence classification tasks like sentiment analysis, spam detection and so on.

The rest of the paper is structured as follows. We present related work in the field first (Section 2), followed by the methodology employed (Section 3). This section includes a description of the dataset, system and network architectures. Section 4 presents the overall results, which are split into classifier results at the sentence level, i.e. disregarding the indexes and therefore not comparable to the SemEval evaluation (Section 4.1) and the overall results at the word level, where explicit indexes are retrieved in order to be evaluated according as in SemEval (Section 4.2). Finally, in Section 5 we come to general conclusions about the project.

2 Related Work

Detecting and resolving locations or toponyms has undergone some changes in its approach. The topic was first dealt with more extensively towards the late 1990s and early 2000s (Larson, 1996; Hill et al., 1999; Smith and Crane, 2001; Leidner et al., 2004). These earliest approaches were aimed mainly at using geographical information for information retrieval, as well as catalogue searches in digital libraries. While the first of these approaches used named entity tagging, as well as specially constructed gazetteers to detect (Larson, 1996; Hill et al., 1999), others went beyond this and used a combination of methods to disambiguate. Smith and Crane (2001) used NE tagging and a gazetteer to detect locations, and disambiguated these using information gathered beforehand. This information, ”local” and ”document” context, is said to include co-occurring words and other locations mentioned throughout the text, respectively. The authors also use ”world knowledge” gathered from other sources, which mainly includes meta information such as coordinates, size, corresponding political entities and so on (Smith and Crane, 2001). Similarly, Leidner et al. (2004) used a combination of simple heuristics, linguistic cues, co-occurrence statistics and discourse information to detect locations and assign coordinates.

TR has shifted from the methods of earlier approaches and followed the trend of using machine learning techniques. Whereas in the past learning techniques lacked data (Smith and Crane, 2001), this is no longer the case. Approaches using machine learning with (indirect) supervision include Hu and Ge (2009) and Speriosu and Baldridge (2013). Hu and Ge (2009) make use of hierarchical structures ensuing from geographical relations, an approach said to perform in an accuracy range of 73.55 to 85.38 percent on an Australian news corpus. Speriosu and Baldridge (2013) on the other hand, present their text-driven approach, which uses context information to resolve toponyms. The classifiers themselves are trained mainly on semi-automatically generated data, obtained primarily from locations tagged in Wikipedia. While the aforementioned approach relies on the availability of gazetteers, a gazetteer-independent approach has been brought forward by DeLozier et al. (2015). This approach, which also relies on a machine learning classifier to disambiguate toponyms, solely uses NER techniques to detect locations, and is said to perform on a state-of-the-art level on the TR-CoNLL (Leidner, 2006) and Civil War corpora (DeLozier et al., 2015).

Most recently, Gritta et al. (2018) have presented a survey of the current state of TR, which they refer to as geoparsing. While the two terms are essentially synonymous, the authors use geoparsing (or geoparsers) to refer to fully fledged end-to-end systems. These include CLAVIN[3], the Edinburgh Parser (Tobin et al., 2010; Grover et al., 2010), Topocluster (DeLozier et al., 2015) and GeoTxt (Karimzadeh et al., 2019). These systems are said to perform at state-of-the-art levels, and are tested on the Local Global Corpus, as well as a corpus compiled by the authors, which is based on Wikipedia and GeoNames data (Gritta et al., 2018). However, it should be mentioned that Topocluster and CLAVIN apply learning techniques. The Edinburgh Parser and GeoTxt rely on NER and heuristics to rank possible candidates.

The evaluation of toponym resolvers is carried out on specifically created datasets or corpora. Leidner (2004) was the first to raise awareness for the need of a gold standard for these purposes. To this end, the paper describes the ongoing effort to create such a corpus, including a custom markup language (TRML) and editor (TAME) (Leidner, 2004). Later, Leidner (2006) describes the resulting corpus of the previous efforts. It is based on news articles, with $6,980$ human-annotated instances of toponyms. The corpus, utilising the CoNLL format, is still used today (DeLozier et al., 2015; Gritta et al., 2018). However, recently concerns have been raised again concerning the availability of datasets for toponym resolution or geoparsing by both Gritta et al. (2018) and Karimzadeh and MacEachren (2019). Gritta et al. (2018) have contributed their own dataset compiled from Wikipedia. In addition, Karimzadeh and MacEachren (2019) present their tool GeoAnnotator which has been developed to aid the compilation of such corpora. The tool is said to not only be useful for the creation of large-scale corpora on a collaborative basis, but also versatile enough to be used for other applications of NLP.

Our System described in Plum et al. (2019) that was submitted to SemEval-2019 Task 12 will serve as the baseline. The approach uses GATE with ANNIE to detect all the occurrences of locations in a text, using custom gazetteers based on GeoNames. Several gazetteers were tested and the best results achieved with a gazetteer of locations with a population of $15,000$ people or more. Following the string matching, two neural network models are used to classify five-word windows around the matched location. For each window a prediction is made whether a real location is contained or not. The method is reported to have a significant drawback, since a five-word context is not enough to carry out proper classification, as the location itself could, for instance, be a multi-word expression. Furthermore, the gazetteer matching carried out beforehand severely limits the overall recall of the approach, as it is not able to detect locations that are written across line-breaks, or are simply not contained in the gazetteer. In contrast to this system, the approach proposed in this paper predicts locations on a sentence-by-sentence basis, then attempts to retrieve the correct index of each location by using a gazetteer lookup.

3 System Description

This section describes the system we developed for detecting toponyms in bio-medical texts. Our approach is based on our system submitted to SemEval-2019 Task 12, described in Plum et al. (2019). It differs mainly in the order of the processing stages, as well as in the architectures that were used. The previous system matches location names using a gazetteer, followed by a machine learning classifier to predict whether the matched location is a proper location or not (Plum et al., 2019). In the present approach, we use a machine learning classifier to predict whether a sentence contains a relevant location first, and on this preselection we perform a gazetteer lookup to identify the specific index range of each location. We also use spaCy NER[4] to compare the effectiveness of the gazetteer.

The approach for the system is split into three steps, which are explained in the following three sections. The first step deals with the preparation of the texts from the dataset. This involves cleaning noisy sections of text and outputting an input

[3] https://clavin.bericotechnologies.com/about-clavin/.

[4] Version 2.1.3, available at https://spacy.io/

file for the machine learning classifier containing all texts split into sentences. Next, each classifier is trained and run in order to obtain predictions at the sentence level. Finally, the output from the classifier is used in conjunction with a gazetteer lookup algorithm (and later spaCy NER) in order to determine the exact indexes of the detected locations.

3.1 Dataset

To ensure comparability with the baseline system, we work with the same dataset from SemEval-2019 Task 12. This dataset is made up of 150 journal articles from PubMed Central and are from the domain of epidemiology (Weissenbacher et al., 2019). As mentioned previously, the main idea behind detecting locations in texts from this domain is to track the spreading of viruses (Weissenbacher et al., 2019). The articles were downloaded as PDFs and converted to plain text using a pdf-to-txt tool by the organisers of the task. The toponyms were manually disambiguated by the organisers and subsequently annotated using the Brat annotator (Stenetorp et al., 2012). Two texts of the training set were removed, as they were unreadable, probably caused by PDF to text conversion problems. Apart from this, we work with the same training and test splits as supplied by the organisers, which are 73 and 45 texts, respectively. It should be pointed out that we had to adjust the annotations of some of the training texts, as these were carried out on texts using CRLF-type line breaks[5] and did not match the indexes read by our system, as it used LF-type line breaks, which lead to the indexes being offset.

The texts had to be prepared for the classification task. As some parts of the texts had been deemed irrelevant by the organisers of SemEval-2019 Task 12 (Weissenbacher et al., 2019), we had to remove these. This includes the references and acknowledgement sections of each article. We also had to remove certain character strings which are specific to texts from the bio-medical domain. Finally, the texts had to be split into sentences and stored with further information in order to be classified in the next step.

3.1.1 Text Cleaning

The cleaning of the texts is performed in two steps. First, all line breaks are removed and replaced with spaces. This is mainly to deal with sentence splitting and string matching problems that could occur over line breaks. For instance, a line break character between *New* and *York* would lead to this location to be detected as *York*, not as *New York*. The line breaks have to be replaced by one space, as the annotations take line breaks into account and add these to the index range of a location. It should be mentioned that this requires the texts to use LF-type line breaks, as any other type would require a different number of replacement characters. For this dataset, it was ensured that all files conform to this standard.

Next, we carry out more methods to clean the texts. Using the guidelines set out by the task organisers and a brief analysis carried out by Plum et al. (2019), we determined that certain parts of the texts are not relevant for detection. This includes references and certain character and word strings that describe biological genome sequences. As these often include toponyms that were excluded from annotation in the SemEval-2019 task, these could also be disregarded. In order to remove all irrelevant parts but retain indexing consistency, we use regular expressions to find and replace certain text sequences with an equivalent number of spaces. As before, we replace each character with a space in order to ensure that the indexes match up with the annotations.

We wanted to test the effectiveness of our cleaning methods. Therefore, we tested our methods with both types of text: texts where only the line breaks have been removed and texts that have been completely cleaned. During testing it was clear that the performance was better on fully cleaned texts, due to the reduction in seemingly random strings that could be detected (i.e. the string *[..] ACG GGG* ***MA*** *AUA UGC [..]* could produce the match *MA*, as in the U.S. state *Massachusetts*).

3.1.2 Sentence Splitting

As the identification of locations primarily happens at the sentence level, each individual text needs to be split into sentences. We use spaCy in order to complete this task. The sentences are then output to a CSV file, containing information on each sentence's text id, as well as its own specific index range. This information is necessary at the stage of identifying the exact index range of

[5] CRLF-type line breaks are commonly used in text files created with Microsoft DOS/Windows operating systems. This type of line break uses two characters to denote the end of a line. LF-type line breaks are commonly used on UNIX/Linux-based operating systems, and only use one character to denote the end of a line.

a location and then producing the annotation files for the SemEval evaluation script. For the machine learning training file, a separate column indicating whether or not a location is contained in the sentence is also included.

3.2 Sentence Level Location Identification

Once the texts have been pre-processed, we run a binary classification on the sentences, predicting whether a sentence contains a location, or not. Table 1 shows some examples from the training dataset which is used for this classification task.

Out of the $17,535$ sentences in the training set, only $2,117$ sentences contain locations. This means that the dataset is highly unbalanced, thus making the classification task quite difficult. The increased difficulty was also caused by the language used. As an example, in the row with the ID PMC2857219, shown in Table 1, the sentence contains *Korea* and is not annotated as a location since it is a part of an organisation name. Also, in the row with the ID PMC2857219, *US* is not annotated as a location, as it functions as an adjective to the word soldier. General named entity recognisers such as spaCy or gazetteer matching could possibly falsely detect that these sentences contain a proper location as defined for this task. Furthermore, sentences similar to ID PMC5837706, shown in 1, caused difficulties, as the text in the sentence was not clear. Considering all of these issues, we need an intelligent classifier that detects whether a sentence contains a location or not, by considering the words that appear in the sentence.

We use five different recurrent network architectures to perform the binary classification task on the sentences: pooled Gated Recurrent Unit (GRU) (3.2.1), stacked Long Short-Term Memory (LSTM) with attention (3.2.2), LSTM and GRU with attention (3.2.3), 2D convolution with pooling (3.2.4) and GRU with capsule (3.2.5). Each classifier was run on prepared and cleaned text (as explained in Section 3.1.1). These models were successfully applied to a number of classification tasks such as GRU for sequence labeling Chung et al. (2014), LSTM for semantic similarity and word analogy Coates and Bollegala (2018), and GRU with capsule for toponym detection Plum et al. (2019). Their success in these tasks inspired us to try them for our problem.

3.2.1 Pooled GRU

This model takes pre-trained fasttext embeddings (Mikolov et al., 2018) as a matrix for the input which is comprised of the vertically stacked embedding vectors corresponding to the words present in the sentence. The matrix can be thought of as a sequence of embedded words. Each of these embedding vectors is fed to the bi-directional GRU (Chung et al., 2014) at their respective timestep. The final timestep output is fed into a max pooling layer and an average pooling layer in parallel (Scherer et al., 2010). Following this, the outputs of the two pooling layers are concatenated and connected to a dense layer (Huang et al., 2017) activated with a sigmoid function. Additionally, there is a spatial dropout (Tompson et al., 2015) between the embedding layer and the bi-directional GRU layer to avoid over-fitting.

The network was trained using adam optimiser (Kingma and Ba, 2015), with a reduced learning rate once learning stagnates. This model has been discussed in Kowsari et al. (2019) as a common model to perform text classification tasks.

3.2.2 Stacked LSTM with Attention

As with the previous model, this model takes pre-trained fasttext embeddings (Mikolov et al., 2018) as an input. Each of these embedding vectors are then fed into a bi-directional LSTM (Schuster and Paliwal, 1997). The output of this layer is again fed into a bi-directional LSTM (Schuster and Paliwal, 1997) with self attention (Vaswani et al., 2017). Finally, the output is connected to two dense layers that are (Huang et al., 2017) activated first with a relu function, and then with a sigmoid function.

Again, this network was trained using adam optimiser (Kingma and Ba, 2015), with a reduced learning rate once learning stagnates. We adopted this model from the Toxic Comment Classification Challenge in Kaggle[6].

3.2.3 LSTM and GRU with Attention

This architecture applies a spatial dropout to the embedding layer (Tompson et al., 2015). The output is then fed in parallel to a bi-directional LSTM-layer (Schuster and Paliwal, 1997) with self attention and a bidirectional GRU-layer (Chung et al., 2014) with self attention

[6] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

id	start	end	sentence	loc?	location
PMC2857219	15686	15753	Dr Jin-Won Song is a professor of microbiology at Korea University.	0	NA
PMC5005937	9817	9928	kConFab recruited multi-generational, multiple-case families through cancer family clinics in Australia.	1	Australia
PMC2857219	14913	14947	These data showed the epidemiologic link between US soldier patients and rodent hosts at the training sites near the Demilitarized Zone in South Korea.	1	South Korea
PMC5837706	14489	14531	(46.9) 395 (39.4) 75 (7.5) 245 (24.4) 329	0	NA

Table 1: Example rows in the training set. Sentences containing a location are represented with 1 in the *loc?* column and 0 otherwise. The *location* column contains the explicit location names contained in the sentence.

(Vaswani et al., 2017). The output from the bi-directional GRU-layer is fed into an average pooling layer and a max pooling layer. The output from these layers and the output of the bi-directional LSTM-layer are concatenated and connected to a dense layer with relu activation. After that, a dropout (Srivastava et al., 2014) is applied to the output and connected to a dense layer activated with a sigmoid function.

While this network was also trained using adam optimiser (Kingma and Ba, 2015), it was trained with a cyclical learning rate (Smith, 2017) this time. Plum et al. (2019) has used this model to predict whether a word window contains a location or not.

3.2.4 2D Convolution with Pooling

The fourth architecture takes a different approach than the previous architectures by using 2D convolution layers (Wu et al., 2018), rather than LSTM- or GRU-layers. The outputs of the embedding layers are connected to four 2D convolution layers (Wu et al., 2018), each with max pooling layers. The outputs of these are concatenated and connected to a dense layer activated with a sigmoid function after applying a dropout (Srivastava et al., 2014).

This network also uses adam optimiser (Kingma and Ba, 2015) and a reduced learning rate once learning stagnates. This model has been used in the Quora Insincere Questions Classification Kaggle competition[7].

3.2.5 GRU with Capsule

Most of the previous architectures rely on a pooling layer. However, this architecture uses a capsule layer (Hinton et al., 2018) rather than pooling layers. After applying a spatial dropout (Tompson et al., 2015) the output of the embedding layer is fed into a bi-directional GRU-layer (Chung et al., 2014). The output is then connected to a capsule layer (Hinton et al., 2018). The output of the capsule layer is flattened and connected to a dense layer with relu activation, a dropout (Srivastava et al., 2014) and batch normalisation applied, and re-connected to a dense layer with sigmoid activation.

The capsule network was trained using adam optimiser (Kingma and Ba, 2015), with a reduced learning rate once learning stagnates. This model has been used in Plum et al. (2019) to predict whether a word window contains a location or not.

3.3 Word Level Location Identification

The location predictions made by the ML-classifier at the sentence level are passed as a CSV file to the string matching script. It runs through each sentence, matching locations from a gazetteer. For matching the strings we use a fast and efficient Aho-Corasick algorithm (Aho and Corasick, 1975). The implementation used is available for the Python programming language[8]

We use a large gazetteer that is comprised of the full list of all locations from the GeoNames database[9]. The main idea behind this is that we want to achieve the highest chance of detecting

[7] https://www.kaggle.com/c/quora-insincere-questions-classification

[8] https://github.com/WojciechMula/pyahocorasick/

[9] http://www.geonames.org/, last accessed 21.05.2019

every location. However, as the gazetteer is so large, is causes a lot of ”noise” during the string matching, including partial matches and numbers that have no meaning. In order to combat this, we apply several filters after finding matches, in order to exclude certain results. We consciously avoided removing anything from the gazetteer itself, as this would be either time-consuming (manual) or could falsely remove desired locations (automatic). Therefore, we use filters to remove all matches with numbers (i.e. *717, 7Palms, 50da*), strings shorter than three characters which are not both uppercase (i.e. *Bl, b1, al* but not *AL, CA, NY*), sub-string matches (i.e. *London* in *Londonderry*) and all lowercase strings (which are usually fragments of location names left in the database, as in *paseo caribe*). The resulting tables are sorted by index, and duplicates removed. Where matches overlap in terms of indexing, we give preference to the longest match. This ensures that in sentences such as *I live in New York*, we detect only *New York* and not *York* (as these are separate entries in the gazetteer).

For comparison purposes, we also employ spaCy to detect locations in the sentences at this stage. We used the standard English web corpus and the spaCy NER algorithm.

4 Results

As our system operates at two levels, we first present the results of location prediction at the sentence level using the five different recurrent network architectures. Next, we present the results of the prediction at the word level. These results are also regarded as our final results, as these predictions yielded the index range of each location, which were evaluated with an evaluation script. The evaluation script that was utilised was the one provided for SemEval-2019 Task 12.

4.1 Sentence Level Prediction

Results of the sentence level predictions for the test set are shown in Table 2. We use precision and recall to evaluate the results. The third model described, LSTM and GRU with Attention, provided the best results for the cleaned text. Despite the dataset being quite unbalanced, the model reported good precision and recall scores of 0.852 and 0.853, which provided a high F1 score, too.

As this is the best model, we use these predictions as the basis for our word level predictions, which are described in the next section. It should be mentioned that we did run some word level predictions on the output of the other classifiers, but as expected the results were always much lower, due to the decreased starting point.

4.2 Toponym Identification

As mentioned previously, we regard the word level predictions as the overall result of the system. The evaluation was carried out in accordance with parameters set out for SemEval-2019 Task 12, featuring strict and overlap categories on macro and micro levels. For the strict measure, predicted locations are only considered as correct if the text span matches the gold standard exactly. For the overlap measure, predictions are considered to be correct if they share a common span of text with the gold annotations. The python script was made available on Bitbucket[10] by the SemEval-2019 Task 12 organizers.

Table 3 shows the results for both the string matching method using gazetteers to extract the locations with a custom script for indexes, as well as the spaCy NER algorithm (only considering locations) and the baseline system submitted to SemEval-2019. Our best results were achieved in the overlap macro classes, and are highlighted in bold. Overall, while we were not able to beat the best precision score of Plum et al. (2019), we came quite close. Nonetheless, we were able to improve the recall significantly, as well as the overall f-score.

The trade-offs that each approach brings with it should become clear when regarding the results.The approach using the GeoNames gazetteer detects a higher number of locations overall. This is due to the simplistic string matching method backed by such a large gazetteer, and comes at the cost of overall precision. The spaCy NER algorithm is much more precise, but is more limited in terms of recall. We find it most likely that this approach does not tag many locations as such, because the texts are still quite noisy, and because we did not train it on our dataset. Due to the small size and unbalanced nature of our dataset, we did not consider training spaCy any further. In the future, given an appropriate dataset from the bio-medical domain, this could perhaps lead to better results.

[10] https://bitbucket.org/dweissen/semevaltask12evaluator/src/master/

Model	Uncleaned			Cleaned		
	P	R	F1	P	R	F1
Pooled GRU	.755	.792	.772	.789	.816	.793
Stacked LSTM with Attention	.795	.784	.789	.826	.796	.813
LSTM and GRU with Attention	.811	.840	.825	**.852**	**.853**	**.852**
2D Convolution with Pooling	.802	.743	.769	.842	.758	.792
GRU with Capsule	.843	.816	.829	.865	.823	.842
All sentences predicted 1	.042	.500	.078	.060	.500	.107
All sentences predicted 0	.457	.500	.478	.439	.500	.468

Table 2: Results for the sentence level classification. We report Precision (P), Recall (R), and F1 for each model (bold indicates the best set of results). Two baseline predictions with all sentences predicted 1 and all sentences predicted 0 are also reported for comparison.

Approach - Level	Strict Detection			Overlap Detection		
	P	R	F1	P	R	F1
Gazetteer - Macro	.524	.791	.631	**.571**	**.840**	**.680**
Gazetteer - Micro	.508	.659	.574	.580	.731	.647
spaCy NER - Macro	.780	.573	.661	**.861**	**.627**	**.726**
spaCy NER - Micro	.726	.413	.526	.823	.468	.597
Baseline - Macro	.828	.474	.603	**.898**	**.496**	**.639**
Baseline - Micro	.816	.339	.479	.893	.365	.518

Table 3: Results for the word level classification. We report the same measures as previously, for the categories described in Section 4. Results are shown for both approaches (bold indicates the best set of results).

5 Conclusion

We have presented a system for toponym detection based on a recurrent network and gazetteer lookup. The approach is novel in that we use the recurrent network to predict whether a sentence contains a location, from which we later extract the exact location by more simpler means. This eliminates the need for a more extensive sequence labelling task, which would require more elaborately annotated training data. Our approach was able to improve on the baseline system presented at SemEval-2019 Task 12 on the same dataset. Albeit not at the same level of precision.

The main conclusion of this paper is that using a hybrid approach, where a machine learning approach is mixed with more traditional gazetteer lookup methods, seems to introduce too many areas where performance is lost. Both in this paper and in Plum et al. (2019) the machine learning or rather neural network architectures on their own perform quite well. However, performing a gazetteer lookup before or after lowers the result significantly. Of course, in both cases these lookups provide necessary information for the system to work, as they provide the index range of each location. The fact that our system performed better in terms of precision when using the popular spaCy NER algorithm, shows that simple gazetteer lookup lacks precision and is probably too simple.

In the future, we would like to introduce an end-to-end system entirely based on machine learning or recurrent network architectures. One emerging approach is to use BERT (Devlin et al., 2018) for token classification which is a fine-tuning model that wraps the BERT model and adds a token-level classifier on top of the BERT model. The recent release of BioBERT (Lee et al., 2019) makes it easier to apply BERT for NER in the bio-medical domain. While a lot more research has been focused on these kind of architectures, we hope to explore tasks other than only sequence labelling. To this end, our aim is to also explore the use of word and context embeddings further.

References

Alfred V. Aho and Margaret J. Corasick. 1975. Efficient string matching: An aid to bibliographic search. *Commun. ACM* 18(6):333–340.

Reed S. Beaman and Barry J. Conn. 2003. Automated geoparsing and georeferencing of Malesian collection locality data. *Telopea* 10(1):43–52.

Junyoung Chung, aglar Gülehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* .

Paul Clough. 2005. Extracting metadata for spatially-aware information retrieval on the internet. In *Proceedings of GIR'05*.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding - computing meta-embeddings by averaging source word embeddings. In *Proceedings of NAACL-HLT 2018*.

Grant DeLozier, Jason Baldridge, and Loretta London. 2015. Gazetteer-independent toponym resolution using geographic word profiles. In *Proceedings of AAAI 2015*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv* abs/1810.04805.

Milan Gritta, Mohammad Taher Pilehvar, Nut Limsopatham, and Nigel Collier. 2018. Whats missing in geographical parsing? *Language Resources and Evaluation* 52(2):603–623.

Claire Grover, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. Use of the Edinburgh geoparser for georeferencing digitized historical collections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368(1925):3875–3889.

Linda L Hill, James Frew, and Qi Zheng. 1999. Geographic names. *D-lib Magazine* 5(1):17.

Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *Proceedings of ICLR 2018*.

You-Heng Hu and Linlin Ge. 2009. *A Supervised Machine Learning Approach to Toponym Disambiguation*, Springer London, pages 117–128.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. *Proceedings of IEEE CVPR 2017* .

Neil Ireson and Fabio Ciravegna. 2010. Toponym Resolution in Social Media. In *Proceedings of ISWC 2010*.

Morteza Karimzadeh and Alan M. MacEachren. 2019. GeoAnnotator: A Collaborative Semi-Automatic Platform for Constructing Geo-Annotated Text Corpora. *ISPRS International Journal of Geo-Information* 8(4).

Morteza Karimzadeh, Scott Pezanowski, Alan M. MacEachren, and Jan O. Wallgrn. 2019. Geotxt: A scalable geoparsing system for unstructured text geolocation. *Transactions in GIS* 23(1):118–136.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *Proceedings of CoRR 2015* .

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text Classification Algorithms: A Survey. *Information* 10(4).

Ray R Larson. 1996. Geographic information retrieval and spatial browsing. *Geographic information systems and libraries: patrons, maps, and spatial information* .

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746* .

Jochen L Leidner. 2004. Towards a reference corpus for automatic toponym resolution evaluation. In *Proceedings of GIR'04*.

Jochen L. Leidner. 2006. An evaluation dataset for the toponym resolution task. *Computers, Environment and Urban Systems* 30(4):400 – 417. Geographic Information Retrieval (GIR).

Jochen L Leidner et al. 2004. Toponym resolution in text:Which Sheffield is it?. In *Proceedings of GIR'04*.

Michael D. Lieberman and Hanan Samet. 2012. Adaptive Context Features for Toponym Resolution in Streaming News. In *Proceedings of ACM SIGIR*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of LREC 2018*.

Jakub Piskorski and Roman Yangarber. 2013. Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization*, Springer, pages 23–49.

Alistair Plum, Tharindu Ranasinghe, Pablo Calleja, Constantin Orasan, and Ruslan Mitkov. 2019. RGCL-WLV at SemEval-2019 Task 12: Toponym Detection. In *Proceedings of SemEval-2019*.

Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of ICANN 2010*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing* 45:2673–2681.

David A. Smith and Gregory Crane. 2001. Disambiguating Geographic Names in a Historical Digital Library. In *Proceedings of ECDL 01*.

Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *Proceedings of IEEE WACV 2017*.

Michael Speriosu and Jason Baldridge. 2013. Text-driven toponym resolution using indirect supervision. In *Proceedings of ACL 2013*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of EACL 2012*.

Richard Tobin, Claire Grover, Kate Byrne, James Reid, and Jo Walsh. 2010. Evaluation of georeferencing. In *Proceedings of GIR'10*.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using Convolutional Networks. *Proceedings of IEEE CVPR 2015* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*.

Davy Weissenbacher, Arjun Magge, Karen O'Connor, Matthew Scotch, and Graciela Gonzalez. 2019. SemEval-2019 Task 12: Toponym resolution in scientific papers. In *Proceedings of SemEval-2019*.

Yunan Wu, Feng Yang, Ying Liu, Xuefan Zha, and Shaofeng Yuan. 2018. A Comparison of 1-D and 2-D Deep Convolutional Neural Networks in ECG Classification. In *Proceedings of IEEE Engineering in Medicine and Biology Society*.

Combining SMT and NMT Back-Translated Data for Efficient NMT

Alberto Poncelas, Maja Popović, Dimitar Shterionov,
Gideon Maillette de Buy Wenniger and **Andy Way**
School of Computing, DCU, ADAPT Centre
{firstname.lastname}@adaptcentre.ie

Abstract

Neural Machine Translation (NMT) models achieve their best performance when large sets of parallel data are used for training. Consequently, techniques for augmenting the training set have become popular recently. One of these methods is back-translation (Sennrich et al., 2016a), which consists on generating synthetic sentences by translating a set of monolingual, target-language sentences using a Machine Translation (MT) model.

Generally, NMT models are used for back-translation. In this work, we analyze the performance of models when the training data is extended with synthetic data using different MT approaches. In particular we investigate back-translated data generated not only by NMT but also by Statistical Machine Translation (SMT) models and combinations of both. The results reveal that the models achieve the best performances when the training set is augmented with back-translated data created by merging different MT approaches.

1 Introduction

Machine translation (MT) nowadays is heavily dependent on the quantity and quality of training data. The amount of available good-quality parallel data for the desired domain and/or language pair is often insufficient to reach the required translation performance. In such cases, it has become the norm to resort to back-translating freely available monolingual data as proposed in (Sennrich et al., 2016a). That is, one can translate a set of sentences from language L2 into L1 with an already trained MT system for the language pair L2→L1. Then create a synthetic parallel corpus from L1 to L2, with the source (L1) side being the translated text and the target side being the monolingual data. Back-translation has been shown to be beneficial not only for MT but also for other NLP tasks where data is scarce, e.g. automatic post-editing (APE) (Junczys-Dowmunt and Grundkiewicz, 2016; Negri et al., 2018). However, the effects of various parameters for creating back-translated (BT) data have not been investigated enough as to indicate what are the optimal conditions in not only creating but also employing such data to train high-quality neural machine translation (NMT) systems.

The work presented in Poncelas et al. (2018) draws an early-stage empirical roadmap to investigating the effects of BT data. In particular, it looks at how the amount of BT data impacts the performance of the final NMT system. In Sennrich et al. (2016a) and Poncelas et al. (2018), the systems used to generate the BT data are neural. However, it has been noted that often different paradigms can contribute differently to a given task. For example, it has been shown that applying an APE system based on NMT technology improves statistical machine translation (SMT) output, but has lower impact on NMT output (Bojar et al., 2017; Chatterjee et al., 2018).

In this work we assess the impact of different amounts of BT data generated by two different types of MT systems – NMT and SMT. Our contribution is two-fold: (i) we provide a systematic comparison of the BT data by building NMT systems with a combination of SMT and NMT BT data and (ii) we identify the effects of BT data that originates from SMT or NMT on the end-quality of the trained NMT system. We aim to answer the question: "What is the best choice for BT data?"

Proceedings of Recent Advances in Natural Language Processing, pages 922–931,
Varna, Bulgaria, Sep 2–4 2019.

2 Preparatory Study: the Effect of Back-Translation when Controlling for the Amount of Training Effort

A typical assumption made when training NMT models, is that when more training data is used, more training effort is warranted. Based on this assumption when training NMT systems what is normally kept constant is the amount of training epochs rather than the amount of training effort in the form of steps/mini-batches. Nevertheless, when adding back-translated data to the training set, while keeping the amount of epochs the same, the effective amount of training increases. It could then be questioned whether the extra training effort in itself does not partly explain the positive effect of back-translation. For this reason, we seek to answer the question: "Does the effect of back-translation change when we control for the amount of training effort, by keeping the total amount of steps/mini-batches constant?". To answer this question we compare the performance of systems trained on purely authentic data to those trained on authentic plus synthetic data, while keeping either the number of steps/mini-batches or the number of epochs constant in both settings:

1. Models trained with 1M auth + 2M synth sentences using the default settings, including 13 training epochs.
2. Models trained on 1M auth data only, trained either:
   (a) using the default settings, including 13 training epochs.
   (b) Trained for 39 epochs, to obtain a same amount of training effort as for the 1M auth + 2M synth sentences model.

When increasing the epochs to 39, we take appropriate measures to keep the starting point and speed of decay of the learning rate constant for the amount of training steps/epochs.[1]

The results of these experiments indicate that training a model on authentic data with $1/3$ of the amount of the total parallel data (authentic + synthetic) for an additional 26 epochs to account for the extra training effort is not required as no significant improvement has been observed. Based on the outcome of these experiments we chose the rest of our experiments.

3 Using Back-Translation from Different Sources

The work of (Sennrich et al., 2016a) showed that adding BT data is beneficial to achieve better translation performances. In this work we compare the details related to the translation hypotheses originating from SMT and NMT back-translated training data as well as combine the data from those two different sources. To the best of our knowledge, this has not been investigated yet.

We compare German-to-English translation hypotheses generated by systems trained (i) only on authentic data, (ii) only on synthetic data, and (iii) on authentic data enhanced with different types of BT data: SMT, NMT. We exploit two types of synthetic and authentic data combinations: (a) randomly selected half of target sentences back-translated by SMT and another half by NMT system, and (b) joining all BT data (thus repeating each target segment).

The translation hypotheses are compared in terms of four automatic evaluation metrics: BLEU (Papineni et al., 2002), TER (Snover et al., 2006), METEOR (Banerjee and Lavie, 2005) and CHRF (Popović, 2015). These metrics give an overall estimate of the quality of the translations with respect to the reference (human translation of the test set). In addition, the translation hypotheses are analyzed in terms of five error categories, lexical variety and syntactic variety.

4 Related Work

A comparison between MT models trained with synthetic and with authentic data that originate from the same source has been presented in Poncelas et al. (2018). They show that while the performances of models trained with both synthetic and authentic data are better than those of models trained with only authentic data, there is a saturation point beyond which the quality does not improve by adding more synthetic data. Nonetheless, models trained only with synthetic (BT) data perform very reasonably, with evaluation scores being close to those of models trained with only authentic parallel data. In fact, when appropriately selected, BT data can be used to enhance NMT

[1] This is implemented by changing the start of the learning rate decay from epoch 8 to epoch 22 $(= 7 * 3 + 1)$ and changing the decay factor from 0.5 to $\sqrt[3]{0.5} = 0.7936$. This way, the learning rate decay starts after the same amount of data when using the 1M auth dataset $(7 \times 3M)$ and the decay rate is maintained at 0.5 for each $3M$ sentences from this point onwards.

models (Poncelas et al., 2019).

Edunov et al. (2018) confirmed that synthetic data can sometimes match the performance of authentic data. In addition, a comprehensive analysis of different methods to generate synthetic source sentences was carried out. This analysis revealed that sampling from the model distribution or noising beam outputs out-performs pure beam search, which is typically used in NMT. Their analysis shows that synthetic data based on sampling and noised beam search provides a stronger training signal than synthetic data based on argmax inference.

One of the experiments reported in Burlot and Yvon (2018) is comparing performance between models trained with NMT and SMT BT data. The best Moses system (Koehn et al., 2007) is almost as good as the NMT system trained with the same (authentic) data, and much faster to train. Improvements obtained with the Moses system trained with a small training corpus are much smaller; this system even decreases the performance for the out-of-domain test. The authors also investigated some properties of BT data and found out that the back-translated sources are on average shorter than authentic ones, syntactically simpler than authentic ones, and contain smaller number of rare events. Furthermore, automatic word alignments tend to be more monotonic between artificial sources and authentic targets than between authentic sources and authentic targets.

Burlot and Yvon (2018) also compared training BT data with authentic data in terms of lexical and syntactic variety, segment length and alignment monotony, however they did not analyze the obtained translation hypotheses. In (Vanmassenhove et al., 2019) it is shown that MT systems trained on authentic and on backtranslated data lead to general loss of linguistic richness in their translation hypotheses.

5 Experimental Settings

For the experiments we have built German-to-English NMT models using the Pytorch port of OpenNMT (Klein et al., 2017). We use the default parameters: 2-layer LSTM with 500 hidden units. The models are trained for the same number of epochs. As the model trained with all authentic data converges after 13 epochs, we use that many iterations to train the models (we use the same amount of epochs). As optimizer we use stochastic gradient descent (SGD), in combination with learning rate decay, halving the learning rate starting from the 8th epoch.

In order to build the models, all data sets are tokenized and truecased and segmented with Byte-Pair Encoding (BPE) (Sennrich et al., 2016b) built on the joint vocabulary using 89500 merge operations. For testing the models we use the test set provided in the WMT 2015 News Translation Task (Bojar et al., 2015). As development set, we use 5K randomly sampled sentences from development sets provided in previous years of WMT.

6 Data

The parallel data used for the experiments has been obtained from WMT 2015 (Bojar et al., 2015). We build two parallel sets with these sentences: *base* (1M sentences) and *auth* (3M sentences). We use the target side of *auth* to create the following datasets:

- *SMTsynth*: Created by translating the target-side sentences of *auth*. The model used to generate the sentences is an SMT model trained with *base* set in the English to German direction. It has been built using the Moses toolkit with default settings, using GIZA++ for word alignment and tuned using MERT (Och, 2003)). The language model (of order 8) is built with the KenLM toolkit (Heafield, 2011) using the German side of *base*.

- *NMTsynth*: Created by translating the target-side sentences of *auth*. The model used to generate the sentences is an NMT model (with the same configuration as described in Section 5 but in the English to German direction) trained with the *base* set.

- *hybrNMTSMT*: Synthetic parallel corpus combining *NMTsynth* and *SMTsynth* sets. It has been built by maintaining the same target side of *auth*, and as source side we alternate between *NMTsynth* and *SMTsynth* each 500K sentences.

- *fullhybrNMTSMT*: Synthetic parallel corpus combining all segments from *NMTsynth* and *SMTsynth* sets (double size, each original target sentence repeated twice with both an NMT and SMT back-translation-generated translation).

7 Experiments

In our experiments, we build models on different portions of the datasets described in Section 6. First, we train an initial NMT model using the *base* data set. Then, in order to investigate how much the models benefit from using synthetic data generated by different approaches, we build models with increasing sizes of data (from the data sets described in Section 6).

The models explored are built with data that ranges from 1M sentences (built with only authentic data from *base* data set) to 4M sentences (consisting on 1M sentences from *base* and 3M sentences generated artificially with different models). We also include the models built with the *fullhybrNMTSMT* set. As this set contains duplicated target-side sentences, the largest model we build contains 7M sentences in total but only 4M distinct target-side sentences.

8 Results

8.1 Controlling the Amount of Training Effort

Table 1 shows the effect of controlling the amount of training effort when using back-translation. It can be observed that increasing the number of epochs from 13 to 39 when using just the 1M base training set does not increase the performance over using just 13 epochs (i.e. not compensating the relatively smaller training set with more epochs), rather it deteriorates it. From these results we conclude that there is no reason to believe that the positive effects of using back-translation is caused by an effectively larger training effort, rather than by the advantage of the larger training set itself. We therefore also conclude that it is reasonable to keep the number of epochs constant across experiments, rather than fixing the amount of training effort as measured by steps/mini-batches, and we do the former throughout the rest of the paper.

8.2 Addition of Synthetic Data from SMT and NMT Models

Table 2 shows the results of the performance of the different NMT models we have built. The subtables indicate the size of the data used for building the models (from 1M to 4M lines). In each column it is indicated whether *base* has been augmented with the *auth*, *SMTsynth*, *NMTsynth*, *hybrNMTSMT*, or *fullhybrNMTSMT* data set.

The results show that adding synthetic data has a positive impact on the performance of the models as all of them achieve improvements when compared to that built only with authentic data *1M base*. These improvements are statistically significant at p=0.01 (computed with multeval (Clark et al., 2011) using Bootstrap Resampling (Koehn, 2004)). However, the increases of quality are different depending on the approach followed to create the BT data.

First, we observe that models in which SMT-generated data is added do not outperform the models built with the same size of authentic data. For example, the models built with 4M sentences (1M authentic and 3M SMT-produced sentences, in cell *+ 3M SMTsynth*) achieve a performance comparable to the model trained with smaller number of sentences of authentic data (such as *+ 1M auth* cell, 2M sentences).

Models built by using NMT-created data have a better performance than those built with data generated by SMT. When performing a pairwise comparison between models using an equal amount of either SMT or NMT-created data, we observe that the latter models outperform the former by around one BLEU point. In fact, the performance of models using NMT-translated sentences is closer to those built with authentic data, and some *NMTsynth* models produce better translation qualities. This is the case of *+1M NMTsynth* model (according to all evaluation metrics) or *+3M NMTsynth* (according to BLEU).

Our experiments also include the performance of models augmented with a combination of SMT- and NMT- generated data. We see that adding *hybrNMTSMT* data, with one half of the data originating from SMT and the other half from NMT models, have performances similar to those models built on authentic data only. According to some evaluation metrics, such as METEOR, the performance is better than *auth* models when adding 1M or 2M artificial sentences (although none of these improvements are statistically significant at p=0.01). For these amount of sentences, it also outperforms those models in which only SMT or only NMT BT data have been included.

The models extended with synthetic data that perform best are *fullhybrNMTSMT* models. Furthermore, they also outperform authentic models when built with less than 4M distinct target-sentences according to BLEU, METEOR

	1M *base*.- 13 Epochs	1M *base*.- 39 Epochs-	1M *base* + 2M *NMTsynth*
BLEU↑	23.40	23.22	25.44
TER↓	57.23	58.21	55.62
METEOR↑	28.09	27.75	29.47
CHRF1↑	50.66	50.18	52.5

Table 1: Results for experimental procedure validation: checking that it is reasonable to use constant number of epochs, not constant amount of training effort, in the experiments.

		1M *base.*	-	-	-	-
1M lines	BLEU↑	23.40	-	-	-	-
	TER↓	57.23	-	-	-	-
	METEOR↑	28.09	-	-	-	-
	CHRF1↑	50.66	-	-	-	-
		+ 1M *auth*	+ 1M *SMT-synth*	+1M *NMT-synth*	+ 1M *hybrNMTSMT*	+ 2M *fullhybrNMTSMT*
2M lines	BLEU↑	24.87	24.38	25.32	25.21	25.34
	TER↓	55.81	56.05	55.66	55.87	55.79
	METEOR↑	29.16	28.93	29.33	29.29	29.47
	CHRF1↑	52.03	51.89	52.25	52.36	52.47
		+ 2M *auth.*	+ 2M *SMT-synth*	+ 2M *NMT-synth*	+ 2M *hybrNMTSMT*	+ 4M *fullhybrNMTSMT*
3M lines	BLEU↑	25.69	24.58	25.44	25.62	25.94
	TER↓	54.99	55.7	55.62	55.25	55.11
	METEOR↑	29.7	29.02	29.47	29.73	29.97
	CHRF1↑	52.77	52.09	52.5	52.89	53.11
		+ 3M *auth*	+ 3M *SMT-synth*	+ 3M *NMT-synth*	+3M *hybrNMTSMT*	+ 6M *fullhybrNMTSMT*
4M lines	BLEU↑	25.97	24.65	26.01	25.83	25.86
	TER↓	54.54	55.58	55.33	55.17	54.95
	METEOR↑	29.91	29.26	29.71	29.74	29.88
	CHRF1↑	53.16	52.24	52.87	52.84	53.11

Table 2: Performance of models built with increasing sizes of authentic set (first column) and different synthetic datasets (last four columns). +1M, +2M and +3M indicate the amount of sentences added to the *base* set (1M authentic sentences).

(showing statistically significant improvements at p=0.01) and CHRF1. Despite that, when using large sizes of data (i.e. adding 3M synthetic sentences) the models built with SMT-generated artificial data have the lowest performances whereas the performance of the other three tends to be similar.

8.3 Further Analysis

In order to better understand the described systems, we carried out more detailed analysis of all translation outputs. We analyzed five error categories: morphological errors, word order, omission, addition and lexical errors, and we compared lexical and syntactic variety of different outputs in terms of vocabulary size and number of distinct POS n-grams. We also analyzed the sentence lengths in different translation hypotheses, however no differences were observed, neither in the average sentence length nor in the distribution of different lengths.

Automatic Error Analysis

For automatic error analysis results, we used Hjerson (Popović, 2011), an open-source tool based on Levenshtein distance, precision and recall. The results are presented in Table 3.

	error class rates↓				
training	morph	order	omission	addition	mistranslation
1M *base*	2.8	9.8	12.0	4.8	29.1
1M *base* + 1M *auth*	2.7	9.5	11.4	4.9	28.2
1M *base* + 1M *SMTsynth*	2.8	10.0	11.6	4.8	28.1
1M *base* + 1M *NMTsynth*	2.7	9.8	10.9	5.0	28.1
1M *base* + 1M *hybrNMTSMT*	2.7	9.6	11.4	5.2	27.7
1M *base* + 1M *fullhybrNMTSMT*	2.6	9.5	11.0	5.2	27.8
1M *base* + 2M *auth*	2.6	9.6	11.2	4.8	27.7
1M *base* + 2M *SMTsynth*	2.7	10.0	11.9	4.5	28.0
1M *base* + 2M *NMTsynth*	2.6	9.7	11.1	5.1	27.9
1M *base* + 2M *hybrNMTSMT*	2.6	9.6	11.0	5.2	27.6
1M *base* + 2M *fullhybrNMTSMT*	2.6	9.6	10.7	5.3	27.4
1M *base* + 3M *auth*	2.7	9.8	11.2	4.6	27.6
1M *base* + 3M *SMTsynth*	2.7	9.8	11.9	4.6	27.9
1M *base* + 3M *NMTsynth*	2.5	9.6	11.3	5.3	27.4
1M *base* + 3M *hybrNMTSMT*	2.6	9.5	11.0	5.1	27.6
1M *base* + 3M *fullhybrNMTSMT*	2.5	9.7	10.8	4.8	27.7

Table 3: Results of automatic error classification into five error categories: morphological error (morph), word order error (order), omission, addition and mistranslation.

It can be seen that morphological errors are slightly improved by any additional data, but it is hard to draw any conclusions. This is not surprising given that our target language, English, is not particularly morphologically rich. Nevertheless, for all three corpus sizes, the numbers are smallest for the full hybrid system, being comparable to the results with adding authentic data.

As for word order, adding SMT data is not particularly beneficial since it either increases (1M and 2M) or does not change (3M) this error type. NMT systems alone do not help much either, except a little bit for the 3M corpus. Hybrid systems yield the best results for this error category for all corpus sizes, reaching or even slightly surpassing the result with authentic data.

Furthermore, all BT data are beneficial for reducing omissions, especially hybrid which can be even better than the authentic data result.

As for additions, no systematic changes can be observed, except an increase for all types of BT data. However, it should be noted that this error category is reported not to be very reliable for comparing different MT outputs (see for example (Popović and Burchardt, 2011)).

The mostly affected error category is mistranslations. All types of additional data are reducing this type of errors, especially the hybrid BT data for 1M and 2M, even surpassing the effect of adding authentic data. As for the 3M corpus, the improvement in this error category is similar to the one by authentic data, but the best option is to use NMT BT data alone.

In total, the clear advantage of using hybrid systems can be noted for mistranslations, omissions and word order which is the most interesting category. This error category is augmented by adding BT SMT data or not affected by adding BT NMT data, but combining two types of data creates beneficial signals in the source text.

Lexical and Syntactic Variety

Lexical and syntactic variety is estimated for each translation hypothesis as well as for the human reference translation. The motivation for this is the observation that machine-translated data is generally lexically poorer and syntactically simpler than human translations or texts written in the original language (Vanmassenhove et al., 2019). We want to see how different or similar our translation hypotheses are in this sense, and also how they relate to the reference.

Lexical variety is measured by vocabulary size (number of distinct words) in the given text, and syntactic variety by number of distinct POS n-grams where n ranges from 1 to 4. The results are shown in Figure 1.

First of all, it can be seen that none of the

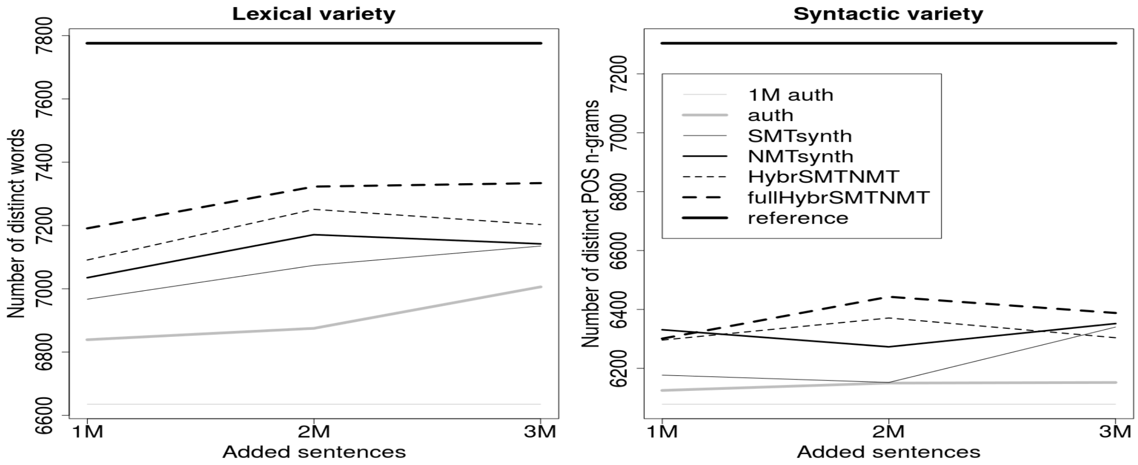


Figure 1: Lexical variety and syntactic variety for all translation hypotheses and for human reference translations.

translation hypotheses reaches the variety of the reference translation (the black line on the top). The difference is even more notable for the syntax, where the differences between translation hypotheses are smaller and the difference between them and the reference is larger than for vocabulary.

Furthermore, it can be seen that for authentic data (thin gray line on the bottom and thick gray line) the variety increases monotonically with adding more text.

Lexical variety is increased by all synthetic data, too, even more than by authentic data, however, for the NMT and hybrid synthetic data the increase for the 3M corpus is smaller than for smaller corpora.

The increase of syntactic variety is lower both for authentic and for synthetic data than the increase of lexical variety. For 1M and 2M corpus, syntactic variety is barely increased by SMT synthetic data whereas NMT and hybrid data are adding more new instances. For the 3M corpus, however, all synthetic methods yield similar syntactic variety, larger than the one obtained by adding authentic data.

Word/POS 4-gram Precision and Recall

Whereas the increase of lexical and syntactic varieties is a positive trend in general, there is no guarantee that the MT systems are not introducing noise thereby. To estimate how many of added words and POS sequences are sensible, we calculate precision and recall of word and POS 4-grams when compared to the given reference translation. The idea is to estimate how much the translation hypotheses are getting closer to the reference. We take word 4-grams instead of single words because it is not only important that a word makes sense in isolation, but also in a context. Of course, it is still possible that some of the new instances are valid despite being different from the given single reference.

The results of precision and recall for word/POS 4-grams are are shown in Figure 2. Several tendencies can be observed:

- hybrid BT data is especially beneficial for the 1M and 2M additional corpora, for 1M even outperforming the authentic additional data, especially regarding word 4-grams;
- NMT BT is the best synthetic option for the 3M additional corpus, however not better than adding 3M of authentic data. This tendency is largest for POS 4-gram precision.
- SMT BT data achieves the lowest scores, especially for POS 4-grams; this is probably related to the fact that it produces less grammatical BT sources, which are then propagated to the translation hypotheses. The differences are largest for the 3M additional corpus, which is probably the reason of diminished effect of the hybrid BT data for this setup.

Overall tendencies are that the hybrid BT data is capable even of outperforming the same amount of authentic data if the amount of added data does not exceed the double size of the baseline authentic data. For larger data, a deterioration can be ob-

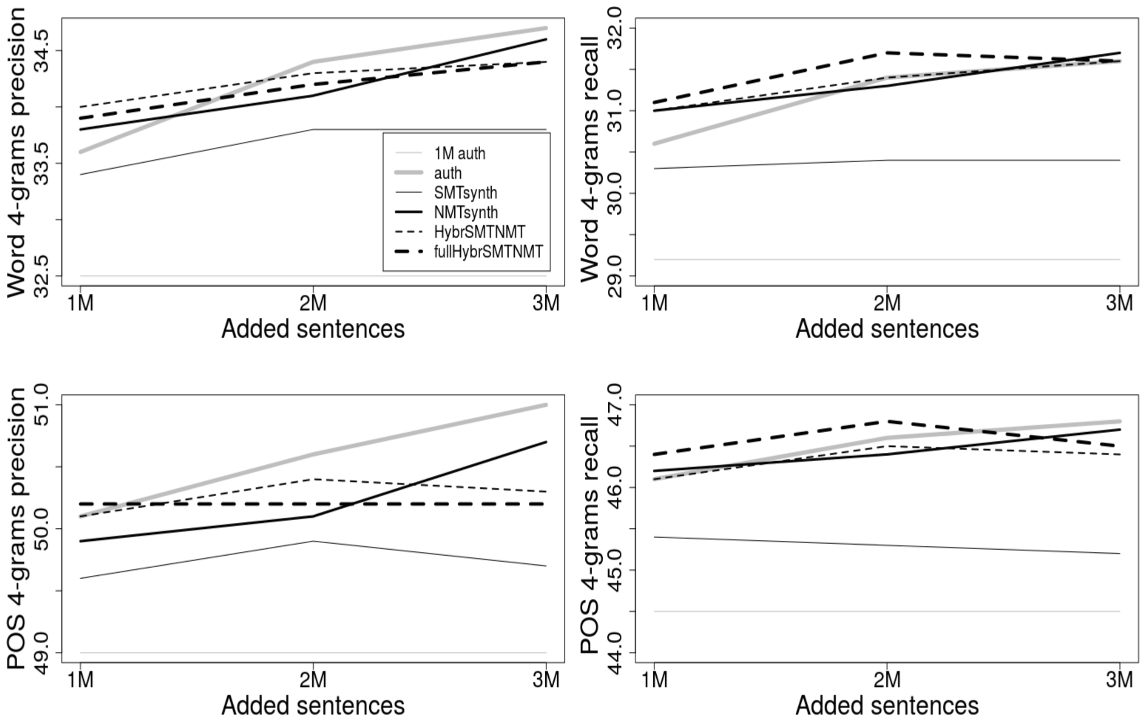


Figure 2: Word/POS 4-gram precision and recall for all translation hypotheses.

served for the SMT BT data, leading to saturation of hybrid models.

Further work dealing with mixing data techniques is necessary, in order to investigate refined selection methods (for example, removing SMT segments which introduce noise).

9 Conclusion and Future Work

In this work we have presented a comparison of the performance of models trained with increasing size of back-translated data. The artificial data sets explored include sentences generated by using an SMT model, and NMT model and a combination of both. Two mixing strategies are explored: randomly selecting one half of the source segments from the SMT BT data and the other half from the NMT BT data, and using all BT source segments thus repeating each target segment.

Some findings from previous work (Burlot and Yvon, 2018) are confirmed, namely that in terms of overall automatic evaluation scores, SMT BT data reaches slightly worse performance than NMT BT data. Our main findings are that mixing SMT and NMT BT data further improves over each data used alone, especially if full hybridisation is used (using two sources for each target side). These data can even reach better performance than adding the same amount of authentic data, mostly by reducing the number of mistranslations, and increasing the lexical and syntactic variety in a positive way (introducing useful new instances).

However, if the amount of synthetic data becomes too large (three times larger than the authentic baseline data), the benefits of hybrid system start to diminish. The most probable reason is the decrease in grammaticality introduced by SMT BT data which becomes dominant for the larger synthetic corpora.

The presented findings offer several directions for the future work, such as exploring efficient strategies for mixing SMT and NMT data for different authentic/synthetic ratios and investigating morphologically richer target languages.

Acknowledgements

This research has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

This work has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 713567.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. Ann Arbor, Michigan, pages 65–72.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark, pages 169–214.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisboa, Portugal, pages 1–46.

Franck Burlot and François Yvon. 2018. Using monolingual data in neural machine translation: a systematic study. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, pages 144–155. http://aclweb.org/anthology/W18-6315.

Rajen Chatterjee, Matteo Negri, Raphael Rubino, and Marco Turchi. 2018. Findings of the WMT 2018 shared task on automatic post-editing. In *WMT (shared task)*. Association for Computational Linguistics, pages 710–725.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*. Portland, Oregon, page 176–181.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 489–500. http://aclweb.org/anthology/D18-1045.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland, pages 187–197.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL*. Berlin, Germany, pages 751–758.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics-System Demonstrations*. Vancouver, Canada, pages 67–72.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain, pages 388–395.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for SMT. In *Proceedings of 45th annual meeting of the ACL on interactive poster & demonstration sessions*. Prague, Czech Republic, pages 177–180.

Matteo Negri, Marco Turchi, Rajen Chatterjee, and Nicola Bertoldi. 2018. ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC*. Miyazaki, Japan.

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *ACL-2003: 41st Annual Meeting of the Association for Computational Linguistics, Proceedings*. Sapporo, Japan, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 311–318.

Alberto Poncelas, Gideon Maillette de Buy Wenniger, and Andy Way. 2019. Adaptation of machine translation models with back-translated data using transductive data selection methods. In *20th International Conference on Computational Linguistics and Intelligent Text Processing*. La Rochelle, France.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating Back translation in Neural Machine Translation. In *Proceedings of the 21st Annual Conference of the European Association for Machine Translation (EAMT 2018)*. Alicante, Spain.

Maja Popović. 2011. Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output. *Prague Bulletin of Mathematical Linguistics* 96:59–68.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal, pages 392–395.

Maja Popović and Aljoscha Burchardt. 2011. From human to automatic error classification for machine translation output. In *Proceedings of the 15th International Conference of the European Association for Machine Translation (EAMT 2011)*. Leuven, Belgium.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, volume 1, pages 1715–1725.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas*. Cambridge, Massachusetts, USA, pages 223–231.

Eva Vanmassenhove, Dimitar Shterionov, and Andy Way. 2019. Loss and Decay of Linguistic Richness in Neural and Statistical Machine Translation. In *Proceedings of the 17th Machine Translation Summit (MTSummit 2019)*. Dublin, Ireland.

Unsupervised dialogue intent detection via hierarchical topic model

Artem Popov[1,2], **Victor Bulatov**[1], **Darya Polyudova**[1], **Eugenia Veselova**[1]
[1]Moscow Institute of Physics and Technology
[2]Lomonosov Moscow State University
popov.artem.s@yandex.ru, viktor.bulatov@phystech.edu,
darya.polyudova@phystech.edu, veselova.er@phystech.edu

Abstract

One of the challenges during a task-oriented chatbot development is the scarce availability of the labeled training data. The best way of getting one is to ask the assessors to tag each dialogue according to its intent. Unfortunately, performing labeling without any provisional collection structure is difficult since the very notion of the intent is ill-defined.

In this paper, we propose a hierarchical multimodal regularized topic model to obtain a first approximation of the intent set. Our rationale for hierarchical models usage is their ability to take into account several degrees of the dialogues relevancy. We attempt to build a model that can distinguish between subject-based (e.g. medicine and transport topics) and action-based (e.g. filing of an application and tracking application status) similarities. In order to achieve this, we divide set of all features into several groups according to part-of-speech analysis. Various feature groups are treated differently on different hierarchy levels.

1 Introduction

One of the most important goals of task-oriented dialogue systems is to identify the user intention from the user utterances. State-of-the-art solutions like (Chen et al., 2017) require a lot of labeled data. User's utterances (one or several for a dialogue) have to be tagged by the intent of the dialogue.

This is a challenging task for a new dialogue collection because the set of all possible intents is unknown. Giving a provisional hierarchical collection structure to assessors could make the intent labeling challenge easier. The resulting labels will be more consistent and better suitable for model training.

Simple intent analysis is based on empirical rules, e.g. "question" intent contains phrase "what is # of #" (Yan et al., 2017). More universal and robust dialogue systems should work without any supervision or defined rules. Such systems can be implemented with automatic extraction of the semantic hierarchy from the query by multi-level clustering, based on different semantic frames (capability, location, characteristics etc.) in sentences (Chen et al., 2015). In our work intents represent a more complex entity which combine all intentions and objectives.

Many previous works take advantage of hierarchical structures in user intention analysis. In paper (Shepitsen et al., 2008) automatic approach through hierarchical clustering for document tagging is used. However, this approach does not take advantage of peculiar phrase features, such as syntax or specific words order. Syntactic parsing of intention was applied in (Gupta et al., 2018) to decompose client intent. This hierarchical representation is similar to a constituency syntax tree. It contains intentions and objects as tree elements and demands deep analysis of every sentence. Attempt to extract subintents along with main intent can be found in paper (Tang et al., 2018), but as proved below it is not necessary to apply neural networks for precise and efficient retrieval of multi-intent, especially in unsupervised task.

We propose a hierarchical multimodal regularized topic model as a simple and efficient solution for accurate approximation of the collection structure. The main contribution of this paper is the construction of a two-level hierarchical topic model using different features on the first and second levels. To the best of our knowledge, this is the first work that investigates that possibility. We introduce a custom evaluation metric which measures the quality of hierarchical relations between topics and intent detection.

The hierarchy structure helps to make a provisional clustering more interpretative. Namely, we require first level topics to describe the dialogue subject and the second level topics to describe the action user is interested in. We accomplish this by incorporating information about part-of-speech (PoS) tags into the model.

This paper is organized as follows. Section two describes popular approaches to an unsupervised text classification. Section three describes our reasoning behind our choices of model architecture. Section four briefly reviews our preprocessing pipeline and introduces several enhancements to the existing NLP techniques. We demonstrate the results of our model in section five. We conclude our work in section six.

2 Text clustering approaches

2.1 Embeddings approaches

The simplest way to build a clustering model on a collection of text documents includes two steps. On the first step, each document is mapped to a real-valued vector. On the second step, one of the standard clustering algorithms is applied to the resulting vectors.

There are many methods to build an embedding of a document. The simplest way is the tf-idf representation. Logistic regression on the tf-idf representation is quite a strong algorithm for the text classification problem. This algorithm is respectable baseline even in deep neural networks research (Park et al., 2019). However, the direct use of the tf-idf representation leads to poor results in the clustering problem because of the curse of dimensionality. Dimensionality reduction methods could be used to improve clustering quality: PCA or Uniform Manifold Approximation and Projection (UMAP, McInnes et al. (2018)).

Another popular approach makes use of different word embeddings (Esposito et al., 2016). First of all, each word is mapped to a real-valued vector. Then the document representation is derived from the embeddings of its words. The most popular embedding models belong to the word2vec family (Mikolov et al., 2013b): CBOW, Skip-gram and their modifications (Mikolov et al. (2013a)). For correct representation word2vec models should be trained on a large collection of documents, for example, Wikipedia. Further improvement in quality of clustering models with embeddings can be achieved through fine-tuning. Similar to the tf-idf approach dimensionality reduction is often employed for the clustering problem (Park et al., 2019). Several averaging schemes can be used to aggregate word embeddings: mean, where all words contribute equally to the document, or idf-weighted, where rare words have a greater contribution than frequent words.

2.2 Topic modeling

Another approach to text clustering problem is topic modeling. The topic model simultaneously computes words and document embeddings and perform clusterization. It should be noted that in some cases topic model-based embeddings outperform traditional word embeddings, (Potapenko et al., 2017). The probability of the word w in the document d is represented by formula below:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t) p(t \mid d) = \sum_{t \in T} \phi_{wt} \theta_{td}$$

where matrix Φ contains probabilities ϕ_{wt} of word w in topic t, matrix Θ contains probabilities θ_{td} of topic t in document d.

Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 2000) is the simplest topic model which describes words in documents by a mixture of hidden topics. The Φ and Θ distributions are obtained via maximization of the likelihood given probabilistic normalization and nonnegativity constraints:

$$L(\Phi, \Theta) = \sum_{d \in D} \sum_{w \in W} n_{dw} \log p(w|d) \;\to\; \max_{\Phi,\Theta}$$

$$\sum_{w \in W} \phi_{wt} = 1, \;\phi_{wt} \geq 0$$

$$\sum_{t \in T} \theta_{td} = 1, \;\theta_{td} \geq 0$$

This optimization problem can be effectively solved via EM-algorithm or its online modifications (Kochedykov et al., 2017).

Latend Dirichlet Allocation (LDA) (Blei et al., 2003) model is an extension of pLSA with a prior estimation of the Φ and Θ, widely used in topic modelling. However, as a solution for both pLSA and LDA optimization problem is not unique, each solution may have different characteristics.

Additive Regularization of Topic Models (ARTM) (Vorontsov and Potapenko, 2015) is non-bayesian extension of likelihood optimization task, providing robustness of the solution by applying different regularizers. Each regularizer is used to pursue different solution characteristics. For example, many varieties of LDA can be obtained from ARTM model by using certain smoothing regularizer; pLSA model is an ARTM model without regularizers. Furthermore, documents can contain not only words but also terms of other modalities (e.g. authors, classes, n-grams), which allow us to select specific for our task language features. In this case, instead of a single Φ matrix, we have several Φ_m matrices for each modality m. Resulting functional to be optimized is the sum of weighted with α_m coefficients modalities likelihoods with regularization terms:

$$\sum_m \alpha_m L(\Phi_m, \Theta_m) + R(\cup_m \Phi_m, \Theta) \rightarrow \max_{\Phi,\Theta}$$

3 Multilevel clustering

Our goal is to build a topic model with topics corresponding to the user's intents. We use the following operational definition of intent: two dialogues (as represented by user's utterances) are said **to have the same intent** if both users would be satisfied with the essentially same reaction by the call centre operator. This definition, while inherently problematic, allows us to highlight several important practical problems:

- Simple bag-of-words (BoW) approach isn't sufficient. Compare: "I want my credit card to be blocked. What should I do¿' and "My credit card is blocked, what should I do¿'.
- In some cases, the intent of conversation is not robust to a single word change. "I want to make an appointment with cardiologist" and "I want to make an appointment with neurologist" are considered to have the same intent since they require the user to perform a virtually identical set of actions. However, "Payment of state duty for a passport" and "Payment of state duty for vehicle" are vastly different.

To account for the BoW problem we add an n-gram modality and p_{tdw} smoothing regularizer (Skachkov and Vorontsov, 2018) for all tokens. The p_{tdw} smoothing regularizer respects the sequential nature of text, making the distributions $p(t|d, w)$ more stable for w belonging to a same local segment. In a way, $p(t|d, w)$ distribution could be interpreted as the analogue for context embeddings in topic modeling world. $p(t|d, w)$ distribution isn't used directly for topic representation, but it is used on the E-step of EM-algorithm for ϕ_{wt} and θ_{td} recalculation.

In order to obtain more control over intent robustness we propose to use a two-level hierarchical topic model. The first level is responsible for coarse-grained similarity, while the second one could take into account less obvious but important differences.

The hierarchical ARTM model consists of two different ARTM models for each level, which are linked to each other. The first level of the hierarchical model can be any ARTM model. The second level is built using regularizer from (Chirkova and Vorontsov, 2016) which ensures that each first-level topic is a convex sum of second-level topics. Various methods could be employed to ensure that each parent topic is connected to only a handful of relevant children topics: one can use either interlevel sparsing regularizer (Chirkova and Vorontsov, 2016) or remove "bad" edges according to EmbedSim metric (Belyy, 2018).

3.1 Distinct hierarchy levels

Building a two-level clustering model is a difficult task due to the inaccuracy of clustering algorithms. Provided that documents in the model first-level clusters are already similar to each other (as they should be), further separation could be complicated (especially if we attempt to subdivide each cluster by the same algorithm). In practice, the second-level clusters tend to repeat first-level clusters at smaller scale instead of demonstrating some meaningful differences. In order to make our model able to distinguish new dissimilarities in clusters on the second level, we adjust algorithm at the second level: in broad strokes, we base the second level of model on different features.

In the context of our problem, separation based on the functional purpose of the model tokens is proposed. We divide all words and n-grams into two groups based on the PoS analysis: "thematic" and "functional". The "functional" group consists of the verb words and n-grams that contain at least one verb. The "thematic" group consists of the nouns and adjectives and n-grams that contain at least one noun and have no verbs. Inspired by multi–level (Tang et al., 2018) and multi–syntactic (Gupta et al., 2018) phrases annotation, among with hierarchical partition, our approach is essential for client goal and subgoals extraction.

The purpose of the first hierarchy level is to determine the conversation subject (the entities the dialogue is about). Hence, at the first level of the hierarchy thematic tokens should have a noticeably higher weight than functional tokens. The purpose of the second level of hierarchy, by contrast, is to determine client intent concerning particular objects (e.g. what action the client is trying to perform). Functional tokens should have higher impact over thematic ones. The tokens unrelated to these two groups are used on both levels and serve as a connection between the layers.

4 Preprocessing

We use standard preprossessing pipeline consisting of tokenization, lemmatization, part-of-speech tagging, n-grams extracting, named entity recognition and spell checking. In this section we describe some details of preprocessing algorithms since the preprocessing is very important for any morphologically rich languages such as Russian.

Data prepossessing pipeline consists of many parts, therefore each part must be relatively fast. That is why we don't use some great powerful approaches such as (Devlin et al., 2018) for NER.

4.1 N-grams extracting

Conventional approach in surpassing the bag-of-word hypothesis of the model is by adding n-grams or collocations into the model. To extract n-grams we use TopMine algorithm (El-Kishky et al., 2014) based on a words co-occurrences statistics.

However, we found it beneficial to implement some modifications. First change alters gathering and usage of word co-occurrences statistics: TopMine differentiates between sequences (w_1, w_2) and (w_2, w_1), which is not desirable for synthetic languages with less strict word order compared to English. To make it better suited to the Russian language, we use multisets as containers for collocations instead of sequences. Second change modifies the extraction process: while the original version of TopMine extracts only disjoint collocations and won't detect sub-collocations (e.g. if n-gram "support vector machines" is extracted, n-gram "support vector" will not be extracted), our modification will extract every high-scoring collocation at the cost of increased memory usage.

4.2 Named entity recognition

There are a lot of references to the speakers' names, company/product names, streets, cities in the dialogue collection. It makes sense to take into account some entities in a special way.

For the named entity recognition problem (NER) different methods are commonly used: rule-based, machine-learning-based or neural-networks-based. We used neural network from Arkhipov et al. (2017) pretrained on a PERSONS-1000 (Vlasova et al. (2014)) for our experiments. We replace all person related tokens by the $\langle \mathrm{PERSON} \rangle$ tag.

4.3 Spell checking

Errors and typos in client utterances are common in the dialogue collection. The simplest way to deal with this problem is to apply a spell checking algorithm. We use Jamspell[1] algorithm for spell checking since its fastness.

We make some modifications to adapt the Jamspell model to our case. First of all, the language model used to select the best correction candidates should be trained on the collection for clustering. This modification takes into account the collection specificity and collection specific words won't be treated as unknown.

Also the set of candidates can be extended. According to the statistics of Yandex search engine[2] word merging error is one of the most popular typos in the dialogues. Hence, we add candidates that are obtained via splitting a word in two.

5 Experiments

We use two dialogue datasets from the Russian call-centres ($\sim 90K$ dialogues in each) in our experiments. The first dataset is collected from client dialogues with various public services. The second dataset is conversation logs of ISP tech support. All dialogues are between a user and a call agent, mean length of a single dialogue is six utterances. Both datasets are proprietary.

5.1 Scoring metric

There are several approaches for measuring the quality of topic model, especially its interpretabil-

[1] Jamspell github

[2] Yandex search errors statistics (on Russian)

ity. The usual procedure involves evaluating the list of most frequent topic words by human experts. However, this approach suffers from several fundamental limitations (Alekseev, 2018). Therefore we choose to employ a different method.

For each dataset, we collect a set of dialogue pairs to score our model. Following the reasoning outlined in the section 3, we generated a number of (d_1, d_2) pairs (where d_i is a dialogue) and asked three human experts to label them. To measure the quality of the model, we compare these labels to the labels predicted by model.

The following list summarizes our approach for model estimation and labeling guidelines for human experts:

- `0`: d_1 and d_2 have nothing in common. Such objects should correspond to the different first-level topics.
- `1`: both d_1 and d_2 are related to the same subject, but there are significant differences. Such dialogues should correspond to the same first-level topic, but to the different second-level topics.
- `2`: d_1 shares an intent with d_2. Such dialogues should correspond to the same first-level and second-level topics.
- `?`: it is impossible to determine the intent for at least one of the dialogues.

We select the best model according to the accuracy metric on a given labeled pairs. Three sets of pairs are used for the estimation ($\sim 12K$ and $\sim 1.5K$ for the first dataset, $\sim 1.5K$ for the second dataset). All model hyperparameters are tuned according to the accuracy on a $12K$ dataset ("1-big"). Two other sets are used to control overfitting ("1-small" and "2-small"). Notably, the good performance on 2-small dataset implies that the model generalizes beyond the initial training dataset.

The same preprocessing procedures are used for both datasets. All tokens are lemmatized, stop-tokens are deleted, simple entities (e-mails, websites e.t.c) are replaced by their tags. Operator utterances are deleted from the dialogue document (they are not informative in our datasets; for example, there are many cases where operator fails to reply at all). Finally, each document is a concatenation of one dialogue user utterances from a single dialogue.

5.2 Baselines

As one of the baselines, we use the following procedure. First, we convert raw texts into real-valued vectors using pretrained embeddings or tf-idf scores in a way described in 2.1. Second, we cluster this dataset via K-Means algorithm. Third, we treat each cluster as a separate collection and perform K-Means algorithm again. As a result, we obtain both first-level and second-level clusters.

Another baseline models are hierarchical topic model without any additional regularizers and hierarchical topic model with Φ and Θ smoothing for both levels. For K-Means based algorithms we tune embeddings dimensionality and both level cluster number. For topic modeling based algorithms we tune both level topics number. As shown in table 1 regularized topic model outperforms K-Means approaches at two out of the three pair sets.

	1-big	1-small	2-small
hKmeans (tf-idf)	0.568	0.593	**0.649**
hKmeans (emb.)	0.615	0.638	0.641
hPLSA	0.603	0.675	0.633
hARTM	**0.636**	**0.683**	0.631

Table 1: Baselines accuracy

5.3 Proposed model perfomance

We use several NLP-based techniques described in 4 to improve main model quality. We start with the hPLSA model. For each problem we test a few approaches and choose the best one. We add all main features one by one, e.g. we choose the best method for extracting n-grams and use it on the next step. We conduct all the experiments in the following order:

1. including additional n-gram modality, choosing between the based and modified n-grams extracting methods, tuning modality weights and topics number;
2. adding p_{tdw} smoothing at the first model level for all tokens, tuning regularizer coefficient and topics number;
3. replacing person related named entities, choosing between the dictionary-based and rnn-based methods;
4. typo correction, choosing between the base and modified algorithm

	1-big	1-small	2-small
hPLSA	0.603	0.675	0.633
+ n-grams base	0.612	0.634	0.633
+ n-grams mod.	**0.635**	**0.674**	**0.655**
+ ptdw smooth.	**0.64**	**0.678**	**0.66**
+ NER dict.	0.634	0.661	0.635
+ NER NN	**0.64**	**0.68**	**0.662**
+ Jamspell	0.635	0.674	0.655
+ mod. Jamspell	**0.657**	**0.686**	**0.663**

Table 2: NLP techniques quality improvement

As the table 2 demonstrates our n-grams extraction method outperforms traditional TopMine algorithm in this task. Replacing persons by a tag does not lead to a great improvement of the quality. Our analysis of hPLSA cluster top-tokens shows that only 3% of the top-tokens are related to persons. After the NER preprocessing the proportion of named entities in top tokens reduces to 0.3%. And at the same time spellchecking improves the performance on all three pair sets. It should be noted that standard Jamspell algorithm leads to a quality decrease.

Finally, we apply feature grouping schemes proposed in 3.1. The results (table 3) turned out to be reassuring. There is a noticeable performance boost for all of the pair sets.

	1-big	1-small	2-small
featured hARTM	0.657	0.686	0.663
+ groups	**0.667**	**0.715**	**0.672**

Table 3: Grouping feature quality improvement

Further, we represent some examples of the model performance. All example texts from examples were translated from Russian to English. In the table 4 all subtopics of the topic "Tariff plan" are presented. Each subtopic described by the characteristic question.

In the table 5 we demonstrate top documents corresponding to the "How do I switch from credit to advance payment?" subtopic.

6 Conclusion

In this paper, we report a success in formalizing the clustering process suitable for unsupervised inference of user intents.

The realization that any intent consists of two crucial parts: the entity relevant to the user's request and the action user wishes to perform helped us to choose a two-level hierarchical model as our main tool. This leads us to design a custom quality metric which takes into account several degrees of the dialogues relevancy.

Tariff plan
How to change the tariff plan?
When did the tariff change happen?
How often can I change my tariff plan?
When will the changes take effect when the tariff is changed?
Why can't I change the tariff?
Why was the tariff plan changed without my knowledge?
Why there are no available tariff plans for the transition?

Table 4: Subtopics of topic "Tariff plan"

How do I switch from credit to advance payment?
How do I switch from credit to advance payment?
Hi. Tell me can we change the credit system of payment to advance? Well thanks!
I need to change my payment from credit to advance.
How to disable credit payment system?
Hello. Change the payment system from credit to advance!
Good morning. How to change the payment system from credit to normal?
Disable the credit payment system.

Table 5: Top documents of subtopic "How do I switch from credit to advance payment?"

Our next step was to devise a PoS-based feature separation and to leverage n-grams, named entities and spellchecking. This allowed us to construct a hierarchical multimodal regularized topic model which outperforms all baseline models.

Acknowledgments

We thank our colleagues Alexey Goncharov and Konstantin Vorontsov from Machine Intelligence Laboratory who provided expertise that greatly assisted the research. We thank Evgeny Egorov for comments that greatly improved the manuscript.

References

V.A. Bulatov V.G. Vorontsov K.V. Alekseev. 2018. Intra-text coherence as a measure of topic models' interpretability. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference Dialogue*. pages 1–13.

Mikhail Y Arkhipov, Mikhail S Burtsev, et al. 2017. Application of a hybrid bi-lstm-crf model to the task of russian named entity recognition. In *Conference on Artificial Intelligence and Natural Language*. Springer, pages 91–103.

A.V. Seleznova M.S. Sholokhov A.K. Vorontsov K.V. Belyy. 2018. Quality evaluation and improvement for hierarchical topic modeling. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference Dialogue*. pages 110–123.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2015. Learning semantic hierarchy with distributed representations for unsupervised spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. 2017. Dialogue act recognition via crf-attentive structured network. *CoRR* abs/1711.05568.

Nadezhda Chirkova and Konstantin Vorontsov. 2016. Additive regularization for hierarchical multimodal topic modeling. *Journal Machine Learning and Data Analysis* 2(2):187–200.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare R Voss, and Jiawei Han. 2014. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment* 8(3):305–316.

Fabrizio Esposito, Anna Corazza, and Francesco Cutugno. 2016. Topic modelling with word embeddings. In *Proceedings of the Third Italian Conference on Computational Linguistics CLiC-it 2016)*. pages 129–134.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *EMNLP*.

Thomas Hofmann. 2000. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization pages 914–920.

Denis Kochedykov, Murat Apishev, Lev Golitsyn, and Konstantin Vorontsov. 2017. Fast and modular regularized topic modelling. In *2017 21st Conference of Open Innovations Association (FRUCT)*. IEEE, pages 182–193.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* .

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Jinuk Park, Chanhee Park, Jeongwoo Kim, Minsoo Cho, and Sanghyun Park. 2019. Adc: Advanced document clustering using contextualized representations. *Expert Systems with Applications* .

Anna Potapenko, Artem Popov, and Konstantin Vorontsov. 2017. Interpretable probabilistic embeddings: bridging the gap between topic models and neural networks. In *Conference on Artificial Intelligence and Natural Language*. Springer, pages 167–180.

Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, pages 259–266.

Nikolay Skachkov and Konstantin Vorontsov. 2018. Improving topic models with segmental structure of texts. In *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference Dialogue*. pages 652–661.

Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li, and Tony Jebara. 2018. Subgoal discovery for hierarchical dialogue policy learning. In *EMNLP*.

Nataliya Vlasova, Elena Syleymanova, and Igor Trofimov. 2014. The russian language collection for the named-entity recognition task. *Language semantics: models and technologies* pages 36–40.

Konstantin Vorontsov and Anna Potapenko. 2015. Additive regularization of topic models. *Machine Learning* 101(1-3):303–323.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Graph Embeddings for Frame Identification

Alexander Popov
IICT, Bulgarian Academy of Sciences
Sofia, Bulgaria
alex.popov
@bultreebank.org

Jennifer Sikos
IMS, University of Stuttgart
Stuttgart, Germany
jen.sikos
@ims.uni-stuttgart.de

Abstract

Lexical resources such as WordNet (Miller, 1995) and FrameNet (Baker et al., 1998) are organized as graphs, where relationships between words are made explicit via the structure of the resource. This work explores how structural information from these lexical resources can lead to gains in a downstream task, namely frame identification. While much of the current work in frame identification uses various neural architectures to predict frames, those neural architectures only use representations of frames based on annotated corpus data. We demonstrate how incorporating knowledge directly from the FrameNet graph structure improves the performance of a neural network-based frame identification system. Specifically, we construct a bidirectional LSTM with a loss function that incorporates various graph- and corpus-based frame embeddings for learning and ultimately achieves strong performance gains with the graph-based embeddings over corpus-based embeddings alone.

1 Introduction

Frames are common scenarios, expressed by their typical participants and the predicates which evoke them. The INFECTING frame, where someone transmits an illness, has participants INFECTED_ENTITY and INFECTION_CAUSE and is evoked by predicates *infect* and *give*. A single predicate can evoke one or more frames; *give*, for instance, can evoke INFECTING (*give someone a cold*) or GIVING (*give someone a present*). The disambiguation of which frame is evoked in context is the task of *frame identification* (FrameID), which is, in essence, a word sense disambiguation task where the senses are frames (Das et al., 2010).

The FrameNet resource (Baker et al., 1998) provides concrete definitions of frames, predicates, and participants (called *frame elements*) and is structured as a hierarchical graph. Frames towards the top of the hierarchy are more abstract (ex. INTENTIONALLY_ACT), while frames lower in the hierarchy are more granular (ex., SUBMITTING_DOCUMENTS). FrameNet's graph structure captures the relationship between frames, their predicates, and their frame elements, such that a single frame can be connected to multiple frames via different relationships (Baker and Ellsworth, 2017). Given the structure of the resource, it is surprising that the graph is not used in current FrameID systems, as the relationships in the FrameNet graph have been previously shown as relevant to frame prediction, especially in cases of unseen predicates (Das et al., 2014).

This work leverages FrameNet's graph structure to boost the performance of neural architectures for FrameID. Specifically, we construct frame embeddings from the FrameNet graph structure and use them as input to a neural network. Although frame embeddings are often used in neural architectures for FrameID, prior work only learns these embeddings from frame-annotated corpus data (Hartmann et al., 2017; Hermann et al., 2014). To our knowledge, this is the first work to incorporate embeddings composed from the FrameNet graph structure itself, thus incorporating all frame, frame element, and predicate relationships that could otherwise be missing from corpus data. We expand our frame graphs with knowledge from another lexical resource, WordNet, to achieve further gains in performance.

Our paper is structured as follows. Section 2 describes prior work in graph embedding models and FrameID. We define our model in Section 3, and

Section 4 focuses on the FrameNet and WordNet graphs, while Sections 5 and 6 explain how we use the FrameNet knowledge base to build graph embeddings. Section 7 outlines our experiments, and Section 8 gives results and further analysis of the model performance.

2 Background

2.1 Graph Embeddings

Graph algorithms, especially random walk algorithms, have been applied to prediction tasks such as word sense disambiguation (Agirre et al., 2014), measuring semantic similarity and relatedness between words (Agirre et al., 2010), and entity linking (Guo et al., 2011). These algorithms traverse over the relations and nodes in a large knowledge base such as WordNet (Miller, 1995) or taxonomies built from Wikipedia links (Cucerzan, 2007) to uncover relationships between the nodes in the knowledge base. However, the effectiveness of shallow neural networks in learning word similarity, as shown by the popular Word2Vec and GloVe models (Mikolov et al., 2013; Pennington et al., 2014), rapidly replaced traditional graph-based methods for word similarity and prediction tasks. Word representations learned in these models, called *embeddings*, provide the latent features of a word in context as low-dimensional vectors. Recent work has sought to incorporate the best of graph algorithms and embeddings by learning representations for words over a neural network while using the structural information found in knowledge bases (Goikoetxea et al., 2015; Faruqui et al., 2015).

Our method for constructing graph embeddings follows the work of Goikoetxea et al. (2015) who use random walks to build a synthetic corpus based on entries in WordNet. In this corpus, a word's contexts are the other words in the knowledge base that it is related to. They use the corpus to generate embeddings for words with CBOW and Skipgram models.

2.2 Neural Architectures for Frame Identification

The first work to use embeddings for FrameID used the WSABIE algorithm to project frames and predicate contexts into the same shared space (Hermann et al., 2014). The authors then apply a pairwise loss to minimize the distance between the frames and their predicate instances. Subsequent FrameID models followed, including a system that constructed frame embeddings using the Word2Vec model (Botschen et al., 2017). More recent state-of-the-art models use contextualized embeddings of frames in the BERT framework (Sikos and Padó, 2019) or joint models with semantic dependencies and frames (Peng et al., 2018). All of these prior neural architectures construct frame embeddings directly from annotated corpora, meaning the frame embeddings that are used to make predictions in the model are limited to frames that are seen in the corpora.

3 Frame Identification Model

This section describes the overall architecture of our FrameID model. We adopt a bidirectional recurrent neural network (Bi-LSTM) that accepts as input different embeddings that represent frames and predicates. This allows us to measure the effectiveness of the pre-trained embeddings (corpus-based, graph-based and combined) for an apples-to-apples comparison within the same setting. The components of our neural network include:

- an input embedding layer; the network can allow two embedding inputs per word – the embeddings from separate sources (corpus/graph-based) are concatenated;
- bidirectional recurrent layers; the forward and backward states are concatenated;
- an output vector with the same dimensionality as the frame embeddings;
- an objective function that compares the final output and the gold frame embeddings; at those steps that do not introduce frame-evoking predicates the network attempts to reconstruct the embedding for the input word itself, so that "infect" should result in the embedding of INFECTING, but "the" should just be reconstructed to the embedding of "the".

At evaluation time, output vectors are compared via cosine similarity to the frame embeddings, where the label for the closest frame is selected as the model's prediction. Figure 1 shows the architecture of the network.

4 FrameNet as a Graph

As described above in Section 1, FrameNet has a structure which connects frames via different semantic relations. Relations in the knowledge base include *Inheritance*, an *is-a* relation akin

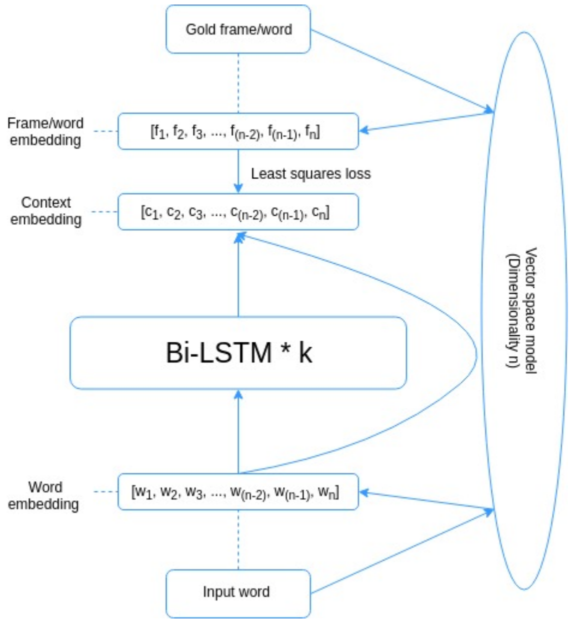

Gold frame/word
Frame/word embedding
$[f_1, f_2, f_3, ..., f_{(n-2)}, f_{(n-1)}, f_n]$
Least squares loss
Context embedding
$[c_1, c_2, c_3, ..., c_{(n-2)}, c_{(n-1)}, c_n]$
Bi-LSTM * k
Word embedding
$[w_1, w_2, w_3, ..., w_{(n-2)}, w_{(n-1)}, w_n]$
Input word
Vector space model
(Dimensionality n)

Figure 1: Bi-LSTM for FrameID. The diagram shows a model with one embedding layer, but combining with an additional embedding source is done trivially via concatenation. The embeddings used for the input and the gold frames can either come from the same or a different vector space. The size of the word/frame embeddings is **n**; **k** is the number of hidden layers.

to hypernymy in WordNet (e.g., REVENGE inherits from REWARDS_AND_PUNISHMENTS), *Using*, which connects a concrete frame to a related, but more abstract frame (e.g. SPEED uses the MOTION frame), and *Subframe*, where the parent frame is a more complex event that subsumes the child frame(s) (e.g. CRIMINAL_PROCESS subsumes ARREST). See Ruppenhofer et al. (2006) for a more detailed description of all possible relation types.

This network of relations is used as the basis of our graph. We construct a graph where each frame is a node, and we build edges between frames according to the available relational information in FrameNet. Frame elements (FEs) are also nodes that are connected to their frames. Some FE nodes, such as AGENT, are widely connected in the graph since they are linked to a diverse set of frames. In order to have a single FE node be distinctive to a single frame, the FE nodes are given unique identifiers. For the unique identifier, the frame name is prepended to the name of the FE together with a colon separator, such as HIRING::EMPLOYEE. However, these unique FE nodes are further connected to their abstract counterparts with an *is-a* relation (REVENGE::AVENGER *is-a* AVENGER), which preserves the connectedness of the FE in FrameNet and also provides some connectivity between frames that share the same FE.

4.1 Extending the FrameNet graph

We empirically find that, on its own, the graph described above does not form a dense enough graph structure. This is due to the fact that few frames can be connected through short paths since some frames contain little to no relations to others and FrameNet FEs are often very specific to particular types of scenarios.

To overcome this limitation, we mapped FrameNet to the WordNet lexical resource. WordNet groups synonymous words of the same part-of-speech category into concept clusters called *synsets* and provides lexical relations between word senses and the relations between synsets. Its dense semantic network provides a rich representation of the ecology of lexical meaning, especially when associated resources are brought into play. This allows for connecting concepts that might not be linked via lexico-semantic relations but nevertheless are mutually dependent due to contextual co-occurrence.

Mapping FrameNet predicates with the Predicate Matrix The mapping process is done in two steps. First, the verbal predicates in FrameNet are connected to their corresponding synsets in WordNet. Since one word can be mapped to several senses in WordNet, this mapping is far from straightforward. We map FrameNet predicates and WordNet synsets through an auxiliary resource called the Predicate Matrix (De Lacalle et al., 2014), which aimed to automatically extend the predicate mappings from SemLink (Palmer, 2009). Within the Predicate Matrix, predicates from a FrameNet frame are linked to concept nodes, which are also connected to a corresponding WordNet synset. Via these synsets, we now have access to WordNet's semantic network, where words are interconnected through lexico-semantic relations such as hypernymy, antonymy, meronymy, as well as through other relations expressing relatedness.

Lexical fillers for FrameNet FEs via Wikipedia and BabelNet The second step in the mapping process is the automatic expansion of FrameNet by extracting typical fillers for FrameNet FEs.

For instance, SELLER is strongly linked to lexical items such as *cashier* or *salesman*, both available in WordNet. While the Predicate Matrix aligns the predicates in FrameNet to WordNet synsets, we additionally align these candidate fillers from WordNet to the FEs they instantiate in FrameNet. Aligning FrameNet's FEs with WordNet is made possible by an automatically created extension of FrameNet (Bryl et al., 2012). The method used by the authors relies on a machine learning model linking nominal lexical fillers for the FEs in the authors' FrameNet reference corpus to Wikipedia pages. The BabelNet resource (Navigli and Ponzetto, 2010) provides mappings between Wikipedia pages and the corresponding WordNet synsets, which are mapped to the frame-FE tuples. Because the process is automated, the correspondences are inevitably noisy. However, it provides a significant expansion of words are not directly given by the FrameNet lexicon but nevertheless intuitively related to the concepts in the frame; e.g., a *cashier* and *salesman* are implicitly related to the COMMERCE_SELL frame. In this sense, this step incorporates more world knowledge into a frame's graph.

5 The FrameNet Graph as a Pseudo-Corpus

Once the nodes in FrameNet and WordNet are aligned, the resulting graph structure is leveraged to produce a large corpus of artificial sentences on which a vector space model can be trained. We follow the methodology of Goikoetxea et al. (2015) for the generation of a pseudo-corpus, where each line contains a sequence of node identifiers visited during one random walk of the algorithm as it moves along the graph. This is done with the UKB tool[1] and its "ukb_walkandprint" functionality. The *wemit_prob* parameter is set to 0.5, which means that in half the cases it emits a dictionary key associated with the node ID generated along the random walk, i.e., half the "words" in the artificial sentences are either predicates or frames corresponding to synsets, as established in the mapping process.

The UKB tool uses a particular dictionary format for the creation of pseudo-corpora. Each line is constructed as follows: a key, often a predicate, begins the entry, followed by its associated WordNet synsets. The result looks like the following:

employment 13968092-n:9 00584367-n:9 01217859-n:6 00947128-n:1

where the synset number is given along with a letter after the synset ("-n"), which signifies the POS category and the number (":9") is the number of instances of this specific sense found in a reference corpus. The *ukb_walkandprint* functionality requires this dictionary for the emission of lexical items from the visited nodes along the random walks. In the case when *wemit_prob* is set to a non-zero value, the tool will pick, with probability *wemit_prob*, one dictionary key associated with the current WordNet synset. The dictionary provides information on what items belong to which synset. If the *dict_weight* parameters are provided, the tool will also take into consideration the count-based weights for each predicate-synset pairing and emit a predicate according to the available probability distribution.

Incorporating FrameNet frames into the UKB dictionary Because the graph now includes frames and FE relations, which are outside of the WordNet lexicon, those need to be included in the dictionary as well. Therefore, we add the frame IDs to the dictionary. This is done in two ways; first, the frames are added as keys corresponding to the synsets to which they have been mapped - below, the frame RENTING is mapped:

Renting 02208537-v:1 02460619-v:1 02208903-v:2

Second, frames are added as values of lexical items that evoke them, as per the information encoded in the FrameNet database. For instance, the predicate *hire.v* is evoked by the RENTING frame:

hire.v Renting:0 02208537-v:0 02460619-v:1 02409412-v:33 Hiring:34[2]

A frame ID can be emitted whenever a connected synset node is visited during a random walk, or if a frame node is visited, it might emit a lexical item connected to the frame as a key in the dictionary. Thus the pseudo-corpus includes both references to specific frames and frame-evoking expressions whose use is contextualized with respect to particular frames. The weights for the frames in the dictionary are calculated by summing up the weights

[1] http://ixa2.si.ehu.es/ukb/

[2] Note that because we are interested in obtaining representations for lexical units (i.e. *lemma.pos* items), the dictionary is modified to include morphological information per each key.

for the lemma-synset pairs mapped to the frames-as-values.

Augmenting the Graph with Wikipedia With the graph and dictionary in place, *ukb_walkandprint* is used to generate 200 million random walks. This pseudo-corpus is then concatenated to a lemmatized and POS-tagged Wikipedia dump (where each word is transformed to a *lemma.pos* lexical item), so that the embeddings can be trained on natural language text in addition to the graph-generated artificial sentences. This is a naive method to combine graph and text information, in that the model is never trained on both kinds of information in one and the same sentence, but it nevertheless allows it to encode real-text syntagmatic information within the lexical space projected from the graph structure. The Wikipedia data is approximately the same size as the pseudo-corpus.

6 Frame Embeddings

Graph embeddings of frames We now have a pseudo-corpus composed from the FrameNet/WordNet graph structure, as described in Section 5. These graph representations are constructed for all the frames in the FrameNet lexicon. We use the popular Word2Vec tool[3] to generate embeddings from the pseudo-corpus. This model uses negative sampling to learn word representations, and we apply the Skipgram variant of the model where a single word is used to predict its neighboring terms in the immediate context. This produces a large inventory of graph embeddings for frames in FrameNet and lexical items, all located in the same space.

Corpus embeddings of frames To compare the graph embeddings with frame embeddings learned from a corpus, we took the freely available embeddings from Sikos and Padó (2018), in which the authors generated frame embeddings using the Word2Vec tool. Specifically, the authors took lemmatized sentences from the FrameNet annotated data and replaced each predicate (e.g., **say.v**) with the frame it evokes:

FrameNet sentence	Officials **say.v** he left
Frame evoked	STATEMENT
Embedding Input	official STATEMENT he left

This resulted in sentences of frames in context, so the example input above would not only produce an embedding for the frame STATEMENT, but also embeddings for the context words such as *official*. In addition to training embeddings on this frame corpus, the authors also provide embeddings based on the original FrameNet annotated corpus. This is important in our case, as we can use the embeddings trained on the original corpus at the input step and the embeddings trained on the frames corpus at the output.

7 Experiments

7.1 Model Setup

The inputs to our model are frame and lexical unit embeddings with 300 dimensions, and the Bi-LSTM has 1 recurrent layer. Each LSTM cell is of size 200, with 0.2 dropout applied to all its sublayers during training. We use the Adam optimizer with a least squares loss function. The training sentences are presented in batches of 128.

The graph that we use for generating the training data for the embedding models consists of 129,101 nodes and 1,146,508 edges. The nodes correspond predominantly to WordNet synsets, with the rest being frame and FE IDs. The synset and FE IDs can be part of the pseudo-corpus or can be omitted from it, depending on the parametrization of the *ukb_walkandprint* command. Depending on this choice, the embeddings will either contain representations of synsets and FEs, or will not, since they are not present as keys in the UKB dictionary.

The graph embeddings were produced by the Word2Vec tool in the Skipgram variant, which has many hyperparameters that can be tuned. We use the following hyperparameters to generate the graph embeddings: size=300; window=15; sample=1e-7; negative=5; iter=7.

7.2 Datasets

We use the annotated data from the FrameNet v1.5 full text annotations to train our models. Sentences are drawn from the balanced BNC corpus[4], and overall there are over 11k frame-evoking predicates with their frames manually annotated. Standard training, development and test splits are defined by Das et al. (2014), where 39 documents are used in training with over 15k target predicates, 16

[3] https://code.google.com/archive/p/word2vec/

[4] http://www.natcorp.ox.ac.uk/

Model	Full Lexicon	Ambiguous	No Lexicon	Unseen
Das et al. (2014)	83.60	69.19	-	23.08
Hermann et al. (2014)	88.41	73.10	-	-
Botschen et al. (2018)	88.82	75.28	81.21	-
I:GloVe & O:FN corpus embeddings	84.76	67.41	63.94	12.36
I&O:Graph embeddings (no FE & synsets)	83.27	64.39	46.29	9.96
I&O:FN corpus embeddings	85.89	69.93	67.82	11.25
I&O:Graph embeddings	86.06	70.29	73.71	19.93
I&O:Graph embeddings + FN corpus embeddings	87.03	72.48	77.15	30.44

Table 1: Evaluation of FrameID with different frame embeddings. The table provides results obtained with the same architecture, but with different input and output embedding models; *I* stands for *input embedding model* and *O* – for *output embedding model*; + means concatenation of vectors from two embedding models.

documents for development with over 4,500 target predicates, and 23 documents for testing with 4,458 target predicates.

7.3 Evaluation Metrics

Standard FrameID systems are evaluated via several different metrics, and we evaluate our models on the most common types. "Full Lexicon" evaluation uses knowledge of the FrameNet lexicon, so that for each predicate we classify the most likely frame given the list of all possible frame candidates for that predicate. "Ambiguous" only runs evaluation on predicates that can evoke multiple frames, thus making the evaluation more challenging by removing predicates that can only evoke one frame. "No Lexicon" reports only results from the classifier, where all frames in the FrameNet database are considered possible candidates. Finally, "Unseen" is the most challenging, as it only evaluates predicates that were not seen in the training data and also does not incorporate knowledge from the lexicon.

8 Results

Results of our models are given in Table 1. The best performing model combines graph-based embeddings with corpus-based embeddings for frame prediction, where strong gains are seen over the basic model that uses corpus-based embeddings alone. "Unseen" results in particular show the largest performance gains, where the combined graph- and corpus-based embeddings allow the model to generalize over new predicates.

The model that uses the popular GloVe embeddings (Pennington et al., 2014) at the input step performs worse than the model that uses the FN corpus embeddings at input and output. This indicates that learning the mapping between predicates and frames is much easier if their corresponding representations are drawn from identical data. The graph-based embeddings are somewhat better than those based solely on the FrameNet annotated corpus; the difference is especially pronounced in the evaluations without the lexicon. However, the graph embedding model that does not incorporate WordNet synsets and FrameNet FEs seems to perform the worst, indicating that those resources provide a strong conceptual skeleton for situating the embeddings of frames and lexical items. What is clear from the results, then, is that both corpus-based and graph-based embeddings contribute to frame prediction and that a richer structure underlying the graph-based embeddings is crucial for improved accuracy.

Our model underperforms compared to other embedding frameworks from Hermann et al. (2014) and Botschen et al. (2018), which can be explained through an examination of the input representation methods used by the different models, as well as their disambiguation strategies. The model by Hermann et al. (2014) constructs an input representation that encodes the syntactic dependency relations found within the predicate context by concatenating the embeddings for the arguments and learning a mapping to a lower-dimensional space. In this way it is similar to our recurrent neural network, but instead of learning the syntactic information implicitly, it feeds it directly, which potentially gives it an advantage. In our experimental setup this could be remedied by training a syntactic dependency parser that shares hidden layer parameters with the FrameID module. Since the FrameNet corpus is not annotated with such data, sequential transfer learning can be employed in order to train on two different signals at different stages (Ruder, 2019).

The Botschen et al. (2018) model is most significantly different from ours in two respects: it uses multimodal embedding representations at the input (textual + visual), and it employs a softmax classifier at the output step, whereas we use MSE as a loss function. Prior work has shown that the first option is more powerful in the context of word sense disambiguation tasks (Popov, 2017). The two classification approaches can be combined, however, in a multitask learning setting to boost accuracy with respect to both (Popov, 2019). Since our main goal in this paper has been to demonstrate the benefits of embedding FrameNet concepts using a graph model, we leave the task of improving the accuracy of the framework for future work.

Nevertheless, our Bi-LSTM does perform well compared to the Das et al. (2014) system, and we get strong results in the "No Lexicon" condition, suggesting the model is able to successfully learn frame categories without any knowledge from the lexicon. The "Unseen" metric shows a complementarity between the graph- and corpus-based embeddings, which further suggests that the two sources of information encode very different lexical and world knowledge.

8.1 Linguistic Analysis

We proceed with a qualitative comparison of frame performance to establish which frames have a boost in performance with graph or corpus-based knowledge. To discriminate the best performing frames within each model, we assign a ranking function for frame performance. We run this ranking function over each model to obtain a ranked list of its best performing frames.

Ranking combines the accuracy of each individual frame $Acc(F_i)$ with an added weight for the frequency of the frame in the corpus $Freq(F_i)$ to obtain an overall rank score $Rank(F_i)$. $Acc(F_i)$ is defined as the number of correctly predicted instances of the frame over the total number of instances. $Freq(F_i)$ is the total number of frame instances over the total number of all frame instances in the test corpus. The $Freq(F_i)$ weight ensures that frames with higher counts in the corpus are ranked higher than frames that have few instances, so therefore a frame with a perfect score will receive a higher rank when there are 80 instances of that frame than a perfect score with only a single instance.

We define $Rank(F_i)$ of a single frame as:

$$Rank(F_i) = Acc(F_i) + Freq(F_i) \tag{1}$$

We then apply the ranking formula to all frames F in the test data and select the 20 highest ranked frames. The highest ranking frames for each model are given in Table 2 and Table 3. Table 2 gives frames that perform better with graph-based knowledge, including AWARENESS, TIME_VECTOR, GOAL, and DEPARTING. Interestingly, AWARENESS and TIME_VECTOR both have a large number of predicates, where AWARENESS has predicates *comprehend.v*, *know.v*, and *understand.v*, and TIME_VECTOR contains mostly function words such as *after.prep*, *before.prep*, and *since.adv*. Both frames have frame elements that appear in multiple related frames, including EXPRESSOR and COGNIZER, which appear in AWARENESS and other frames relating to mental activity, and LANDMARK_EVENT in the TIME_VECTOR frame, which also appears in the closely related TEMPORAL_COLLOCATION frame. This suggests that their graph structures are large but the knowledge in these frame graphs are tied to a specific domain.

Alternatively, Table 3 shows frames that perform better in the corpus-based model, including ORIGIN, LEADERSHIP, and NATURAL_FEATURES. Their predicates are mostly restricted to one sense – ORIGIN, for instance, has predicates *jamaican.n*, *canadian.n*, and *french.a*, while NATURAL_FEATURES has predicates *mountain.n* and *continent.n*. However, their frame elements are more likely to appear across different domains. LOCALE is a frame element of the NATURAL_FEATURES frame, and it appears in POLITICAL_LOCALES which has many politically-related predicates and politically-related frames, all relatively unrelated to the concept of NATURAL_FEATURES. ENTITY is a frame element of the ORIGIN frame, and it appears in AGING and EVENTIVE_AFFECTING frames – both are also only loosely related to the concept of an ORIGIN. Frames that perform better under the corpus-based model, then, have frame elements and related frames that are more spread out and distantly connected in the graph, so it is perhaps not too sur-

Top 20 frames in graph- versus corpus-based frame embedding models. Frames in **bold** are not in the top 20 of the other model, and thus have benefited from knowledge found in either the graph (Table 2) or corpus (Table 3).

Top Graph-based Frames	G-Rank	C-Rank
BUILDINGS	1	2
QUANTITY	2	3
PEOPLE	3	4
VEHICLE	4	5
AWARENESS	**5**	**269**
KINSHIP	6	6
ASSISTANCE	7	7
INCREMENT	8	8
TIME_VECTOR	**9**	**279**
POLITICAL_LOCALES	10	9
ROADWAYS	11	11
KILLING	12	12
IMPORTANCE	13	13
WEAPON	14	14
INTENTIONALLY_ACT	15	17
ECONOMY	16	18
BUILDING	17	19
GOAL	**18**	**400**
DISCUSSION	19	20
DEPARTING	**20**	**21**

Table 2: Top graph embedding-based frames with the graph model rank (**G-rank**) and the corpus model rank (**C-Rank**)

Top Corpus-based Frames	C-Rank	G-Rank
LOCATIVE_RELATION	**1**	**269**
BUILDINGS	2	1
QUANTITY	3	2
PEOPLE	4	3
VEHICLE	5	4
KINSHIP	6	6
ASSISTANCE	7	7
INCREMENT	8	8
POLITICAL_LOCALES	9	10
ORIGIN	**10**	**325**
ROADWAYS	11	11
KILLING	12	12
IMPORTANCE	13	13
WEAPON	14	14
NATURAL_FEATURES	**15**	**286**
LEADERSHIP	**16**	**277**
INTENTIONALLY_ACT	17	15
ECONOMY	18	16
BUILDING	19	17
DISCUSSION	20	19

Table 3: Top corpus embedding-based frames with the corpus model rank (**C-Rank**) and the graph model rank (**G-rank**)

prising that under these conditions the graph embeddings do not help in prediction.

9 Conclusion

Our results here demonstrate that neural networks can achieve a significant boost when combining representations learned from corpus data with representations learned from knowledge graphs. Many frames that perform poorly in pure corpus-based models improve in the graph-based models. The graph-based model seems to learn better when there is a large set of domain-specific knowledge extracted from the frame's graph. Alternatively, corpus models provide benefits to frames whose graph structures are more diffuse, suggesting the corpus knowledge is better at helping the model to narrow down the sense of the predicate by using the contextual cues found in the annotated data. By combining both embedding types, we achieve strong gains where the combined model can draw advantages from both graph- and corpus-based embeddings.

Acknowledgements

Alexander Popov's contribution has been supported by the Bulgarian Ministry of Education and Science under the National Research Programme "Young scientists and postdoctoral students" approved by DCM # 577 / 17.08.2018.

References

Eneko Agirre, Montse Cuadros, German Rigau, and Aitor Soroa. 2010. Exploring knowledge bases for similarity. In *LREC*.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics* 40(1):57–84.

Collin Baker and Michael Ellsworth. 2017. Graph methods for multilingual framenets. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*. pages 45–50.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 86–90.

Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly Sergieh, and Stefan Roth. 2018. Multimodal frame identification with multilingual evaluation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 1481–1491. https://doi.org/10.18653/v1/N18-1134.

Teresa Botschen, Hatem Mousselly Sergieh, and Iryna Gurevych. 2017. Prediction of frame-to-frame relations in the framenet hierarchy with frame embeddings. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*. pages 146–156.

Volha Bryl, Sara Tonelli, Claudio Giuliano, and Luciano Serafini. 2012. A novel framenet-based resource for the semantic web. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, New York, NY, USA, SAC '12, pages 360–365. https://doi.org/10.1145/2245276.2245346.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. pages 708–716.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics* 40(1):9–56.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. Association for Computational Linguistics, pages 948–956.

Maddalen Lopez De Lacalle, Egoitz Laparra, and German Rigau. 2014. Predicate matrix: extending semlink through wordnet mappings. In *LREC*. pages 903–909.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*. pages 1606–1615. http://aclweb.org/anthology/N/N15/N15-1184.pdf.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the 2015 conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies*. pages 1434–1439.

Yuhang Guo, Wanxiang Che, Ting Liu, and Sheng Li. 2011. A graph-based method for entity linking. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. pages 1010–1018.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet semantic role labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, Valencia, Spain, pages 471–482. https://www.aclweb.org/anthology/E17-1045.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1448–1458.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. http://arxiv.org/abs/1301.3781.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2010. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics.

Martha Palmer. 2009. Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the generative lexicon conference*. GenLex-09, Pisa, Italy, pages 9–15.

Hao Peng, Sam Thomson, Swabha Swayamdipta, and Noah A. Smith. 2018. Learning joint semantic parsers from disjoint data. *CoRR* abs/1804.05990. http://arxiv.org/abs/1804.05990.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Alexander Popov. 2017. Word sense disambiguation with recurrent neural networks. In *Proceedings of the Student Research Workshop associated with RANLP*. pages 25–34.

Alexander Popov. 2019. Lexical modeling for natural language processing. In *Selected papers from the CLARIN Annual Conference 2018, Pisa, 8-10 October 2018*. Linköping University Electronic Press, 159, pages 147–160.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, NATIONAL UNIVERSITY OF IRELAND, GALWAY.

Josef Ruppenhofer, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk. 2006. Framenet ii: Extended theory and practice .

Jennifer Sikos and Sebastian Padó. 2018. Using embeddings to compare framenet frames across languages. In *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*. pages 91–101.

Jennifer Sikos and Sebastian Padó. 2019. Frame identification as categorization: Exemplars vs prototypes in embeddingland. In *International Conference on Computational Semantics(IWCS)*.

Know Your Graph. State-of-the-Art Knowledge-Based WSD

Alexander Popov and **Kiril Simov** and **Petya Osenova**
IICT, Bulgarian Academy of Sciences
Sofia, Bulgaria
{alex.popov|kivs|petya}@bultreebank.org

Abstract

This paper introduces several improvements over the current state of the art in *knowledge-based* word sense disambiguation. Those innovations are the result of modifying and enriching a knowledge base created originally on the basis of WordNet. They reflect several separate but connected strategies: manipulating the shape and the content of the knowledge base, assigning weights over the relations in the knowledge base, and the addition of new relations to it. The main contribution of the paper is to demonstrate that the previously proposed knowledge bases organize linguistic and world knowledge suboptimally for the task of word sense disambiguation. In doing so, the paper also establishes a new state of the art for knowledge-based approaches. Its best models are competitive in the broader context of supervised systems as well.

1 Introduction

Word sense disambiguation (WSD) is a longstanding task in natural language processing (NLP), which has been approached through several broad families of computational techniques. While supervised learning models typically achieve the highest accuracy scores, problems related to the lack of gold corpora, data sparseness and suboptimal granularity of the computational lexicons have plagued the field and prevented significant breakthroughs. This paper presents results achieved with *knowledge-based word sense disambiguation* (KBWSD) algorithms as an alternative pathway. It builds on previous work in the subfield and demonstrates that KBWSD can achieve accuracy scores near and in some cases even at the state-of-the-art, as a rule dominated by supervised approaches. Moreover, the experimental results indicate that in-depth analysis of knowledge representation and knowledge enrichment hold significant promises – both as an alternative to supervised WSD, able to sidestep data-related issues, and as a source of potentially powerful new training signals.

The contribution of the article is fourfold. Firstly, it presents a novel strategy for constructing knowledge bases used in KBWSD, showing that a structure centered on word senses rather than on synsets can be more effective for this particular task. Secondly, another novel approach to structuring the graph is explored in using word and synset embedding models in order to assign weights to relation arcs in the knowledge graph. Thirdly, a number of avenues for the enrichment of the semantic network used for KBWSD are pursued, thus linking WordNet to external resources. Finally, a new state of the art *for KBWSD* is established, which is competitive even when compared with supervised systems.

The paper is structured as follows: the next section presents related work; section 3 deals with some preliminary experiments aimed at replicating previous work described in the literature, but in a slightly modified setting; section 4 outlines several strategies for improving the structure of the knowledge base used for KBWSD; the penultimate section reports on our core experimental work, and the final section concludes the paper.

2 Related Work

State-of-the-art results in the broader field of WSD have been recently summarized in the Unified Evaluation Framework[1] (UEF) by Raganato et al. (2017a), which focuses on the *all-words*

[1] http://lcl.uniroma1.it/wsdeval

disambiguation task based on WordNet 3.0 (Fellbaum, 2012). The top-performing systems in UEF, across supervised and knowledge-based ones, are an SVM model – an extension of the popular IMS system (Zhong and Ng, 2010), and a recurrent neural network model – context2vec (Melamud et al., 2016), with F1 scores on the concatenation of all evaluation corpora ranging between 69% and 69.7%. The supervised models are trained either on the SemCor corpus (Miller et al., 1993) or jointly on SemCor and the semi-automatically constructed OMSTI corpus (Taghipour and Ng, 2015). Very recently deep learning approaches have produced results above the 70% threshold (Luo et al., 2018a,b).

KBWSD has attracted a lot of interest from researchers over the years, since, at least notionally, it does not require any training data or additional resources beyond a computational lexicon and in some cases a knowledge base (KB). Furthermore, if the latter resources are designed in such a way so as to be domain-independent, this could provide a big advantage in dealing with data of heterogeneous origins. One of the earliest popular KBWSD methods is due to Lesk (1986); it takes the dictionary definitions of word senses for target words that occur together in a shared context and calculates the degree of overlap between them, seeking to maximise the latter metric. KBWSD systems, however, typically perform less well than supervised models, due to a number of hurdles, such as: the non-trivial issue of how to structure a KB and what to put inside it; how to explore a KB most effectively; how to integrate various pieces of knowledge into a holistic representation of meaning.

One of the most successful KBWSD approaches has been to use algorithms from the *Random Walks on Graph* family in order to obtain sense representations over particular textual contexts. For instance, Mihalcea (2005) constructs a subgraph with the possible word senses in a context and then runs *PageRank* (Page et al., 1999) over it in order to calculate the most prominent senses. Agirre and Soroa (2009) present an influential update on this method, within the UKB tool for WSD[2]. In addition to the *static* version of PageRank (Spr; introduced in Agirre and Soroa (2008)), which also constructs a sub-graph from the WordNet semantic network as a preliminary step, they put forward *Personalized PageRank*, in which the whole KB is used, with the context words serving to inject probability mass into all candidate word senses. The final state of the PageRank vector over the graph indicates which are the most relevant concepts in the particular context. Two variants are described: *Ppr* and *Ppr_w2w*, where the second one is modified so as to put additional emphasis on connected concepts and away from the target senses themselves. The second strategy is used to prevent competing but related word senses from bolstering each other inordinately, but also makes the algorithm significantly slower. Another related KBWSD system is *Babelfy*, which uses a *Random Walk with Restart* algorithm (Moro et al., 2014).

Two discussion points regarding Random Walks on Graph approaches are central in the context of this article: *knowledge base enrichment* and *algorithm parametrization*. Simov et al. (2016) have shown that the addition of new relation sets to the baseline WordNet 3.0 semantic network can have significant positive effects on the performance of the PageRank algorithm. Adding relations extracted from the manually disambiguated word sense glosses, for instance, is a major improvement; including dependency-based relations between manually disambiguated words from SemCor has also led to big error reductions. Therefore the enrichment and structural optimization of the KBs is clearly one possible avenue for the improvement of KBWSD accuracy. On the issue of optimal parameters selection, Agirre et al. (2018) have proposed an updated default parametrization for using the UKB system for PageRank KBWSD. They consider various possible changes: changing the length of the context under consideration, the number of iterations, the value of the damping factor, the use of word sense frequency information from WordNet, etc. This new default configuration yields much higher accuracy scores on the UEF evaluation data sets – in fact those are the best reported results for a KBWSD system that we are aware of. They are significantly above the challenging most frequent sense (MFS) baseline, and are not very far even from the accuracy scores reported for supervised systems.

3 Preliminary Experiments

Following the impressive improvements achieved via parameter optimization in Agirre et al. (2018)

[2] http://ixa2.si.ehu.es/ukb/

and the observations by Simov et al. (2016) that KB extension can lead to significant improvements over using just the baseline WordNet relations, we undertook to combine the contributions of these two lines of research. In these preliminary experiments, version 3.2 of the UKB system has been used, which is set by default to the parameters described in Agirre et al. (2018). The immediate aim of the experimentation was to ascertain whether this optimized parametrization generates analogous positive effects when an extended KB is used in conjunction with the PageRank algorithm.

We have performed an exhaustive combination and evaluation of the relation sets presented in Simov et al. (2016). In several cases we have been able to obtain improvements over the results reported in Agirre et al. (2018). In addition to reusing these sets of relations, we have decided to further enrich the knowledge graph by assigning weights to the relation arcs, making use of a synset embedding model in order to calculate node similarities. We have constructed such a model following the methodology described in Goikoetxea et al. (2015): using the UKB system's random walk function to produce an artificial corpus of node sequences, then training a Skip-gram model (Mikolov et al., 2013) over it. Thus, for each relation in the KB of the kind `(synset_1, synset_2)` its weight is set to the value of the cosine similarity function between the two synsets. The best results for the two types of setting – without and with weights – are reported in Table 1, in lines 9 and 10 respectively. Merely reusing the additional relation sets leads to an improvement of 0.9%. Setting the relation weights via the embedding model produces further gain of 0.4%.

4 Improving the Knowledge Base

In this section we present further changes to the knowledge base in order to improve accuracy. These modifications can be summed up as follows:

- Changing the shape of the graph by representing nodes in terms of word senses
- Defining weights on the knowledge graph arcs via embedding models
- Extraction of new relations from different external sources

We present each of these in turn.

4.1 Sense-Based vs. Synset-Based WSD

When constructing the KB, the UKB system generalizes lexical meaning in a number of ways. One of these is the use of synset IDs as nodes in the semantic network. Here is an example of a relation as represented in the KB distributed together with the UKB system:

```
u:00007846-n v:04618781-n s:derivation
```

where `u:00007846-n` and `v:04618781-n` represent two nodes in the KB and where there exists a relation between them. The source of this connection is a derivational relation encoded in WordNet. `00007846-n` is the identifier[3] for the synset including the following senses[4]:

```
person%1:03:00::, individual%1:03:00::,
someone%1:03:00::, somebody%1:03:00::,
mortal%1:03:00::, soul%1:03:00::
```

The second node corresponds to the following synset:

```
personhood%1:07:00::
```

As made apparent by this example, the derivational relation holds only between two of these senses[5]:

```
u:person%1:03:00::-n
        v:personhood%1:07:00::-n
                         s:derivation
```

We have exploited this original representation in order to construct the new KB format. The biggest open question we have been faced with is how to represent the synonymy relation. The first option considered was representing it in the same way as in the WordNet distribution:

```
00007846-n person%1:03:00::-n
00007846-n individual%1:03:00::-n
00007846-n someone%1:03:00::-n
00007846-n somebody%1:03:00::-n
00007846-n mortal%1:03:00::-n
00007846-n soul%1:03:00::-n
```

This approach was our first choice. The relation representation in the KB follows the distinction between *lexical relations* holding among senses and *semantic relations* holding among synsets. After several experiments with this new representation of the KB it became clear that the novel format does not lead to improved accuracy.

Subsequently, we have excluded the synset identifiers from the representation of the KB. In

[3] This identifier is formed out of the original identifiers in WordNet. The original part-of-speech prefix is deleted and a part-of-speech suffix is added in its place.

[4] A sense in WordNet is defined as a combination of a lemma and a concept represented by a synset.

[5] This is how derivational and other kinds of lexical relations are represented in the original distribution of WordNet.

order to do this, a decision had to be made on how to represent the semantic and the synonymy relations. The option to represent them via calculating complete Cartesian products seems unrealistic. Hence our decision to utilize another source of information encoded in the distribution of WordNet – the ordering of the senses within each synset, which are supposed to represent their relative lexicographic importance. The main node for each synset is thus represented by the first sense in it. All remaining senses are mapped to this node, which is taken to be representative for the synset. All the semantic relations are mapped to the central nodes as well. The lexical relations are represented as before – between the corresponding word senses.

The above synset is now represented as follows:

```
person%1:03:00::-n individual%1:03:00::-n
person%1:03:00::-n someone%1:03:00::-n
person%1:03:00::-n somebody%1:03:00::-n
person%1:03:00::-n mortal%1:03:00::-n
person%1:03:00::-n soul%1:03:00::-n
```

The sense-centric experiments reported in the article have been performed using this format of the KB.

The next step in transforming the KB is the construction of a set of relations extracted from external sources, such as the ones described in Simov et al. (2016). Included in those is the set of external relations distributed together with the UKB system. This set is constructed on the basis of a WordNet gloss corpus, in which some of the words used in the gloss for each synset are annotated with the appropriate word senses from WordNet. Each constructed relation holds between the synset associated with the gloss and the synset of the corresponding sense in the annotation. It is relatively easy to reproduce this type of relation on the basis of the gloss corpus.

The relations represented in the **GraphRelSC** set are extracted from the SemCor corpus. This set of relations includes two types of nodes: for the semantically annotated words and for the nodes in the dependency tree of the corresponding sentence. The dependency analyses of the sentences are not part of the original annotation of SemCor. Thus, those cannot be reused for the construction of sense-based relations. The good news is that we do not need to have the actual dependency annotation, because in **GraphRelSC** the nodes corresponding to the dependency nodes are numbered with the numerical IDs of the documents, the sentences, and the words in the original corpus. Thus, through a simple mapping we have been able to substitute the synset identifiers with the actual sense annotations in the corpus. In Table 1 this new set of relations is referred to as **SC**.

In addition to **GraphRelSC**, we have extracted a set of co-occurrence relations from each sentence in SemCor. This new set of relations is called **SCR**. We have not been able to reconstruct the **WN30gl** set of relations because the mapping from it to the original annotation is not straightforward. Similarly, we have decided not to use the third set of relations, **WN30glCon**.[6]

4.2 Relation Weighting

Each relation in the KB can be assigned an individual weight. These weights are exploited within the ranking algorithms implemented in the UKB system. The original sets of relations do not assign any weights to the arcs. In our preliminary experiments we have assigned each relation a weight determined by the similarity of its associated nodes. This similarity is derived by calculating the cosine similarity of the vector representations for the corresponding nodes.

In order to assign weights, then, appropriate embedding models are necessary to provide vector representations. In the case of the preliminary experiments, we have used a synset embedding model constructed via random walks along the WordNet KB[7]. This is not the case with the new format of the KB where the nodes are represented as senses in WordNet. A direct construction of sense embeddings is also an option, which has not been realized for the purposes of this paper, and will be addressed in future research. Here we use pretrained embeddings.

Each word sense in WordNet is connected to a lemma and a synset. Thus, we could use either the synset embeddings, or the lemma (word) embeddings, or some combination between them. The first option has been disregarded since it does not distinguish between the various senses in the synonym sets. We have performed experiments with pretrained word embeddings such as GoogleNews[8] and Glove[9] (the 300-dimensional version),

[6] All new sets of relations, as well as the lexicon, available at `http://bultreebank.org/en/resources/`.

[7] All embedding models referred to here are also made accessible at `http://bultreebank.org/en/resources/`.

[8] `https://code.google.com/archive/p/word2vec/`

[9] `https://nlp.stanford.edu/projects/`

also with lemma embeddings trained analogously to the synset embeddings (see Goikoetxea et al. (2015) on how to generate pseudo corpora from WordNet and subsequently train embedding models on them).

For instance, for the sense `person%1:03:00::-n`, the associated list of synset and lemma IDs includes `00007846-n` and `person`. In the case when only a lemma embedding is available, the corresponding vector is used. In the case of multi-word expressions like `physical_object%1:03:00::-n`, the average of the vectors for the different component words is calculated. In the case when there are both a synset and a lemma embedding available, the concatenation of the two vectors is considered. The following embeddings have been used in the current work:

- **GoogleNews.** A word embedding model trained over 100 billion running words. The vectors are of size 300.
- **Glove.** Word embeddings trained over global contexts, as described in Pennington et al. (2014).
- **WN30WN30glConOne**. Synset and lemma embeddings trained by Simov et al. (2017).
- **WN30WN30glConOneWiki**. In this case a lemmatized Wikipedia corpus has been added to the pseudo corpus, in order to balance information from the knowledge graph with actual text data.

The weights associated with the different word senses in the new lexicon, which also play an important role for the optimal performance of the UKB system, are the same as those associated with synset identifiers in the original lexicon in the UKB distribution. This is so because the frequencies have been originally determined on the basis of sense occurencies in "various semantic concordance texts" (according to the documentation, quoted in Agirre et al. (2018)). Consequently, the conjunction of lemma and synset ID in the lexicon provides a unique mapping to a singular word sense, regardless of the structure of the KB.

4.3 Linking WordNet to VerbNet and FrameNet

VerbNet (Schuler, 2005) and FrameNet (Baker et al., 1998) are structured lexical resources which provide information that is, from a theoretical point of view, complementary to that within WordNet. While the latter's semantic network is a rich representation of lexical semantics, the former give more insight into sentential semantics.

VerbNet classes bring together verbs that share the same syntactic subcategorization and semantic valency patterns. Membership in a VerbNet class does not necessarily indicate that the lexical items have a similar meaning (although that is often the case), but that they share some kind of structurally analogous behaviour, which is certainly a kind of information that is not present in WordNet. For instance, the verb *buy* is in the same class as *hire, lease, rent*, but in the same class are also the verbs *catch, choose, pluck, slaughter*[10].

FrameNet organizes lexical knowledge around particular procedural scenarios called *frames* (Fillmore, 1968). An example of a FrameNet frame would be *Commerce_buy*[11], which specifies *frame elements*, i.e. participants and specifications of the situation, such as *Buyer*, *Goods*, *Seller*, *Means*, *Money*, etc. A frame is activated by certain *lexical units*, in this case *buy, buyer, client, purchase (noun), purchase (verb), purchaser*. Frames can be linked with one another, through relations like *Inherits from, Is used by*, etc.

As part of the effort to complement already existing relation sets for KBWSD, the WordNet semantic network has been partially connected to those of VerbNet and FrameNet. The Predicate Matrix resource[12] (De Lacalle et al., 2014, 2016), which automatically maps the WordNet, VerbNet, FrameNet, PropBank and MCR indices, has been used to obtain most of the cross-mappings; in the case of VerbNet, some of the verbs have already been mapped to WordNet senses, but Predicate Matrix can be used to extend the coverage of the mapping. In this way all WordNet sense identifiers that could be mapped to predicates in VerbNet classes and FrameNet frames have been organized in structures that reflect this kind of membership in the external resources. This has been done in a way that is similar to the one described earlier in relation to graphically connecting the various word senses in a synset. That is, the word sense

`glove/`

[10] `http://verbs.colorado.edu/verb-index/vn/get-13.5.1.php`

[11] The frame index is available at `https://framenet.icsi.berkeley.edu/fndrupal/frameIndex`.

[12] `http://adimen.si.ehu.es/web/PredicateMatrix`

N	Knowledge base	SNE-2	SNE-3	SME-07	SME-13	SME-15	ALL
1	WN$_{\text{synsets}}$	64.6	62.8	51.6	67.9	66.6	64.3
2	WN$_{\text{senses}}$	66.5	61.2	51.6	66.5	70.5	64.8
3	WN+VNM$_{\text{senses+w}}$	67.6	61.4	53.2	65.9	71.6	65.3
4	WN+VNM+FNM$_{\text{senses+w}}$	67.4	62.4	53.6	66.0	71.4	65.5
5	WN+VNM+FNM+FNR$_{\text{senses+w}}$	67.1	62.7	54.3	65.8	71.1	65.4
6	WN+GL$_{\text{synsets}}$(Agirre et al., 2018)	68.8	66.1	53.0	68.8	70.3	67.3
7	WN+GL$_{\text{senses}}$	69.0	65.7	55.4	69.2	71.8	67.7
8	WN+GL$_{\text{senses+w}}$	69.3	66.0	55.2	69.4	71.6	67.9
9	WN+GL+SC$_{\text{synsets}}$	70.1	67.0	53.0	69.5	70.9	68.2
10	WN+GL+SC$_{\text{synsets+w}}$	**70.4**	67.6	53.4	69.5	71.9	68.6
11	WN+GL+SC$_{\text{senses}}$	70.1	67.8	57.4	69.0	72.2	68.8
12	WN+GL+SC$_{\text{senses+w+ctx=20,35}}$	70.2	67.8	**58.2**	69.1	72.4	68.9
13	WN+GL+SC+VNM+FNM$_{\text{senses+w}}$	69.7	67.6	57.4	68.6	**72.5**	68.5
14	WN+GL+SC$_{\text{senses+w+ctx=10,15,25,30}}$	70.3	**67.9**	57.8	**69.8**	71.8	**69.0**

Table 1: Accuracy scores on the UEF data sets with different KBs. Only the results for the Ppr_w2w mode of the PageRank algorithm are reported. The synset-based models use KBs where the nodes are represented by synset IDs in WordNet; sense-based models use the new KB configurations described in the paper; the *+w* subscript means that a model takes into account relation weights. **WN** stands for the original WordNet relations; **GL** – the relations from the annotated gloss corpus; **SC** – the relations from the automatically parsed SemCor corpus; **VNM** – the sense groupings from VerbNet; **FNM** – the sense groupings from FrameNet; **FNR** – the links between FrameNet predicate senses and role-type senses. All experiments use the default parametrization from Agirre et al. (2018), with the exception of the cases marked with a subscript *ctx=num*, where the context windows have been changed to include *num* words. The best result (line 14) is achieved for contexts with 10, 15, 25 and 30 words. **SNE** stands for *Senseval* and **SME** stands for *SemEval*.

WSD system	SNE-2	SNE-3	SME-07	SME-13	SME-15	ALL
Luo et al. (2018a)	**72.8**	70.3	—*	68.5	**72.8**	**71.1**
Luo et al. (2018b)	72.2	**70.5**	—*	67.2	72.6	70.6
Raganato et al. (2017b)	72.0	69.1	**64.8***	66.9	71.5	69.9
Iacobacci et al. (2016)†	73.3	69.6	61.1	66.7	70.4	69.7
Melamud et al. (2016)†	72.3	68.2	**61.5**	67.2	71.7	69.4
Agirre et al. (2018)†	68.8	66.1	53.0	68.8	70.3	67.3
Moro et al. (2014)†	67.0	63.5	51.6	66.4	70.3	65.5
WN 1st sense†	66.8	66.2	55.2	63.0	67.8	65.2
This work$_{\text{best}}$	70.3	67.9	57.8	**69.8**	71.8	69.0

Table 2: The state of the art across WSD systems. The dagger symbol indicates that the result is reported in the UEF (Raganato et al., 2017a). * Luo et al. (2018a,b) do not report accuracy on SME07 since it is used as a development set; this is also true for Raganato et al. (2017b), which however does report the result. The KBWSD models in the table (ours and those by Agirre and Moro) are deterministic, i.e. they would always produce the same results with particular KBs, as no actual training is involved.

in a class/frame with the highest associated frequency of use is promoted to main node status that stands in for the whole structure, then all the rest of the senses are connected to it by setting a connection weight according to their respective frequencies (including a +1-count smoothing). The verb class hierarchy and inter-frame connection relations are also included in the new subgraph.

Additionally, the automatically created FrameNet extension[13] by Bryl et al. (2012), which

[13] `https://dh.fbk.eu/technologies/wordnet-sense-repository-framenet-extension`

maps FrameNet roles to WordNet synsets, has been used to add links between predicate word senses grouped in frames and role-filling word senses. These new relations are weighted in accordance with the numbers provided by the generated sense repository, which correspond to frequencies of use in an automatically tagged corpus. The procedures described above should provide higher inter-predicate connection density (the class/frame membership relations) in the KB and also more syntagmatically oriented relations (the FrameNet role senses mapped to frame lexical unit senses).

5 Experiments

In this section we provide several empirical points of view to the currently presented project of extending and optimizing the knowledge graph used for WSD. First we examine how the reconfiguration of the graph in terms of word senses compares with the synset-based graph and show that the new structure outperforms the previous one. In parallel with that we demonstrate that the introduced KB extensions lead to significant improvements over the baseline graphs. We also provide our own contribution to the optimal parametrization of the UKB system.

KB permutations Table 1 shows the various combinations of KB relations, KB structuring and parametrization of the UKB system. Several noteworthy observations come to the fore:

1. By comparing lines 1&2, 6&7, 9&11, it becomes clear that the KBs structured around word senses perform better than those where nodes are represented by synset IDs.
2. The addition of relations extracted from the VerbNet-WordNet-FrameNet (gold and automatic) mappings does improve the baseline results over the WN relations (lines 3-5). VNM builds over WN accuracy, FNM builds over WN+VNM, while FNR does not seem to decisively improve results (though on some data sets it does help – SNE-3 and SME-07).
3. The addition of the gloss and SemCor relations has a very significant effect on accuracy when compared to using just the baseline WN relations (lines 6-14).
4. The VN-FN relations do not seem to reliably improve accuracy when added to the gloss and SemCor relations, in fact they improve accuracy only on one data set (SME-15) and in the rest of the cases bring it down (line 13).
5. The default parameters from Agirre et al. (2018) are indeed a good optimization of the UKB system. We have been able to improve the result with the best KB only in one case, which we report here, and the improvement is not very big (0.1%; line 14). The result nevertheless indicates that there is space for optimizing the interaction between KB and algorithm.

Comparison with state-of-the-art models Table 2 situates our best result in the context of the state-of-the-art results in WSD at large. Again, several observations are worth pointing out:

1. The combination of UKB with our best-performing graph comfortably beats the WN 1st sense heuristic, which is not the case for many WSD systems.
2. The model significantly outperforms all KB-WSD models reported in the UEF, including the improved parametrization of Agirre et al. (2018), whose results we have improved upon with 1.7%.
3. Our result is at this point very close to the top-performing WSD systems, regardless of whether they are supervised or not. This has typically not been the case for KBWSD systems. For one data set (SME-13) the present model achieves the highest result of all[14] and for another one (SME-15) it is better than all but the two leading supervised systems.

Improvements with *static PageRank* Finally, in table 3 we show the performance of three different versions of the PageRank algorithm, as implemented in the UKB system and described in Agirre and Soroa (2009). The results with the static version have been very significantly improved in comparison to the results reported in Agirre et al. (2018). The static mode performs better than the WN 1st sense heuristic when used with the specified graphs. The difference between the *w1* and *w2* models is as follows: with *w1* weights are set via a combination of **WN30WN30glConOne** embeddings for synsets and lemmas, and with *w2* – via a combination of

[14] SME-13 contains sense annotations of nouns only; with its extensive taxonomic network, WordNet is a powerful tool for representing nominal meaning. More detailed analysis is required to ascertain the real reasons for the high accuracy scores.

Knowledge base	**Spr**	**Ppr**	**Ppr_w2w**
WN+GL$_{\text{synsets}}$(Agirre et al., 2018)	57.7	**65.6**	67.3
WN+GL+SC$_{\text{senses+w1}}$	65.5	65.2	**68.9**
WN+GL+SC$_{\text{senses+w2}}$	66.5	65.1	68.0
WN+GL+SC+SCR$_{\text{senses+w2}}$	**66.6**	65.1	68.1

Table 3: Comparison between different PageRank versions (accuracy measured on all UEF test data sets). **Spr** stands for *static* mode, **Ppr** is *Personalized PageRank*, **Ppr_w2r** is *Ppr with emphasis on neighbouring concepts*. *w1* denotes weights set via the combination of **WN30WN30glConOne** embeddings for synsets and lemmas; *w2* denotes the combination of **WN30WN30glConOneWiki** embeddings.

WN30WN30glConOneWiki embeddings, again for synsets and for lemmas. The **SCR** relation set, which is built on the basis of co-occurrences of word senses in SemCor, contributes positively to the best static model. Compared with previous state-of-the-art results when using static PageRank, here we can see an improvement of nearly 9%.

Bearing in mind that the static version of the algorithm is much faster than the personalized ones, these results should also be interpreted as important, as they demonstrate that KB improvement might be even more beneficial for less sophisticated methods that nevertheless offer a good tradeoff in terms of speed of execution.

6 Conslusion

We have presented results from a series of experiments with a KBWSD system with state-of-the-art default parametrization and have shown that accuracy can be further improved through the manipulation and extension of the KB. The present models achieve the highest reported accuracy scores for a KBWSD system that we are aware of; they also enter in the close orbit of the highest-scoring supervised systems, achieving a new state of the art on the Semeval-13 data set.

Further improvement to the content of the KB is certainly possible. Based on the reported experiments, an intuition emerges that the relation weighting schema has to have a dynamic character. This would correspond to the promotion and demotion of semantic features within context. The current implementation relies on the PageRank algorithms to maintain this kind of dynamics. The experiments also demonstrate that the different weighting schemata improve the performance of different algorithms, which suggests complex patterns of interaction between algorithm and graph structure. The current schemata do not distinguish between different kinds of relations. The difference between paradigmatic and syntagmatic relations necessitates different weighting approaches. Using the cosine similarity measure over the entire embedding space seems to be a suboptimal blanket strategy. In our future research we plan to train relation embeddings, following the approach for generating pseudo corpora, as in the case of training synset and lemma embeddings.

Additionally, we plan on further investigating strategies for generalizing knowledge from external resources such as VerbNet and FrameNet, as well as other ones which can be mapped to WordNet, such as PropBank (Kingsbury and Palmer, 2002) and the OntoNotes sense groupings (Snow et al., 2007). As has been demonstrated, structuring a lexico-semantic graph in an optimal way can make a big difference. Detailed error analysis and sophisticated linguistic theory should be employed in order to capture the principles underlying a good knowledge base.

Acknowledgements

This research has been funded by the Bulgarian National Science Fund grant number 02/12/2016 — *Deep Models of Semantic Knowledge (DemoSem)*. Alexander Popov's contribution has been also supported by the Bulgarian Ministry of Education and Science under the National Research Programme "Young scientists and postdoctoral students" approved by DCM # 577 / 17.08.2018.

References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2018. The risk of sub-optimal use of open source NLP software: UKB is inadvertently state-of-the-art in knowledge-based WSD. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 29–33. Association for Computational Linguistics.

Eneko Agirre and Aitor Soroa. 2008. Using the multilingual central repository for Graph-based Word Sense Disambiguation. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA). Http://www.lrec-conf.org/proceedings/lrec2008/.

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 33–41, Athens, Greece. Association for Computational Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley Framenet Project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*, COLING '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Volha Bryl, Sara Tonelli, Claudio Giuliano, and Luciano Serafini. 2012. A novel framenet-based resource for the semantic web. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 360–365, New York, NY, USA. ACM.

Maddalen Lopez De Lacalle, Egoitz Laparra, Itziar Aldabe, and German Rigau. 2016. Predicate Matrix: automatically extending the semantic interoperability between predicate resources. *Language Resources and Evaluation*, 50(2):263–289.

Maddalen Lopez De Lacalle, Egoitz Laparra, and German Rigau. 2014. Predicate Matrix: extending SemLink through WordNet mappings. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Christiane Fellbaum. 2012. Wordnet. *The Encyclopedia of Applied Linguistics*.

Charles J. Fillmore. 1968. The case for case. In E. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart, and Winston.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *HLT-NAACL*, pages 1434–1439. The Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA).

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

Fuli Luo, Tianyu Liu, Zexue He, Qiaolin Xia, Zhifang Sui, and Baobao Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411. Association for Computational Linguistics.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. 2018b. Incorporating Glosses into Neural Word Sense Disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2473–2482. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Rada Mihalcea. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 411–418, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A Semantic Concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 303–308, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word Sense Disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 99–110.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural Sequence Learning Models for Word Sense Disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Kiril Simov, Petya Osenova, and Alexander Popov. 2016. Using Context Information for Knowledge-Based Word Sense Disambiguation. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 130–139, Cham. Springer International Publishing.

Kiril Simov, Petya Osenova, and Alexander Popov. 2017. Comparison of word embeddings from different knowledge graphs. In *Language, Data, and Knowledge*, pages 213–221, Cham. Springer International Publishing.

Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. 2007. Learning to merge word senses. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1005–1014, Prague, Czech Republic. Association for Computational Linguistics.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 338–344, Beijing, China. Association for Computational Linguistics.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. Association for Computational Linguistics.

Are Ambiguous Conjunctions Problematic for Machine Translation?

Maja Popović and Sheila Castilho
ADAPT Centre,
School of Computing, DCU
Ireland
{firstname.lastname}@adaptcentre.ie

Abstract

The translation of ambiguous words still poses challenges for machine translation. In this work, we carry out a systematic quantitative analysis regarding the ability of different machine translation systems to disambiguate the source language conjunctions "but" and "and". We evaluate specialised test sets focused on the translation of these two conjunctions. The test sets contain source languages that do not distinguish different variants of the given conjunction, whereas the target languages do. In total, we evaluate the conjunction "but" on 20 translation outputs, and the conjunction "and" on 10. All machine translation systems almost perfectly recognise one variant of the target conjunction, especially for the source conjunction "but". The other target variant, however, represents a challenge for machine translation systems, with accuracy varying from 50% to 95% for "but" and from 20% to 57% for "and". The major error for all systems is replacing the correct target variant with the opposite one.

1 Introduction

Ambiguous words are often difficult to translate automatically, even by the state-of-the-art neural machine (NMT) systems. Whereas the NMT approach significantly improved fluency (grammar) of MT outputs compared to the previous state-of-the-art statistical phrase-based (PBMT) models, adequacy (meaning preservation) is still often problematic (Castilho et al., 2017; Klubička et al., 2018). Adequacy is even more problematic for ambiguous words which have two or more meanings depending on the context.

Therefore, ambiguity of nouns, verbs and pronouns has been investigated extensively in recent years (Guillou et al., 2018; Müller et al., 2018; Rios Gonzales et al., 2017, 2018). However, to the best of our knowledge, no results for ambiguity of conjunctions have been reported so far. The only work dealing with conjunctions and machine translation (Huang, 1983) explores conjunction scope for rule-based MT systems and does not address the ambiguity. It should be noted, though, that the conjunction ambiguity is more structural than lexical: it is mainly related to certain aspects of grammar involving the arrangement of words and word types. Therefore, the conjunction ambiguity is related more to fluency than it is to adequacy.

In this work, we present the results of quantitative analysis addressing machine translation of two potentially ambiguous conjunctions, "but" and "and". The analysis of the conjunction "but" is carried out for {English,French}-into-{Spanish,German,Serbian,Croatian} translation directions, and the conjunction "and" is analysed on {English,Portuguese}-into-{Serbian,Croatian} outputs. Evaluation is carried out on specialised test sets[1] designed for evaluating translation of these ambiguous conjunctions. Instead of comparing the translation output with a reference human translation, our evaluation is based on presence or absence of the correct target language conjunction variant in the translation output. For a small number of sentences with both or with none of the target variants (about 1-2%), manual inspection is carried out.

	but_1	but_2
en	We wanted to go to the beach, **but** we went back to the hotel.	We didn't want to go to the hotel, **but** to the beach.
de	Wir wollten zum Strand gehen, **aber** wir sind zurück zum Hotel gegangen.	Wir wollten nicht zum Hotel, **sondern** zum Strand.
es	Queríamos ir a la playa, **pero** hemos vuelto al hotel.	No queríamos ir al hotel, **sino** a la playa.
sr (hr)	Hteli smo da idemo na plažu **ali** smo se vratili u hotel.	Nismo hteli da idemo u hotel **nego** na plažu.
en	She will not come **but** she can call.	She will not come **but** call.
es	No va a venir, **pero** puede llamar.	No va a venir, **sino** a llamar.
de	Sie kommt nicht, **aber** sie kann anrufen.	Sie kommt nicht **sondern** ruft an.
sr (hr)	Neće doći **ali** može da zove.	Neće doći **nego** će zvati.

Table 1: Examples of difference between the two variants of the English conjunction "but".

domain	lang.	but_1	but_2
News	En-De	65.2	34.8
	En-Sr	79.7	20.3
	En-Hr	78.3	21.7
Subtitles	En-De	97.2	2.8
	Fr-De	96.6	3.4
	En-Sr	97.1	2.9

Table 2: Distribution of sentences requiring each of the two target language variants of the source conjunction "but" in different publicly available parallel corpora.

2 Related Work

Lexical ambiguity as a challenge for machine translation has received a lot of attention in recent years. Rios Gonzales et al. (2017) and Rios Gonzales et al. (2018) focus on ambiguous German nouns, while Guillou et al. (2018) and Müller et al. (2018) investigate ambiguous English pronouns. Broader linguistic evaluations presented in Burchardt et al. (2017) and Klubička et al. (2018) also include ambiguity, but conjunctions are not mentioned in any context.

Syntactic ambiguity for rule-based English-to-Bulgarian machine translation is investigated in Pericliev (1984), but these ambiguities are not related to conjunctions. Ambiguity of conjunctions "and" and "or" is investigated for requirement specifications in Sharma et al. (2014) and for legal texts in Adams and Kaye (2006), however without any relation to (either human or machine) translation.

The first work dealing with conjunctions and machine translation is described in Huang (1983). It explores conjunction scope for English parser to be used in rule-based MT systems, but it does not address the ambiguity. Another work related to conjunctions and machine translation is the work of Xu et al. (2014), who proposes using conjunctions for Chinese sentence segmentation in order to achieve better translation quality. Some problems with translating conjunctions by phrase-based machine translation systems involving South Slavic languages are mentioned in Popović and Arčan (2015), but without any systematic quantitative analysis.

To the best of our knowledge, our work represents the first experiments related to ambiguous conjunctions and machine translation. We report the results of an extensive evaluation showing that certain conjunction ambiguities pose a challenge to the state-of-the-art machine translation systems.

3 Ambiguity of "But" and "And"

3.1 Conjunction "But"

In some languages, there are two possible variants of the conjunction "but". One variant, but_1, can be used after either a positive or a negative clause. The other variant, but_2, is used after a negative clause when expressing a contradiction. The first clause in the sentence must contain a negation marker, and the second part of the sentence must contradict the first part of the sentence.

Three examples can be seen in Table 1. The sentences on the left have the same context, same or similar meaning, and contain similar words as

[1] https://github.com/m-popovic/evaluating-ambiguous-conjunctions-MT

	and_1	and_2
en	The walls **and** the door are white.	The walls are white **and** the door is black.
hr (sr)	Zidovi **i** vrata su bijeli.	Zidovi su bijeli **a** vrata su crna.
en	I studied for the whole day **and** I learned a lot.	I studied for the whole day **and** I didn't learn anything.
hr (sr)	Učio sam cijeli dan **i** svašta naučio.	Učio sam cijeli dan **a** ništa nisam naučio.
en	Years passed, and he came back.	Years passed, and he still hadn't come back.
hr (sr)	Prošle su godine **i** on se vratio.	Prošle su godine **a** on se još nije vratio.
en	Who is this and what is he doing here?	And what is he doing here?
hr (sr)	Tko je to **i** što on radi ovdje?	**A** što on radi ovdje?

Table 3: Examples of difference between the two variants of the English conjunction "and".

domain	lang.	and_1	and_2
News	En-Sr	60.0	40.0
	En-Hr	59.4	40.6
Subtitles	En-Sr	62.2	37.8
	Pt-Hr	60.4	39.6

Table 4: Distribution of sentences with two types of conjunctions "and" in different publicly available parallel corpora.

the sentences on the right. Nevertheless, the conjunction "but" in all sentences on the left should be translated into but_1 and in those on the right as but_2. This also illustrates the previously mentioned structural nature of conjunction ambiguity.

Generally, sentences with the first variant, but_1, can be found more frequently in the data. Table 2 presents the distribution of the two types of sentences with the conjunction "but" found in publicly available data for several language pairs in two domains: news and subtitles.

3.2 Conjunction "And"

Some target languages, such as Serbian and Croatian, distinguish two variants of the conjunction "and". The first variant, and_1, is used to connect non-contrasting actions or ideas, for example to indicate that one action follows another in the chronological order, or that one idea is the expected result of another. The second variant, and_2, is used to indicate that the two connected facts are different: it introduces a new or different meaning, that is, it introduces an idea that is different or opposite to the idea that is desired, expected or stated previously. Both variants are used to start a new sentence or clause that continues or adds to a previous sentence or clause, however and_2 is adding some new, different or unexpected facts.

Four examples can be seen in Table 3. Similarly to the examples for the conjunction "but", the sentences on the left have similar meaning and contain similar words as the sentences on the right, but all "ands" on the left should be translated into and_1 and those on the right into and_2.

Table 4 presents the distribution of the two types of conjunction "and" in publicly available news and subtitles data. Again, the first variant, and_1, is more frequent, although the difference is smaller than between the two variants of the conjunction "but".

4 Experimental Set-Up

4.1 Test Sets

In order to estimate a system's capability to translate ambiguous conjunctions, evaluation is performed on specialised test sets specifically designed for the conjunctions "but" and "and" and their two variants.

The test sets are created semi-automatically using the multilingual subtitles corpora[2] (Tiedemann, 2012). Only short segments (up to 20 words) were included, all noise was removed, and rare named entities which could introduce additional effects were avoided or replaced. Thus, about 1000 source sentences in English and in French were prepared for the conjunction "but", and 250 source sentences in English and in Portuguese for conjunction "and". Detailed corpus statistics are presented in Table 5.

It should be noted that although the test sets were created using a bilingual corpus, the resulting test sets do not contain any reference translations. The reason for this is twofold: on the one hand, bilingual manual filtering of noisy and complex

[2] http://www.opensubtitles.org/

(a) Statistics of the test sets for the source conjunction "but"

source language	target conjunction	number of sentences	number of running words	vocabulary size	average sent. length
English	all	1066	13655	2252	12.8
	but_2	858	11058	2043	12.9
	but_1	208	2597	560	12.5
French	all	1010	12963	2162	12.8
	but_2	806	10478	1823	13.0
	but_1	204	2485	673	12.2

(b) Statistics of test sets for the source conjunction "and"

source language	target conjunction	number of sentences	number of running words	vocabulary size	average sent. length
English	all	258	3217	769	12.5
	and_2	206	2651	691	12.9
	and_1	52	566	248	10.9
Portuguese	all	250	2763	908	11.0
	and_2	199	2218	767	11.1
	and_1	51	546	264	10.7

Table 5: Statistics of the test sets for (a) conjunction "but" and (b) conjunction "and": number of sentences, number of running words, vocabulary size and average sentence length.

content would be very time and resource consuming. On the other hand, reference translations are not really needed, because we are interested only in conjunction disambiguation, therefore, checking the conjunction in the translation hypothesis is sufficient.

In order to encourage and enable future research on the topic, the developed test sets are made publicly available.[3]

4.2 MT Outputs

The English and the French test sets for the conjunction "but" were translated into four target languages that distinguish the two variants but_1 and but_2 in the same way, namely Spanish, German, Serbian and Croatian. The English and the Portuguese test sets for the conjunction "and" were translated into Serbian and Croatian. For all translation directions, two publicly available on-line systems, "Google Translate"[4] and "Bing Translator"[5] are used. All on-line translations were generated between 26th and 30th April 2019.

In addition to this, for English-to-German and English-to-Serbian translation, two internal systems trained on much smaller amounts of data from the news domain were available. Two English-to-German systems, one NMT and one PBMT, are trained on one million parallel sentences from the WMT[6] data. Two English-to-Serbian systems, also one NMT and one PBMT, are trained on the SETimes corpus (Tyers and Alperen, 2010) containing about 200k sentence pairs.

In total, the conjunction "but" is analysed on 20 MT outputs, and the conjunction "and" on 10 MT outputs.

4.3 Evaluation

The majority of sentences are checked automatically, however for a small number of sentences a manual inspection is needed. For each sentence, there are four possible outcomes of the automatic evaluation:

- only the correct conjunction is found
  ⇒ correct
- only the opposite conjunction is found
  ⇒ incorrect
- both conjunctions are found
  ⇒ manual inspection

[3] https://github.com/m-popovic/evaluating-ambiguous-conjunctions-MT

[4] https://translate.google.com/

[5] https://www.bing.com/translator

[6] http://www.statmt.org/wmt17/

- none of the two conjunctions are found

  ⇒ manual inspection

Manual inspection is carried out in the following way: if the structure of a sentence with additional or without any conjunctions is correct, then the sentence is considered correct.

All errors which are not related to the conjunction choice are ignored, both by automatic and by manual evaluation.

5 Results

The results for both conjunctions and all language pairs are presented in the form of percentage of sentences automatically identified as correct ("aut."), identified as correct after both automatic check and manual inspection ("full"), and automatically identified as incorrect because the source conjunction is translated into the opposite conjunction ("opposite").

5.1 Conjunction "But"

The results for the source conjunction "but" can be seen in Table 6.

Target variant *but*$_1$: Recognising the target conjunction but_1 is generally not problematic: the percentage of correct sentences is almost 100% for all on-line systems, and close to 100% even for the scarce-data $setimes$ systems. Apparently, there is no correlation between this accuracy and the amount of the training data, because the two wmt systems have lower accuracies than the two $setimes$ systems. These lowest accuracies are still very high, i.e. close to 90%.

Target variant *but*$_2$: For this variant, the situation is, however, different. On-line systems translate it correctly in 73-95% of cases, and the predominant problem for the rest of the cases is translating it into the opposite variant but_1 (5-25%). The four scarce-data systems are struggling much more with this variant, PBMT systems more than NMT ones. In addition, the influence of the amount of data is obvious, since $setimes$ systems are performing much worse than wmt systems. For these four systems, replacing but_2 with but_1 is the most frequent problem too, but there were more sentences requiring manual inspection than for the on-line systems.

Both target variants: Manual inspection revealed that this is generally not problematic for any of the systems and language pairs: it can happen if "however", "yet" or similar words that can be translated as but_1 are present in the source sentence.

None of the two target variants: Only a small number of such sentences are generated by on-line systems when translating from English if the English sentence has a structure "not only X, but Y, too". In these cases, the sentence is paraphrased in the way "not only X, Y too". For the French source, a number of other sentence structures are paraphrased, and the majority of these paraphrases are correct. An example can be seen in Table 7.

As for the scarce-data systems, manual inspection revealed that the wmt NMT system left many sentences untranslated, of which mainly those requiring the conjunction but_2. However, a number of those requiring the conjunction but_1 were also left untranslated. As for PBMT systems, many of the sentences in the output used a possible translation for but_1 such as "however", which is of course not correct for sentences requiring but_2. A number of PBMT sentences had a number of other error types and a low fluency in general. Apart from this, when translating from English into Serbian and Croatian, $Google$ and $Bing$ often translated "but" as "except", which is not a possible option for any of the source sentence structures.

5.2 Conjunction "And"

The results for the source conjunction "and" can be seen in Table 8.

Target variant *and*$_1$: Similarly to the translation of "but", the target variant and_1 is not really problematic for any of the systems, with almost all accuracies close to 100% and all larger than 92%. Also, there is apparently no correlation between this accuracy and the amount of the training data, since the $Bing$ system has the lowest accuracy and it is certainly trained on more data than the two $setimes$ systems.

Target variant *and*$_2$: This target variant is definitely problematic, and much more challenging than but_2: the highest accuracy is 56.3%. Similarly to the conjunction "but", the predominant problem is replacing and_2 with and_1. Also, the accuracies are lower for the scarce-training $setimes$ systems. Nevertheless, one curiosity can be noted: the PBMT $setimes$ system better disambiguates the conjunction "and" than the NMT $setimes$ system. Given that NMT systems are generally more sensitive to the scarcity of training data (Koehn and Knowles, 2017) than PBMT

		but_2			but_1		
language		correct		opposite	correct		opposite
pair	system	aut.	full	(but_1)	aut.	full	(but_2)
en-es	google	93.3	93.4	6.2	100	100	0
	bing	76.5	76.6	22.6	100	100	0
en-de	google	88.6	89.1	10.3	99.5	100	0
	bing	94.3	94.4	5.3	99.0	99.0	1.0
	wmt-pbmt	48.9	48.9	46.8	90.9	92.8	4.8
	wmt-nmt	60.5	60.5	28.6	84.6	84.6	0.5
en-sr	google	90.3	90.3	9.3	99.0	99.5	0.5
	bing	79.6	79.6	18.9	100	100	0
	setimes-pbmt	7.0	7.0	88.3	95.7	100	0
	setimes-nmt	53.5	53.6	43.8	95.7	98.6	1.0
en-hr	google	91.8	91.8	7.7	99.5	99.5	0.5
	bing	73.6	73.6	25.4	100	100	0
fr-es	google	92.6	93.8	6.0	100	100	0
	bing	72.0	75.1	24.1	100	100	0
fr-de	google	87.6	89.0	10.4	100	100	0
	bing	88.5	92.9	6.7	100	100	0
fr-sr	google	90.6	90.9	8.5	100	100	0
	bing	75.2	77.3	21.7	100	100	0
fr-hr	google	90.6	90.9	8.4	100	100	0
	bing	72.0	73.9	24.5	100	100	0

Table 6: Percentage of correct target language conjunctions retrieved automatically and by full evaluation, and percentage of opposite target conjunctions for the source language conjunction "but".

source	Ce n'est pas une étoile **mais** un cristal.
source (en gloss)	It is not a star **but** a crystal.
output (es)	No es una estrella, es un cristal.
output (de)	Es ist kein Stern, es ist ein Kristall.
output (en gloss)	It's not a star, it's a crystal.

Table 7: Example of correct translation without any of the two conjunction variants (mostly occurring in French-to-Spanish and French-to-German on-line systems).

systems, disambiguating the conjunction "and" is probably more sensitive to the amount of the training data than disambiguating the conjunction "but". More experiments on systems trained on different training data sizes should be carried out in the future.

Both target variants: Manual inspection revealed that all sentences with both "and" variants are correct: the affected source sentences contain "too", "as well" or similar, which can be correctly translated as and_1. Three examples can be seen in Table 9.

None of the two target variants: A small number of sentences containing "neither" were correctly paraphrased in the target language so that it does not need any of the two "and" variants. These cases are found only in translations generated by on-line systems. An example can be seen in Table 10.

6 Summary and Outlook

We present a targeted evaluation of 20 translation outputs regarding their performance in lexical choice for the ambiguous source conjunction "but", and 6 (10) systems regarding the source conjunction "and". For the source conjunction "but", we observe that all systems almost perfectly recognise the target conjunction but_1, whereas accuracies for the other target variant but_2 are lower, and depend on the model (NMT performs better than PBMT), as well as on the amount of training data. For on-line systems trained on large amounts

		and_2			and_1		
language		correct		opposite	correct		opposite
pair	system	aut.	full	(and_1)	aut.	full	(and_2)
en-sr	google	38.8	39.8	58.8	98.1	98.1	1.9
	bing	33.5	33.5	62.2	92.3	92.3	5.8
	setimes-pbmt	22.3	22.3	75.4	98.1	98.1	1.9
	setimes-nmt	13.6	13.6	83.5	96.2	98.1	0
en-hr	google	42.7	43.7	54.9	96.2	96.2	3.8
	bing	54.8	56.3	41.9	96.2	96.2	1.9
pt-sr	google	35.7	36.2	59.3	92.2	92.2	5.9
	bing	32.7	32.7	61.8	94.1	94.1	5.9
pt-hr	google	37.7	37.7	57.3	90.2	90.2	3.9
	bing	49.2	49.2	46.2	92.2	92.2	7.8

Table 8: Percentage of correct target language conjunctions retrieved automatically and by full evaluation, and percentage of opposite target conjunctions for the source language conjunction "and".

source:	I can swim well **and so** do you.
output (hr) :	ja mogu plivati dobro, **a i** ti.
output (en gloss):	I can swim well, and_2 and_1 you.
source:	Holly travels a lot, **and** her sister, **too**.
output (hr)	Holly puno putuje, **a i** njena sestra.
output (en gloss):	Holly travels a lot, and_2 and_1 her sister.
source:	John likes burger, **and** he likes fish, **too**.
output (hr)	John voli hamburger, **a** voli **i** ribu.
output (en gloss):	John likes burger, and_2 likes and_1 fish.

Table 9: Examples of correct translations into Croatian with both target variants of "and".

source	I don't want you here, **and neither** does my wife.
output (hr)	Ne želim te ovdje, **kao ni** moja žena.
output (en gloss)	I don't want you here, **as not** my wife.

Table 10: Example of correct translation into Croatian without any of the two conjunction variants.

of data, accuracies range from 73% to 94%, and for the systems trained on small amounts between 50% and 60%. The errors for all systems are mostly caused by replacing the conjunction but_2 with the alternative conjunction but_1.

As for the conjunction "and", its first variant and_1 is also not problematic, even for the systems trained on small amounts of data. The variant and_2 is, however, much more challenging than but_2, with the highest accuracy of 56.3%. In addition, disambiguation of this conjunction seems to be more sensitive to the training data scarcity for NMT than for PBMT. More systematic experiments including both models and different sizes of the training corpus should be carried out in the future to better understand this finding.

In addition to this, there are many other directions for future work. The current study is focused on only two ambiguous conjunctions and only two target language variants for each of them. More conjunctions and ambiguities should be investigated in the future, as well as more source and target languages. Quantitative analysis of correlation between the conjunction disambiguation and overall performance should be a part of future work, too. Also, improving a MT system by, for example, adding more parallel data containing "difficult" target conjunction variants should be investigated as well.

7 Acknowledgements

We would like to thank Alberto Poncelas and Pintu Lohar for providing PBMT and NMT translation outputs for English-to-German and English-to-Serbian. The ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Kenneth A. Adams and Alan S. Kaye. 2006. Revisiting the Ambiguity of "And" and "Or" in Legal Drafting. *St. John's Law Review* 90(4).

Aljoscha Burchardt, Vivien Macketanz, Jonathan Dehdari, Georg Heigold, Jan-Thorsten Peter, and Philip Williams. 2017. A Linguistic Evaluation of Rule-Based, Phrase-Based, and Neural MT Engines. *The Prague Bulletin of Mathematical Linguistics* 108(1):159–170.

Sheila Castilho, Joss Moorkens, Federico Gaspari, Rico Sennrich, Vilelmini Sosoni, Yota Georgakopoulou, Pintu Lohar, Andy Way, Antonio Miceli Barone, and Maria Gialama. 2017. A comparative quality evaluation of pbsmt and nmt using professional translators. In *Proceedings of MT Summit XVI*. pages 116–131.

Liane Guillou, Christian Hardmeier, Ekaterina Lapshinova-Koltunski, and Sharid Loiciga. 2018. A Pronoun Test Suite Evaluation of the English–German MT Systems at WMT 2018. In *Proceedings of the 3rd Conference on Machine Translation (WMT 2018)*. Association for Computational Linguistics, Belgium, Brussels, pages 576–583.

Xiuming Huang. 1983. Dealing with Conjunctions in a Machine Translation Environment. In *Proceedings of the 1st Conference on European Chapter of the Association for Computational Linguistics (EACL 1983)*. Pisa, Italy, pages 81–85.

Filip Klubička, Antonio Toral, and Víctor M. Sánchez-Cartagena. 2018. Quantitative Fine-grained Human Evaluation of Machine Translation Systems: A Case Study on English to Croatian. *Machine Translation* 32(3):195–215.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Vancouver, pages 28–39.

Mathias Müller, Annette Rios Gonzales, Elena Voita, and Rico Sennrich. 2018. A Large-Scale Test Set for the Evaluation of Context-Aware Pronoun Translation in Neural Machine Translation. In *Proceedings of the 3rd Conference on Machine Translation (WMT 2018)*. Association for Computational Linguistics, Belgium, Brussels, pages 61–72.

Vladimir Pericliev. 1984. Handling syntactical ambiguity in machine translation. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics (ACL 1984)*. Stanford, California, USA, pages 521–524.

Maja Popović and Mihael Arčan. 2015. Identifying main obstacles for statistical machine translation of morphologically rich south Slavic languages. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation (EAMT 2015)*. Antalya, Turkey, pages 97–104.

Annette Rios Gonzales, Laura Mascarell, and Rico Sennrich. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *Proceedings of the 2nd Conference on Machine Translation (WMT 2017)*. Copenhagen, Denmark, pages 11–19.

Annette Rios Gonzales, Mathias Mller, and Rico Sennrich. 2018. The word sense disambiguation test suite at wmt18. In *Proceedings of the 3rd Conference on Machine Translation (WMT 2018)*. Association for Computational Linguistics, Belgium, Brussels, pages 594–602.

Richa Sharma, Jaspreet Bhatia, and K. K. Biswas. 2014. Machine Learning for Constituency Test of Coordinating Conjunctions in Requirements Specifications. In *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE 2014)*. Hyderabad, India, pages 25–31.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*. Istanbul, Turkey, pages 2214–2218.

Francis M Tyers and Murat Serdar Alperen. 2010. South-east European Times: A parallel corpus of Balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages*. Valetta, Malta, pages 49–53.

Li Fang Xu, Yun Zhu, Li Jiao Yang, and Yao Hong Jin. 2014. Research on sentence segmentation with conjunctions in patent machine translation. In *Applied Science, Materials Science and Information Technologies in Industry*. Trans Tech Publications, volume 513 of *Applied Mechanics and Materials*, pages 4605–4609.

ULSAna: Universal Language Semantic Analyzer

Ondřej Pražák
NTIS
Faculty of Applied Sciences
University of West Bohemia
Technická 8, 306 14 Plzeň
ondfa@ntis.zcu.cz

Miloslav Konopík
Dpt. of Computer Science and Engineering
Faculty of Applied Sciences
University of West Bohemia
Technická 8, 306 14 Plzeň
konopik@kiv.zcu.cz

Abstract

We present a live cross-lingual system capable of producing shallow semantic annotations of natural language sentences for 51 languages at this time. The domain of the input sentences is in principle unconstrained. The system uses single training data (in English) for all the languages. The resulting semantic annotations are therefore consistent across different languages. We use CoNLL Semantic Role Labeling training data and Universal dependencies as the basis for the system. The system is publicly available and supports processing data in batches; therefore, it can be easily used by the community for research tasks.

1 Introduction

In this work, we present a major outcome in our journey to build a system capable of producing semantic annotations for domain and language unconstrained natural language sentences. Currently, we rely on the Semantic Role Labeling (SRL) annotation scheme (Gildea and Jurafsky, 2002). The SRL goal is to determine semantic relationships (*semantic roles*) of given *predicates* (see examples in Figure 1). Verbs, such as "believe" or "cook", are natural predicates but certain nouns can be accepted as predicates as well (see the third line in the example). The semantic roles are specific for each predicate; however, the meaning of the roles is mostly shared across predicates. The core roles are denoted by *A0* (usually Agent), *A1* (usually Patient) and *A2*. Additional roles are modifier arguments (*AM-**), restriction arguments (*R-**) and others. We selected SRL because we believe that the annotations are simple enough to be generalized for different languages and target domains but at the same time expressive enough to bring a useful insight into the sentence semantics.

(1) [He]$_{\text{A0}}$ believes [in what he plays] $_{\text{A1}}$.

(2) Can [you] $_{\text{A0}}$ cook [the dinner] $_{\text{A1}}$?

(3) [The nation‘s] $_{\text{AM-LOC}}$ largest [pension]$_{\text{A1}}$ fund,

Figure 1: Three examples of shallow semantic annotations: 1) and 2) are examples of verb predicates and 3) of a noun predicate.

In order to be able to produce semantic annotations for more languages, we employ the *cross-lingual* SRL. Cross-lingual SRL takes the training data from a source language (usually English) and builds a language independent model that can be applied to target languages. The advantage of the cross-lingual SRL is that it ensures coherent annotation for all supported languages because it trains on single training data for all the languages. This does not apply to the monolingual SRL where the tagsets and annotation guidelines change with every training dataset.

In our approach, we heavily depend on Universal Dependencies (UD) (Nivre et al., 2016). They are the primary means to transfer the learned rules from one language to another. With universal dependencies, we can create language independent parse trees. It means that sentences with the same syntactic structure share (in theory) the same parse trees for all (supported) languages. We train our machine learning model on the UD trees to capture the syntactic patterns required for semantic role labeling. We do not use any lexical information or any other language dependent features. Our only information for SRL comes from UD trees. Thus, the resulting model can be applied to any of the supported languages.

The system we present in this paper is a web-based application written in Java – see the screen-

Proceedings of Recent Advances in Natural Language Processing, pages 967–972,
Varna, Bulgaria, Sep 2–4 2019.

shots in Figures 3 and 2. The system allows a user to input a natural language sentence in any of the 51 languages. The system outputs SRL annotations (predicates and corresponding semantic roles) of the input sentences. The semantic roles are associated with syntactic tree nodes. The video demonstration of the system is available here: `https://www.youtube.com/watch?v=8QPKCegHT_c`. The system itself can be accessed at the following address: `http://liks.fav.zcu.cz/ulsana`. We intend to support the system for public use for several years.

2 Related Work

Approaches to cross-lingual SRL can be divided into three main categories: **1)** *Annotation projection* methods attempt to transfer annotations from one language to another and then they train an SRL system on the transferred annotations. **2)** *Model transfer* approaches are designed to use language-independent features to train a universal model which can be applied to languages that support the designed features. **3)** Methods based on *unsupervised training* require no annotated data; however, the models have difficulties in assigning meaningful labels to predicate arguments.

Annotation projection Padó and Lapata (2009) transfer annotations to a target language via word alignments obtained from parallel corpora. Annesi and Basili (2010) use a similar approach and extend it with an HMM model to increase the transfer accuracy.

Model transfer Kozhevnikov and Titov (2013) use cross-lingual word mappings and cross-lingual semantic clusters obtained from parallel corpora, and cross-lingual features extracted from unlabelled syntactic dependencies to create a cross-lingual SRL system. In (Kozhevnikov and Titov, 2014), they try to find a mapping between language-specific models using parallel data automatically.

Unsupervised SRL Grenager and Manning (2006) deploy unsupervised learning using the EM algorithm based upon a structured probabilistic model of the domain. Lang and Lapata (2011) discover arguments of verb predicates with high accuracy using a small set of rules. A split-merge clustering is consequently applied to assign (nameless) roles to the discovered arguments. Titov and Klementiev (2012) propose a superior argument clustering by using the Chinese restaurant process.

Our approach belongs among the model transfer approaches. Most of the other state-of-the-art approaches to SRL rely on lexical features (e.g. word lemmas). In the cross-language scenario, such features require bilingual models (e.g. word mapping via machine translation or bilingual clusters). In our demonstration application, we show a multi-language model that is capable of producing annotations for many languages. Therefore, we omit all bilingual features including the lexical features from our model.

3 System Description

In this section, we describe the core of the cross-lingual system we use in our demo. The system is described in our original paper (Pražák and Konopik, 2017) in full details. Here, we explain only the basic principles. The system described in (Pražák and Konopik, 2017) is available for five languages only. In this demo, we extended the system for 51 languages.

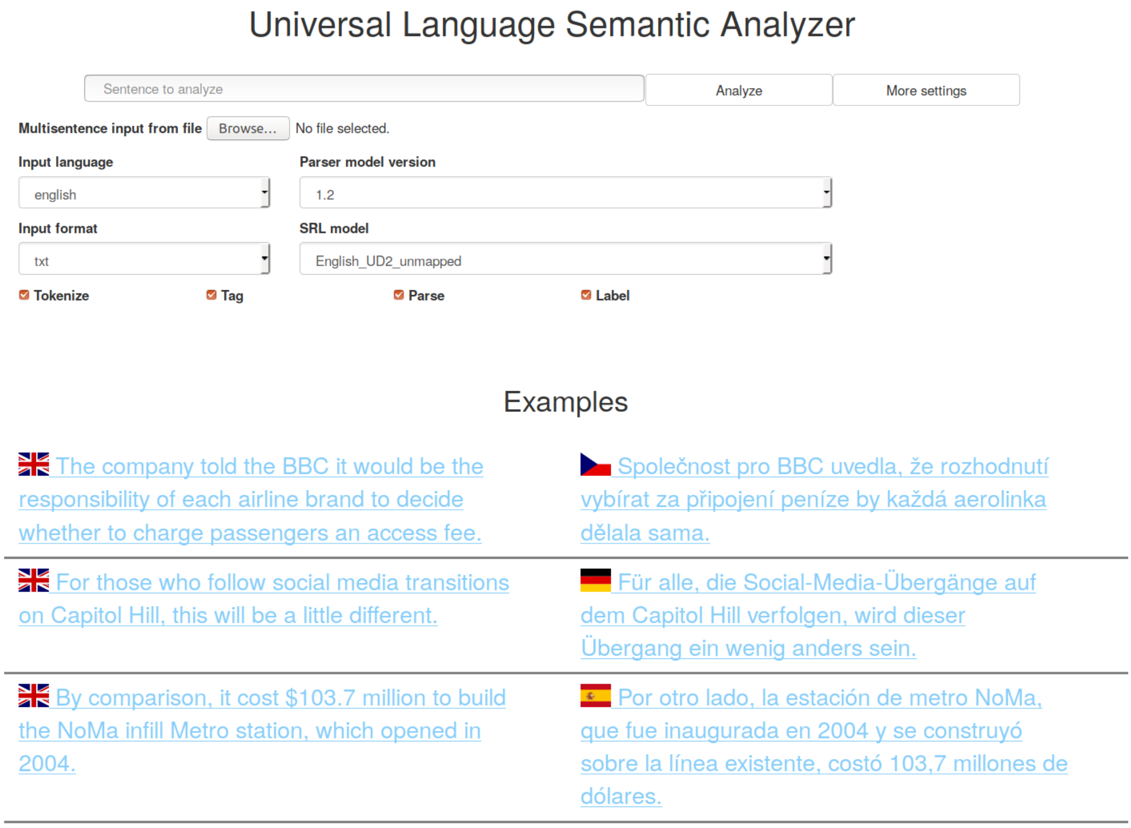
3.1 Training Dataset and Annotation Conversion

We train our system on UD parse trees. However, there are no such training data that would contain SRL annotations on UD trees. Therefore, we proposed an algorithm to convert existing SRL annotations built on SD[1] trees.

The conversion process is by no means straightforward. The main source of complications stems from different approaches to choose head words for syntactic phrases in UD trees. To solve this issue, we proposed optimization algorithms that attempt to select the most appropriate heads for UD trees which would cover the same phrases as the heads in original SD trees. In many cases, there is no such word in UD trees which could be used as the new head. In such cases, we choose the head that minimizes the annotation error. The details of the proposed conversion algorithms are presented in the original paper – Section 4.

In our application, we use the CoNLL 2009 English dataset (Hajič et al., 2009). The corpus

[1] SD stands for Standard or language-Specific Dependencies, e.g. Stanford dependencies – urlhttps://nlp.stanford.edu/software/stanford-dependencies.shtml.



Universal Language Semantic Analyzer

Sentence to analyze | Analyze | More settings

Multisentence input from file Browse... No file selected.

Input language: english

Parser model version: 1.2

Input format: txt

SRL model: English_UD2_unmapped

☑ Tokenize ☑ Tag ☑ Parse ☑ Label

Examples

The company told the BBC it would be the responsibility of each airline brand to decide whether to charge passengers an access fee.	Společnost pro BBC uvedla, že rozhodnutí vybírat za připojení peníze by každá aerolinka dělala sama.
For those who follow social media transitions on Capitol Hill, this will be a little different.	Für alle, die Social-Media-Übergänge auf dem Capitol Hill verfolgen, wird dieser Übergang ein wenig anders sein.
By comparison, it cost $103.7 million to build the NoMa infill Metro station, which opened in 2004.	Por otro lado, la estación de metro NoMa, que fue inaugurada en 2004 y se construyó sobre la línea existente, costó 103,7 millones de dólares.

Figure 2: Application screenshot

includes syntactic dependencies (from the Penn Treebank [TB]) and semantic dependencies (from PropBank [PB] and NomBank [NB]).

3.2 Universal Dependencies Parser

Our system requires syntactic trees in the UD annotation scheme. We rely on the freely available tool UDPipe (Straka et al., 2016). It contains pretrained models for all the languages we support in our application. We use models provided with parsers based on UD. The algorithms were developed on UD v1.2 models based on UD 2.0 were also added, and they achieve better results (about +1% of labeling accuracy).

3.3 Classifier & Features

We train a supervised system based upon the Maximum Entropy classifier using the Brainy tool (Konkol, 2014). We use separate models for verb and non-verb predicates.

All features employed in our system are syntactic:

- *Predicate-argument distance* – the distance between the locations of a predicate and an argument in a sentence.
- *POS* – part-of-speech of the predicate, the argument and their parent nodes.
- *Dependency relation* – dependency tree relation of the predicate, the argument and their parent nodes.
- *Directed path* – dependency tree path from the predicate to the argument including the indication of the dependency directions.
- *Undirected path* – the list of relations from the predicate to the argument.
- *Verb voice* – indication of active/passive voice.
- *Other syntactic features* – `feats` column in CoNLL 2009 format with additional information about the words.
- *Bigram features* – predicate-argument bigrams of the part-of-speech and the dependency relations.

The dependency path features are encoded as a probability of a word being a predicate argument (or having a specific role) given the path. These features are more general, and the resulting vectors have a smaller dimension. Also, the cost function is smoother, and thus the model is easier to train.

3.4 Web Application Description

We created a Java web UI for the SRL annotation and its visualization. We use *TikZ* for visualization of the trees. The *TikZ* output is converted to *SVG* which is then shown in the browser. The application takes its input either in plain text or in various CoNLL formats. Input can be a single sentence or a file with sentences separated by new lines. On the input, a user has to select an input language (one of 51 supported languages) because syntactic parsers are language-dependent and the application cannot determine the language automatically at this time. The application can parse the sentences syntactically and semantically. After these steps (if requested) the annotations are visualized in the SVG format and showed in the browser. The user can download analyzed sentences in *svg*, *pdf* or raw *CoNLLu* format.

3.5 Application Use Cases

Research Experiments The primary purpose of our application is to help the researchers to get familiar with the capabilities of cross-lingual semantic processing. We also want to demonstrate the power and limitations of Universal Dependencies. Users can work with examples entered into the input field, but they can also use the batch processing feature. In this way, users can obtain SRL annotations of larger corpora that can be used in the consequent research.

Language Learning The application can also help users who are learning a new language. Our application shows the structure of a sentence and the basic roles of the main phrases in the sentence. In this way, users can more easily understand the semantic structure of the sentence.

Translation Cross-lingual SRL can be used either in the machine or human translation. When translating a sentence, we aim to preserve the semantic structure of the sentence. We can achieve that by studying both structures of the source and target input sentences.

3.6 Known issues

- Parser errors – Since our system relies solely on syntactic features, it is very sensitive to parser errors. When the sentences match the domain of training data of UD annotations (mostly news domain) the parse trees are generally quite correct. We produce mostly correct SRL annotations with correct parser trees. However, our system is usually unable to classify correctly when the parse trees contain significant errors.

- Visualizing complex relations – Our system sometimes struggles with visualizing complex relations. It those circumstances the resulting visualization can be confusing.

4 Future Work

In future work, we plan to adopt the end-2-end approach to Semantic Role Labeling. We intend to attach the SRL annotation after UD parsing and use a global cost function to optimize the UD parsing and the SRL annotation simultaneously. In order to apply the end-2-end approach, we might have to switch to the SyntaxNet[2] UD parser. We expect to be able to produce more robust SRL annotations with one global optimization function.

Next, we plan to focus on lexical features. We want to stay with the idea of one model for many languages. Therefore, we need to use cross-lingual embeddings as lexical features.

5 Conclusion

We have created a semantic role labeling system with a massively multilingual model. A single model can be used for SRL in 51 different languages. The system supports large inputs, and therefore it can be used to annotate entire datasets for various NLP tasks.

Acknowledgments

This publication was supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications. Computational resources were provided by the CESNET LM2015042 and

[2] https://opensource.google.com/projects/syntaxnet

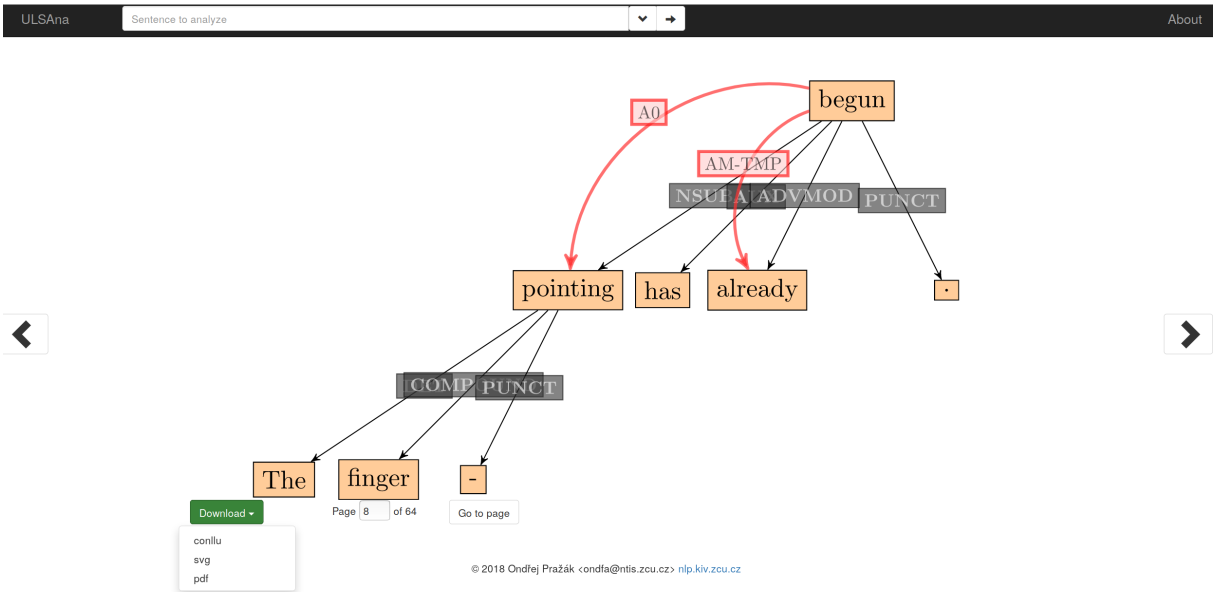


Figure 3: Application screenshot – sentence visualization example

the CERIT Scientific Cloud LM2015085, provided under the programme ”Projects of Large Research, Development, and Innovations Infrastructures”.

References

Paolo Annesi and Roberto Basili. 2010. Cross-lingual alignment of FrameNet annotations through hidden markov models. In *Proceedings of the 11th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing’10, pages 12–25, Berlin, Heidelberg. Springer-Verlag.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Trond Grenager and Christopher D. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP ’06, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Michal Konkol. 2014. Brainy: A machine learning library. In *Artificial Intelligence and Soft Computing*, volume 8468 of *Lecture Notes in Computer Science*, pages 490–499. Springer International Publishing.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1190–1200.

Mikhail Kozhevnikov and Ivan Titov. 2014. Cross-lingual model transfer using feature representation projection. In *ACL (2)*, pages 579–585.

Joel Lang and Mirella Lapata. 2011. Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1117–1126, Portland, Oregon, USA. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Ondřej Pražák and Miloslav Konopik. 2017. Cross-lingual srl based upon universal dependencies. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 592–600, Varna, Bulgaria. INCOMA Ltd.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Paris, France. European Language Resources Association (ELRA).

Ivan Titov and Alexandre Klementiev. 2012. A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 12–22, Stroudsburg, PA, USA. Association for Computational Linguistics.

Machine Learning Approach to Fact-Checking in West Slavic Languages

Pavel Přibáň[1,2], **Tomáš Hercig**[2], and **Josef Steinberger**[1]

[1]Department of Computer Science and Engineering,
Faculty of Applied Sciences, University of West Bohemia, Czech Republic
[2]NTIS – New Technologies for the Information Society,
Faculty of Applied Sciences, University of West Bohemia, Czech Republic
{pribanp,tigi,jstein}@kiv.zcu.cz
http://nlp.kiv.zcu.cz

Abstract

Fake news detection and closely-related fact-checking have recently attracted a lot of attention. Automatization of these tasks has been already studied for English. For other languages, only a few studies can be found (e.g. (Baly et al., 2018)), and to the best of our knowledge, no research has been conducted for West Slavic languages. In this paper, we present datasets for Czech, Polish, and Slovak. We also ran initial experiments which set a baseline for further research into this area.

1 Introduction & Motivation

Fake news is designed to incite agitation against an individual or a group of people. Its aim is to influence and manipulate public opinion on targeted topics. Fake news detection, including fact-checking, which can be used as the first step of a detection system, are currently receiving a lot of attention in the research community and journalism.

This attention is apparent from the rise of fact-checking websites that verify mainly political claims (see the list of signatories of the code of principles of the International Fact-Checking Network[1]). Research related to these tasks is on the rise in a variety of fields, including natural language processing, machine learning, knowledge representation, databases, and journalism (Thorne and Vlachos, 2018).

The automation of these tasks or their parts would greatly benefit journalism and perhaps help the public to verify the credibility of various media. It is evident that fact-checking needs external knowledge or detailed context. However, in order to achieve the goal of a robust automatic fact-checking system, we must first find a way how to evaluate such a system. For English, there are publicly available datasets that researchers can use to evaluate their systems. However, no systematic research has been conducted in West Slavic languages yet; thus we establish a common ground for further research by providing large datasets for fact-checking in Czech, Polish, and Slovak languages including initial experiments which reveal the complexity of the task. We set a baseline which uses standard machine learning approach, and set an upper bound which uses manually created external knowledge.

2 Related Work

This section presents a brief overview of related work, for a more detailed survey, please refer, for example to Thorne and Vlachos (2018).

For the development of the first fact-checking systems, Vlachos and Riedel (2014) manually labeled a dataset and defined fact-checking as the assignment of a truth Boolean value to a claim made in a particular context. They also discussed baseline approaches to fact-checking.

Wang (2017) presented a dataset of 12.8K manually labeled statements from the Politifact[2] website. He experimented with logistic regression, support vector machines, Long Short-Term Memory neural networks (LSTM), and convolutional neural networks (CNN). He then introduced a modified neural network architecture integrating text with other meta-data. He performed similar experiments to our work on English dataset with six labels achieving 27.7% accuracy as the best result.

Tacchini et al. (2017) showed that fake news

[1] https://ifcncodeofprinciples.poynter.org/signatories

[2] https://www.politifact.com/

Language	MISLEADING	UNVERIFIABLE	FALSE	TRUE	ALL
Czech	848 (9.3%)	1343 (14.8%)	1222 (13.5%)	5669 (62.4%)	9082 (100%)
Polish	313 (11.0%)	113 (4.0%)	648 (22.9%)	1761 (62.1%)	2835 (100%)
Slovak	1146 (9.1%)	1751 (13.9%)	1670 (13.3%)	7987 (63.6%)	12554 (100%)

Table 1: Dataset label statistics.

Lang.	M.	U.	F.	T.	ALL
CS	39 / 44	38 / 44	33 / 39	33 / 38	34 / 39
PL	28 / 32	19 / 25	19 / 24	22 / 26	22 / 26
SK	36 / 40	36 / 40	29 / 33	32 / 36	32 / 37

Table 2: Dataset size in words (median/average).

could be detected based on user likes. Using an adaptation of a Boolean label crowdsourcing algorithm, they were able to detect hoaxes with 99% accuracy. Their dataset consists of 15.5K posts (58% fake news, 42% real news) with over 2,300K likes from 900K users.

Jin et al. (2017) focused on detecting fake news on Twitter related to the U.S. presidential elections. They labeled the data according to the Snopes[3] website. They analysed tweets of followers of the presidential candidates.

Yang et al. (2018) used a dataset of 20K news (12K fake news, 8K real news) for fake news detection. They used a modified convolutional neural network trained using the title, images and text of the news articles, making use of both explicit and latent features to detect fake news. They achieved F_1-measure of 92% overcoming a baseline LSTM text-based model by 3%. They presented a thorough analysis of the dataset, including text style and image resolution.

In this paper we present the following novel contributions:

1. The availability of multi-lingual data for non-English languages is lacking. Our paper addresses this need.

2. The dataset also contains reasoning for labeling each claim - this can be used in future research, e.g. argumentation mining.

3. The claims are also labeled by Political party affiliations - this may facilitate fine-grained analysis.

3 Dataset

We provide three datasets for fact-checking - one for each language downloaded from the following fact-checking websites.

- Czech (https://demagog.cz/)
- Polish (http://demagog.org.pl/)
- Slovak (http://www.demagog.sk/)

Each dataset contains claims of politicians[4] annotated with one of four classes: FALSE, TRUE, UNVERIFIABLE, and MISLEADING. The labels have the following meaning:

- FALSE These statements are not in line with publicly available numbers or information. It may also be a situation where the calculation method of the indicator differs, but none of these sources confirms the number or claim in question.

- TRUE Statement using the right information in the right context.

- UNVERIFIABLE If it is not possible to find the source of the claim, or it is not possible to confirm or refute it based on the available information.

- MISLEADING These are statements that use correct facts, but in a wrong or incomplete context, or are being torn out or otherwise distorted from the original context. These are inappropriate or disproportionate comparisons.

The labels are manually annotated by the authors of the corresponding language websites. The dataset also contains information about the speaker and his or her political affiliation. The reasoning[5] for the given label is also included in the dataset. The data were downloaded from the respective websites in April 2018. The following example has been translated into English.

[3] https://www.snopes.com/

[4] Other publicly active people such as journalist are included in the dataset as well.

[5] Including external knowledge.

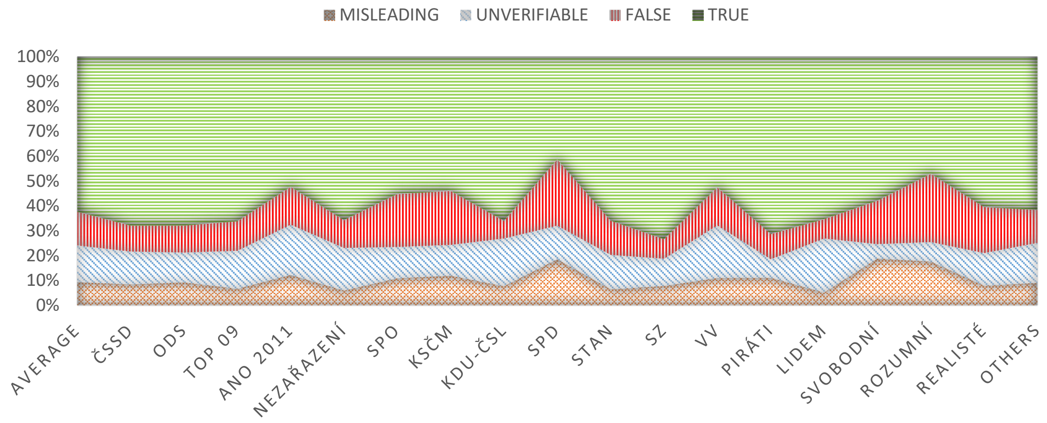


Figure 1: Czech Political Parties Statistics

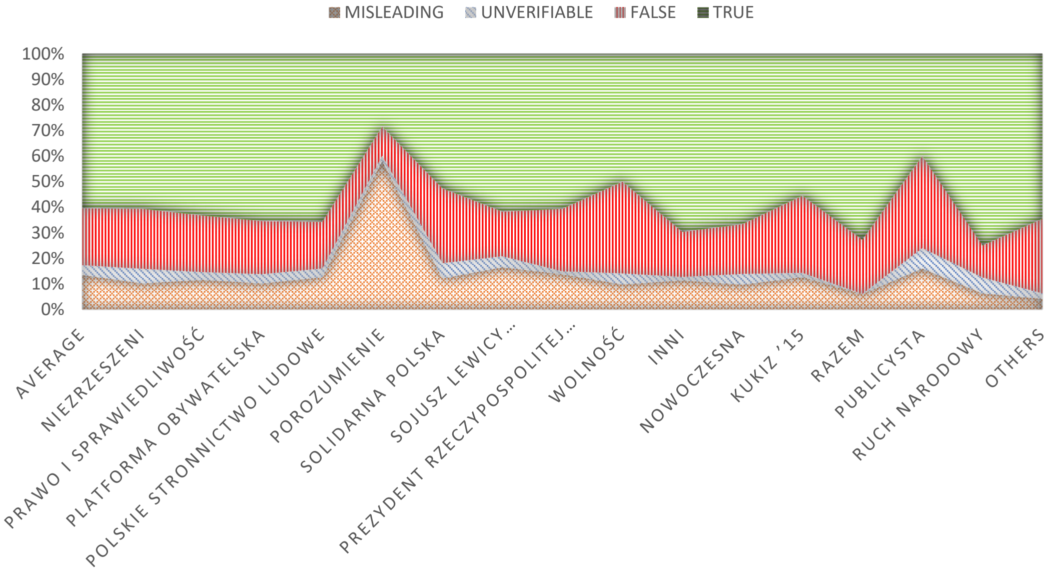


Figure 2: Polish Political Parties Statistics

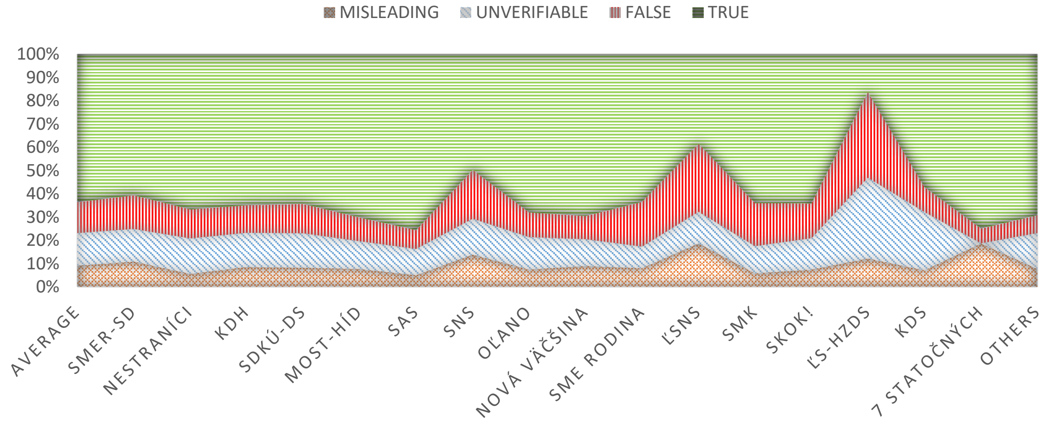


Figure 3: Slovak Political Parties Statistics

> Miloš Zeman (SPO) → FALSE
> CLAIM: "The Swedes have seven million inhabitants."
> REASONING: Sweden has according to the latest official data from November 2017 10,113,000 inhabitants.

The data distribution, according to the labels, is shown in Table 1. Table 2 shows the median and the average number of words in a claim.

We compare the label distribution among selected political parties with the most claims. Figures 1, 2, and 3 show the average label distribution and the distribution for the selected political parties sorted by a number of claims for Czech, Polish, and Slovak languages. Note that the labels *Nezařazení* for Czech, *Niezrzeszeni* for Polish, and *Nestraníci* for Slovak represent claims of people without any political party affiliation. The `Other` label is the average of the rest[6] of the political parties present in the dataset [7].

It is clear that the claims of some political parties often tend to be truth compared to other parties. This phenomenon can be observed for all three languages. The opposite applies to the Czech parties *SPD*, *Rozumní*, Polish party *Porozumienie* and Slovak party *LS-HZDS*. However, the inconsistency of `UNVERIFIABLE` label across languages was more surprising. We believe that it is caused by differences in labeling i.e. that in the Polish dataset the `UNVERIFIABLE` label is used only under stringent rules in comparison with the other two languages.

4 Experiments

We performed identical classification experiments for each language to allow a comparison for future research. The main goal of these experiments is to illustrate the complexity of the task and to set a baseline for these datasets.

We use 10-fold cross-validation for the evaluation of both balanced and imbalanced datasets. We also perform binary experiments only with `FALSE` and `TRUE` classes. The input for the classifier is either the text of a claim or a text of a claim supplemented by the reasoning text. Experiments using the reasoning text set up an upper bound of performance that can be achieved with an automatic approach. Our evaluation metrics are macro-average F_1 score and accuracy.

The reasoning text often contains words or phrases which are strictly related to the assigned label, for example, *Výrok je pravdivý* (The statement is true) for the `TRUE` label or *Výrok nelze ověřit* (The statement is unverifiable) for the `UNVERIFIABLE` label. We call these words *give-away words* as they alone will be a sufficient source of information for the classifier. In other words, the reasoning text in a large number of cases de facto contains the label.

We removed these words from the reasoning text and repeated the experiments with the modified reasoning text. The list of removed give-away words was manually selected from the words with highest label occurrence ratio[8]. All words were selected only if they occurred at least 20 times in the corresponding label class. Finally, we manually chose words and removed them from the reasoning text, see Table 3 that contains examples of the removed give-away words. For Czech, we removed $9,601$ words out of $1,552,878$, for Slovak, we removed $9,147$ words out of $2,146,465$ and for Polish, we removed 573 words out of $367,435$. The complete list of the removed words is available at `http://nlp.kiv.zcu.cz/projects/fact-checking`.

Czech	Polish	Slovak
nepravdivý	fałszywą	nepravdivý
pravdivíy	prawdziwą	pravdivý
neověřitelný	nieweryfikowalną	neoverite'ný
neodpovídá	manipulację	nevieme

Table 3: Examples of give-away words.

4.1 Models Settings

The preprocessing includes tokenization using NLTK *TreebankWordTokenizer* (Bird et al., 2009), text lowercasing, removing HTML tags and entities. No other preprocessing steps are employed. We use Logistic Regression classifier from the LIBLINEAR library (Fan et al., 2008) with penalty parameter $C = 1$ and L2 regularization (see Fan et al. (2008) for detailed description), along with

[6] The rest of the parties that had fewer claims than the selected parties. In the Czech dataset, this includes the `null` value used for people who changed parties over time.

[7] The dataset is available for research purposes at `http://nlp.kiv.zcu.cz/projects/fact-checking`

[8] The number of occurrences of words for a given label divided by the total frequency. We selected words with a ratio ≥ 0.8 for the `TRUE` label, and words with a ratio ≥ 0.6 for the other three labels.

		Czech		Polish		Slovak	
Dataset	Labels	Macro F_1	Accuracy	Macro F_1	Accuracy	Macro F_1	Accuracy
Imbalanced	4	0.21 / 0.19	0.25 / 0.62	0.21 / 0.19	0.25 / 0.62	0.21 / 0.19	0.25 / 0.64
Balanced	4	0.25 / 0.25	0.25 / 0.25	0.25 / 0.25	0.25 / 0.25	0.25 / 0.25	0.25 / 0.25
Imbalanced	2	0.35 / 0.45	0.35 / 0.82	0.34 / 0.42	0.34 / 0.73	0.33 / 0.45	0.33 / 0.83
Balanced	2	0.50 / 0.50	0.50 / 0.50	0.50 / 0.50	0.50 / 0.50	0.50 / 0.50	0.50 / 0.50

Results of *random / majority* class classifiers.

Table 4: Results of a random and majority class (separated by slash *random / majority*) classification. For example, the accuracy for Czech imbalanced dataset for all four labels is 0.25 for the random classifier, 0.62 for the majority class classifier.

		Czech		Polish		Slovak	
Dataset	Labels	Macro F_1	Accuracy	Macro F_1	Accuracy	Macro F_1	Accuracy
Imbalanced*	4	0.26	0.61	0.25	0.60	0.27	0.62
Balanced*	4	0.31	0.31	0.26	0.26	0.35	0.35
Imbalanced*	2	0.48	0.81	0.49	0.72	0.51	0.82
Balanced*	2	0.57	0.57	0.54	0.55	0.58	0.58
Imbalanced†	4	0.87	0.91	0.45	0.69	0.79	0.86
Balanced†	4	0.85	0.85	0.46	0.47	0.78	0.78
Imbalanced†	2	0.88	0.94	0.63	0.77	0.86	0.93
Balanced†	2	0.86	0.87	0.64	0.64	0.85	0.85
Imbalanced‡	4	0.51	0.72	0.36	0.64	0.53	0.72
Balanced‡	4	0.54	0.54	0.41	0.43	0.56	0.56
Imbalanced‡	2	0.65	0.85	0.59	0.74	0.67	0.85
Balanced‡	2	0.71	0.71	0.61	0.61	0.68	0.68

* dataset only with claim

† dataset with both claim and reasoning.

‡ dataset with both claim and reasoning without give-away words.

Table 5: Results of logistic regression classification.

unigram and bigram features. Experiments with the reasoning are performed on a combination of the claim text and the reasoning text. First, the reasoning text and the claim text are concatenated, and then we extract the unigram and bigram features. These features are used as an input to the classifier.

4.2 Results

We report results for the experiments for all three languages, including results of a random and majority class classification in Table 4.

In Table 5 we show the results for the Logistic Regression classifier on the balanced and imbalanced datasets for the following text combinations:

- claim
- claim & reasoning
- claim & reasoning without give-away words

On the balanced dataset we can see that using only unigrams and bigrams as features is not enough for the classifier as the results are only slightly better than the majority baseline; thus more sophisticated methods are needed to extract the information contained in the reasoning part of the dataset.

We can see that the results achieved on both claim and reasoning are very high (F_1 0.87, accuracy 0.91 for Czech) confirming our hypothesis that in ideal conditions this task could be solved by a machine learning algorithm. In the case of using only the claim on the balanced binary dataset, accuracy drops to 0.57, 0.55, and 0.58 for Czech, Polish, and Slovak, respectively. Polish appears as the most challenging language as the results are lower compared to the other two languages. One reason could be a smaller size of the Polish dataset (see Table 2).

In the case of experiments with the *give-away words*, results are still much higher than in experiments where only the claim was used (F_1 0.53, accuracy 0.72 for Slovak). We observed the highest performance drop for experiments with all four labels, especially for Czech, in comparison to the experiments with the original reasoning text. The performance of the Polish model was least affected. This was caused by the low number of removed words (573 words out of 367,435 in total) for Polish. Thus the assumption that the reasoning contains only give-away words is false; leading us to believe that some information about the validity of the claim is contained in the reasoning.

5 Conclusion

This paper represents the initial research of fact-checking in Czech, Polish, and Slovak languages.

- We presented datasets for fact-checking in three West Slavic languages and provided them to the research community.
- We ran initial experiments which revealed baseline results for further research.

It is clear that this task is very challenging. However, we showed that when a machine learning approach uses label reasoning in addition to the claim, it can perform very well. Although such human-written reasoning rather sets a performance upper bound, the way to go forward might include generating such reasoning automatically using external data.

Disclaimer

Any views, findings, and conclusions expressed in this article are based on a thorough analysis of very limited and dated data. They do not necessarily reflect the views, opinions, or official positions of the authors or the University of West Bohemia.

Acknowledgments

This work has been supported by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications, by the project LO1506 of the Czech Ministry of Education, Youth and Sports under the program NPU I and from ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)".

References

Ramy Baly, Mitra Mohtarami, James Glass, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2018. Integrating Stance Detection and Fact Checking in a Unified Corpus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, pages 21–27. https://doi.org/10.18653/v1/N18-2004.

S Bird, E Loper, and E Klein. 2009. Natural language processing with Python:“O’Reilly Media Inc.“ .

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research* 9:1871–1874.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, Yu Wang, and Jiebo Luo. 2017. Detection and Analysis of 2016 US Presidential Election Related Rumors on Twitter. In Dongwon Lee, Yu-Ru Lin, Nathaniel Osgood, and Robert Thomson, editors, *Social, Cultural, and Behavioral Modeling*. Springer International Publishing, Cham, pages 14–24.

Eugenio Tacchini, Gabriele Ballarin, Marco L Della Vedova, Stefano Moret, and Luca de Alfaro. 2017. Some like it hoax: Automated fake news detection in social networks. *CoRR* abs/1704.07506. http://arxiv.org/abs/1704.07506.

James Thorne and Andreas Vlachos. 2018. Automated Fact Checking: Task Formulations, Methods and Future Directions. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 3346–3359. http://aclweb.org/anthology/C18-1283.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Association for Computational Linguistics, Baltimore, MD, USA, pages 18–22. http://www.aclweb.org/anthology/W14-2508.

William Yang Wang. 2017. “Liar, Liar Pants on Fire“”: A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 422–426. https://doi.org/10.18653/v1/P17-2067.

Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. 2018. TI-CNN: Convolutional Neural Networks for Fake News Detection. *CoRR* abs/1806.00749. http://arxiv.org/abs/1806.00749.

NE-Table: A Neural Key-Value Table for Named Entities

Janarthanan Rajendran[*]
University of Michigan
rjana@umich.edu

Jatin Ganhotra[*]
IBM Research
jatinganhotra@us.ibm.com

Xiaoxiao Guo
IBM Research
xiaoxiao.guo@ibm.com

Mo Yu
IBM Research
yum@us.ibm.com

Satinder Singh
University of Michigan
baveja@umich.edu

Lazaros Polymenakos[†]
Amazon Alexa-AI Research
polyml@amazon.com

Abstract

Many Natural Language Processing (NLP) tasks depend on using Named Entities (NEs) that are contained in texts and in external knowledge sources. While this is easy for humans, the present neural methods that rely on learned word embeddings may not perform well for these NLP tasks, especially in the presence of Out-Of-Vocabulary (OOV) or rare NEs. In this paper, we propose a solution for this problem, and present empirical evaluations on: a) a structured Question-Answering task, b) three related Goal-Oriented dialog tasks, and c) a Reading-Comprehension task[1], which show that the proposed method can be effective in dealing with both in-vocabulary and OOV NEs.

1 Introduction

We come across Named Entities (NEs) in many Natural Language Processing (NLP) tasks. In tasks such as Question-Answering (QA) and goal-oriented dialog, NEs play a crucial role in task completion. Examples include QA systems for retrieving information about courses offered at a university, and dialog systems that perform restaurant reservation, flight ticket booking, and so on. In many cases, these tasks also involve interaction with external knowledge sources such as DataBases (DB) which could have a large number of NEs. In these tasks NEs include people's names, restaurant names, locations etc.

There has been a lot of interest in building neural methods for NLP tasks. Past work has developed multiple methods for addressing the unique challenges to neural methods posed by NEs. One straightforward method is to add each and every NE (including those in the DB) to the vocabulary. This method has been evaluated for only synthetic or small tasks (Neelakantan et al., 2015). For real world tasks, especially those with large DBs, this causes an explosion in the vocabulary size and hence the number of parameters to learn. There is also the problem of not being able to learn *good* neural embeddings for individual NEs, as individual NEs (e.g., a particular phone number) generally occur only a few times in a dataset. Another previously proposed method is to encode all the NEs with random embeddings and keep them fixed throughout (Yin et al., 2015), but here we lose the meaning associated with the neural embeddings and risk interference and correlation with others in unexpected ways.

A third method is to first recognize the NE-type with either NE taggers (Finkel et al., 2005) or entity linkers (Cucerzan, 2007; Guo et al., 2013), and then replace them with NE-type tags. For example, all location names could be replaced with the tag *NE_location*. This prevents the explosion in vocabulary size; however, the system loses the ability to distinguish and reference different NEs of the same type. There is also the possibility of new NEs arising during the test time. In fact, many of the Out-Of-Vocabulary (OOV) words that arise during test time in many NLP tasks (e.g. Bordes and Weston (2016)) are NEs. Furthermore, in many scenarios it is easier and accurate to work with the actual exact values of NEs rather than neural embeddings, like providing a phone number to a user or searching for a faculty name over a DB. None of the above neural methods have the ability to work with exact NE values.

[*] Equal Contribution

[†] This work was done when the author was at IBM Research, NY.

[1] We create extended versions of dialog bAbI tasks 1,2 and 4 and OOV versions of the CBT test set - https://github.com/IBM/ne-table-datasets/

Proceedings of Recent Advances in Natural Language Processing, pages 980–993,
Varna, Bulgaria, Sep 2–4 2019.

In this paper, we propose a novel neural method that addresses all the aforementioned issues. There are three aspects to our method.

- On-the-fly-generation: Neural embeddings for the NEs are generated on the fly using their context information. This avoids the explosion in vocabulary size, while still providing meaningful and distinguishable neural embeddings for the different NEs.
- Key-Value-Table: The generated embeddings are stored in a table (*NE-Table*), with embeddings as the keys (key-embeddings) and exact NEs as the values (NE-values).
- On-the-fly-Retrieval: The NE-values can later be retrieved from the *NE-Table* by attending over the key-embeddings, providing the ability to interact with exact NE values.

We demonstrate our method on a reading-comprehension task, a simple structured Question-Answering (QA) task, and three goal-oriented dialog tasks. Our method achieves 10% increase in accuracy for Reading-Comprehension, 19% increase for structured-QA and around 90% increase for goal-oriented dialog tasks, with respect to their corresponding baselines.

2 NE-Table: A Neural Key-Value Table for Named Entities

Our proposed method (Figure 1) has three aspects.

On-the-fly-generation. Neural embeddings for the NEs are generated on the fly using their context information (shown as the *NE-Embedding Generation Module* in Fig 1), instead of adding them to the vocabulary. The context information depends on the task. For a dialog task, the context is the full dialog so far, including the present utterance which has the NE in it. For the QA task, context is the sentence in which the NE appears. For the Reading Comprehension task, the sentence where the NE occurs or potentially the full story can be used as the context. The context could also include the NE-type information when available. The *NE-Embedding Generation Module*, denoted (f_ϕ), takes the context embedding as input and outputs the NE-Embedding. For our purposes, f_ϕ is an multi-layer perceptron (MLP). The problem of explosion in vocabulary size is avoided, as NEs are not part of the vocabulary and the NE-Embeddings are generated on the fly. Our proposed method also generates unique embeddings for different NEs with the same NE-type. This is better than replacing a NE with its NE-type as that results in all NEs with the same NE-type having the same embedding and hence, losing the ability to distinguish different NEs with the same NE-type. The generated NE-Embeddings are meaningful as they are learned from the context, in comparison to fixed random embeddings and can also be used as the learned neural embedding for that NE word from thereon.

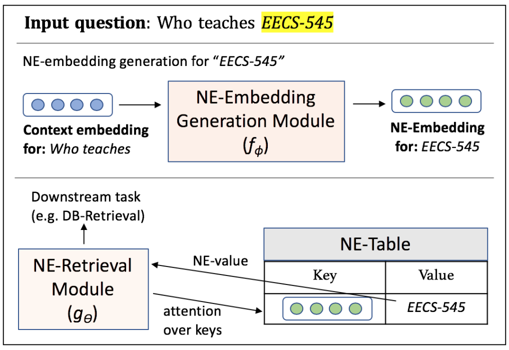

Input question: Who teaches EECS-545
NE-embedding generation for "EECS-545"
Context embedding for: Who teaches
NE-Embedding Generation Module (f_ϕ)
NE-Embedding for: EECS-545
Downstream task (e.g. DB-Retrieval)
NE-Retrieval Module (g_θ)
NE-value
attention over keys
NE-Table
Key
Value
EECS-545

Figure 1: For input question - *Who teaches EECS-545*, the *NE-Embedding Generation Module* (f_ϕ) takes the context embedding as input and generates a NE-Embedding for the NE *EECS-545*. The NE-Embedding is stored in *NE-Table* with its actual value *EECS-545*. The *NE-Retrieval Module* (g_θ) performs attention over the keys in *NE-Table* to retrieve the NE-value. We show a simple example here to illustrate f_ϕ and g_θ. Depending on the task, the context can vary and the *NE-Table* can have more entries.

Key-Value-Table. As discussed in the previous section, there are many scenarios where it is easier and more accurate to work with the exact values of NEs rather than their neural embeddings, like providing a phone number to a user or searching for a faculty name over a DB. For this purpose, the generated NE-Embedding, along with its exact NE value is stored in a table, *NE-Table*, as a key-value pair, with the embedding as key (key-embedding) and the exact NE as value (NE-value).

On-the-fly-Retrieval. The NE-value can later be retrieved from the *NE-Table* by performing attention over the key-embeddings in the *NE-Table*. This is performed by the *NE-Retrieval Module* (g_θ) shown in Figure 1. For our purposes, g_θ is an MLP. The input to *NE-Retrieval Module* also depends on the task. For dialog task, the dialog state vector is used, which has the information of the dialog so far. For QA task, the encoding of the input question is used. For Reading Comprehension task, the full story is used as input to the retrieval module. The retrieved NE-value can be used in the output utterance (e.g. providing a phone number) or to do an exact match over values in a DB (e.g. searching for a faculty name in a DB).

While the matching of a NE-value retrieved from the *NE-Table*, with other NEs in the DB is performed through exact value match, the actual retrieval of that NE from the *NE-Table* happens using attention in the neural embedding space (using a dot product in our experiments). This allows the training of the *NE-Retrieval Module* using the supervision obtained from the downstream module (e.g., a DB retrieval module) that uses the retrieved NE-value. This also provides supervision for training the *NE-Embedding Generation Module*. Our intuition is that, this would encourage the *NE-Embedding Generation Module* to generate embeddings for the NEs such that the embeddings have relevant and enough information to allow the *NE-Retrieval module* to attend and retrieve them correctly when required later.

Since the embeddings are generated on the fly using the context, the above method works equally well for new NEs that come during test time as it would for the NEs present in the training data. We show examples for *NE-Table* for the dialog and Reading Comprehension task in Figure 2. A new, separate *NE-Table* is created for each data instance based on the task. For example, in the dialog task, each dialog will have its own separate *NE-Table*. Only the NEs that have appeared in the dialog so far will be present in its corresponding *NE-Table*. The same NE occurring in different dialogs will have different dialog-context-dependent embeddings in their corresponding *NE-Table*. Similarly, for the reading comprehension task, each story will have a separate *NE-Table* with the NEs present in that story and for the QA task, each question will have a separate *NE-Table*. Note that, a NE that occurs multiple times in the same dialog/story/question will also have multiple unique embeddings in the *NE-Table* because of differing contexts as shown in Figure 2 (right).

3 Experiments and Results

We evaluate our proposed method on three types of tasks: a reading-comprehension task, a structured-QA task and three goal-oriented dialog tasks. Our proposed method is generic and can be added to the state-of-the-art approaches for these tasks. But instead of implementing 3 separate specialized neural architectures, we chose the end-to-end memory network architecture from Sukhbaatar et al. (2015) as the base architecture for our tasks. This allows us to evaluate the advantage gained by adding our method to the base architecture instead of trying to get state-of-the-art performance in a particular task/dataset.

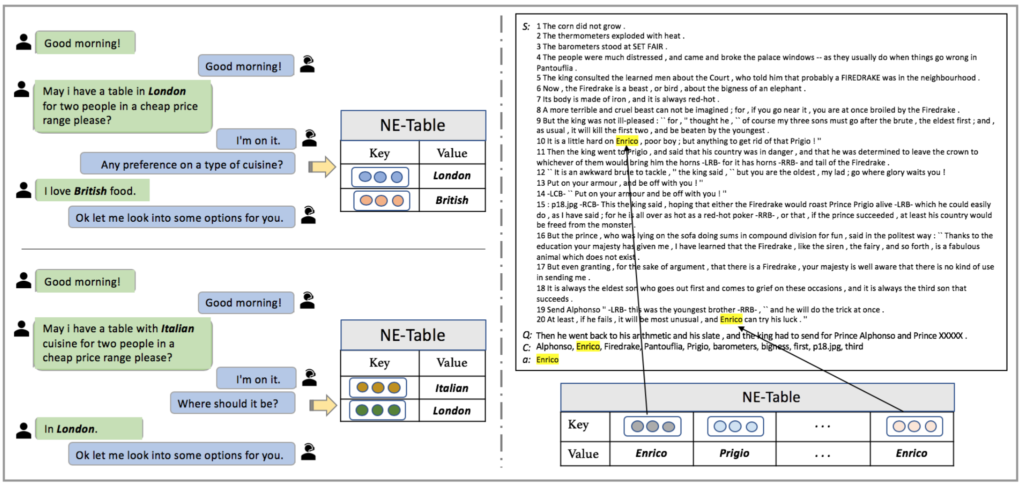
Model	Validation	Test
W/O-NE-Table (BoW)	49.55	41.69
W/O-NE-Table (LSTM)	49.40	41.10
With-NE-Table (BoW)	57.05	51.28
With-NE-Table (LSTM)	55.75	51.08

Table 1: Results (accuracy %) on CBT-NE dataset

3.1 Reading Comprehension Task

The Children's Book Test dataset (CBT), built from children's books from ProjectGutenberg, was introduced by Hill et al. (2015) to test the role of memory and context in language processing and understanding. Questions are formed by enumerating 21 consecutive sentences, where the first 20 sentences form the *story* (S), and a *word* (a) is removed from the 21st sentence, which then implicitly becomes the *query* (q). The specific task is to predict the correct answer word (a) from a set of 10 candidate words (C) present in the story or the query. We test our method on the NE questions subset of the CBT dataset.

We use the Window memory architecture proposed by Hill et al. (2015) for the CBT dataset as our baseline. In Memory Networks (Sukhbaatar et al., 2015), each complete sentence of S is encoded and represented in a separate memory slot. For the CBT, this setting would yield exactly 20 memories for S. In Window memory, instead of a full sentence from the story, a phrase is encoded and represented in a separate memory slot. Each phrase s corresponds to a window of text from the story S centred on an individual mention of a candidate c in S. The window is constructed as span of words $w_{i-(b-1)/2}$ ... w_i ... $w_{i+(b-1)/2}$ where b is window size and $w_i \in C$ is an instance of one of the candidate words in the question. We perform two baseline evaluations: encoding the windows using a) Bag-of-Words (BoW) and b) LSTM (Hochreiter and Schmidhuber, 1997).



Good morning!

Good morning!

May i have a table in ***London*** for two people in a cheap price range please?

I'm on it.

Any preference on a type of cuisine?

I love ***British*** food.

Ok let me look into some options for you.

Good morning!

Good morning!

May i have a table with ***Italian*** cuisine for two people in a cheap price range please?

I'm on it.

Where should it be?

In ***London***.

Ok let me look into some options for you.

S: 1 The corn did not grow .
2 The thermometers exploded with heat .
3 The barometers stood at SET FAIR .
4 The people were much distressed , and came and broke the palace windows -- as they usually do when things go wrong in Pantouflia .
5 The king consulted the learned men about the Court , who told him that probably a FIREDRAKE was in the neighbourhood .
6 Now , the Firedrake is a beast , or bird , about the bigness of an elephant .
7 Its body is made of iron , and it is always red-hot .
8 A more terrible and cruel beast can not be imagined ; for , if you go near it , you are at once broiled by the Firedrake .
9 But the king was not ill-pleased : `` for , '' thought he , `` of course my three sons must go after the brute , the eldest first ; and , as usual , it will kill the first two , and be beaten by the youngest .
10 It is a little hard on Enrico , poor boy ; but anything to get rid of that Prigio ! ''
11 Then the king went to Prigio , and said that his country was in danger , and that he was determined to leave the crown to whichever of them would bring him the horns -LRB- for it has horns -RRB- and tail of the Firedrake .
12 `` It is an awkward brute to tackle , '' the king said , `` but you are the oldest , my lad ; go where glory waits you !
13 Put on your armour , and be off with you ! ''
14 -LCB- `` Put on your armour and be off with you ! ''
15 : p18.jpg -RCB- This the king said , hoping that either the Firedrake would roast Prince Prigio alive -LRB- which he could easily do , as I have said ; for he is all over as hot as a red-hot poker -RRB- , or that , if the prince succeeded , at least his country would be freed from the monster .
16 But the prince , who was lying on the sofa doing sums in compound division for fun , said in the politest way : `` Thanks to the education your majesty has given me , I have learned that the Firedrake , like the siren , the fairy , and so forth , is a fabulous animal which does not exist .
17 But even granting , for the sake of argument , that there is a Firedrake , your majesty is well aware that there is no kind of use in sending me .
18 It is always the eldest son who goes out first and comes to grief on these occasions , and it is always the third son that succeeds .
19 Send Alphonso '' -LRB- this was the youngest brother -RRB- , `` and he will do the trick at once .
20 At least , if he fails , it will be most unusual , and Enrico can try his luck . ''

Q: Then he went back to his arithmetic and his slate , and the king had to send for Prince Alphonso and Prince XXXXX .
C: Alphonso, Enrico, Firedrake, Pantouflia, Prigio, barometers, bigness, first, p18.jpg, third
a: Enrico

Figure 2: *Left:* **Two dialogs from bAbI task-1**. A user (in green) chats with a dialog system (in blue) to book a table. Each dialog has its own separate *NE-Table* and a separate NE-Embedding is generated for the NE *London* though it appears in both dialogs. *Right:* **Question from CBT** . NE *Enrico* (highlighted in yellow) occurs twice in the context *S*, where a separate NE-Embedding is generated for each occurrence.

For each NE[2], the corresponding window is fed to an LSTM to create the context embedding. The context embedding is used as input to *NE-Embedding Generation Module* (f_ϕ), as shown in Figure 1, to generate the corresponding NE-Embedding, which is added to the *NE-Table*. The NE-Embeddings are also added to window memory, in place of the NEs. The query (q) embedding is used to attend over the memory (list of encoded window memory slots) to get relevant information from the memory. The internal state generated is given as input to the *NE-Retrieval Module* (g_θ), for retrieving the correct NE answer (a). Table 1 shows that replacing the baseline with our method achieves higher performance on both BoW and LSTM baseline models, across both validation and test sets. We use a window size of 5 as in Hill et al. (2015). We think that since the window size is small, both BoW and LSTM models perform similarly. We provide model training and hyper-parameter details in the Appendix.

To further evaluate the impact of OOV NEs, we created additional OOV test sets by replacing NEs in the test set with new NEs not present in the train and validation sets. We generate 5 such OOV test sets with varying percentage of OOV NEs (20%, 40%, 60%, 80% and 100%). Figure 3 shows the comparison of our model with the baselines on OOV test-sets. The baseline models perform poorly as OOV% increases, decreasing to as low as 5% from 41%. We observe only a slight reduction in accuracy for the NE-Table models from 51% to 46% because the new entities are also part of the windows, used to generate NE-Embeddings. These experiments illustrate that our model performance is robust to OOV NEs.

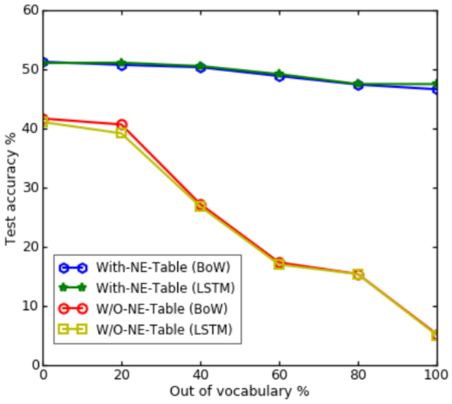


Figure 3: Results on CBT-NE OOV test sets

The next two tasks, structured-QA and goal-oriented dialog involve retrieval from an external DB. This is performed by the *DB-Retrieval Module* (h_ψ), which uses a multiple-attention based neural retrieval mechanism. We describe this next and then present results on the 2 tasks.

[2] The NEs present in the story are identified by the Stanford Core NLP NER system (Manning et al., 2014).

3.2 Multiple-Attention Based Neural Retrieval Mechanism

In both structured-QA and goal-oriented dialog tasks, the external information is present in a single database table, where each row corresponds to a new entity of interest and the columns correspond to the different attributes associated with it. For example, in our structured-QA (which is about course offerings at a university) DB, each row corresponds to a course and the columns correspond to course attributes, such as course number, course name, instructor name, etc. Each column of the table has a column heading, which labels the attribute of that column. These headings are also part of the vocabulary. While the non-NEs present in the DB are part of the vocabulary and represented by their learned neural embeddings, the NEs are not part of the vocabulary and are represented by their exact values.

The *DB-Retrieval Module* performs attention over both attributes(columns) as well as rows to select the final cell(s) in 3 steps. In step 1, the column(s) that the final cell(s) belong to are selected by attention over the column heading embeddings. For the question *Who teaches EECS545?*, step 1 selects the column *'instructor name'*. In step 2, separate attention is performed over the column headings to select the columns, which are used to represent the rows (to retrieve the final cell) and column *'course number'* is selected. Step 3 is to do attention over the rows. For each non-NE column selected in step 2, the cell embeddings are added together along each row, to generate an embedding for each row. Attention is performed over these row embeddings to select row(s). For each NE-column selected in step 2, a NE-value is retrieved from the *NE-Table* to do an exact match search over that NE-column to select matching row(s). The intersection of these matching row(s) gives the final set of selected row(s), and their intersection with the set of column(s) selected in step 1 gives the retrieved cell(s). For our example, only one column is selected to represent the rows: *'course number'*, which is a NE-column. Therefore, a NE value is retrieved from the *NE-Table* (EECS545) and an exact match search is done over the *'course number'* column.

The input to the *DB-Retrieval Module* depends on the task. For the dialog task, the dialog state vector is used, which has the information of the dialog so far. For the QA task, the encoding of the input question is used. All the attention operations in our experiments are performed through dot product followed by a sigmoid operation, which allows for multiple selections. Additional details and further explanation of the retrieval mechanism with examples are provided in Appendix. Note that *NE-Table* can potentially be used with other neural retrieval mechanisms. The multiple-attention mechanism described above is only one of the several neural retrieval mechanisms (Yin et al., 2015).

3.3 Structured-QA from DB

The task here is to retrieve an answer (single cell in a table) from DB in response to structured one line questions. We used the details of course offerings at a university to create structured Question-Answer (QA) pairs. The DB is a single table of 100 rows (Courses) and 4 columns (Course Number, Course Name, Department, Credits)[3], where course numbers and course names are treated as NEs. The QA pairs are generated through random sampling from the DB, following the format -

```
Q: Col-1-type Col-1-value Col-2-type ?
A: Col-2-value
```

with the following being a specific example:

```
Q: Course Number EECS545 Credits ?  A: 4.
```

500 QA pairs were created and split randomly between training and test set (400-100), where the random split results in some NEs (OOV) in the test set, not present in the training set. The task was specifically constructed to be simple to show the impact of OOV NEs on the model performance and evaluate our proposed method.

The experiments were performed with two models. Both models use a Recurrent Neural Network (RNN) to encode the question and use the multiple-attention based neural retrieval mechanism to retrieve answers. The baseline model (*W/O-NE-Table*) does not distinguish NEs from normal words, and all words (including NEs) that occur in questions and DB are part of the vocabulary. The *With-NE-Table* model uses our proposed method and builds *NE-table* (course numbers and course names are the NEs in this task). In the *With-NE-Table* model, when a NE word is encountered, the hidden state of the RNN at the previous time step (word) is used as input to the *NE-Embedding Generation Module* (f_ϕ). The

[3] number of unique course numbers - 100, unique course names - 96, unique dept names - 10 and unique credits - 4

Model	Retrieval accuracy (%)
W/O-NE-Table	81.0
With-NE-Table	100.0

Table 2: Results on structured-QA task

NE-Embedding generated by f_ϕ is then fed back to the RNN to continue encoding the question. The generated NE-Embedding is also stored in the *NE-Table* associated with this question. The final hidden state of the RNN obtained after encoding the full question is provided as input to the *DB-Retrieval Module* (h_ψ).

For our example, both models perform attention over the column headings to identify the correct column *Credits* required for the answer. Then, both models attend over column headings to find the column *Course Number* used for representing the rows. For *W/O-NE-Table* model, since all course numbers are part of vocabulary, each row is represented by neural embeddings associated with course numbers and attention is done over the row embeddings. For *With-NE-Table* model, since course numbers are NEs, each row is represented with exact course number values. A neural attention over *NE-Table* is performed to return the NE value, *EECS545*, which is then used to perform an exact match with the course number values. We provide model training details in Appendix.

Table 2 shows the retrieval accuracy for both models. While the test accuracy for *With-NE-Table* is 100%, it drops to 81% for *W/O-NE-Table* model. Further analysis shows that out of the 19% drop, 11% is due to OOV NEs encountered at test time. These OOV NEs are in the DB, and hence are part of the vocabulary for the *W/O-NE-Table* model, but have random embeddings which did not change during the training time (as they were never encountered during the training). The rest 8% drop can be attributed to the model's inability to learn good embeddings for unique NEs that were rarely seen during training. However, these issues do not pose a problem for our *With-NE-Table* model, since we generate embedding for a NE on the fly for each question based on the context. This solves both problems: a) whether an NE occurred rarely or b) it was not present in training data at all. The *With-NE-Table* model should also easily scale to large datasets with any number of NEs without drop in performance.

3.4 Goal-Oriented Dialog Tasks

The Dialog bAbI tasks dataset was introduced by Bordes and Weston (2016) as a testbed to break down the strengths and shortcomings of end-to-end goal-oriented dialog systems The task domain is restaurant reservation and there are 5 tasks - Task 1: Issuing API calls, Task 2: Updating API calls, Task 3: Displaying Options, Task 4: Providing extra information and Task 5: Conducting full dialogs (combination of tasks 1-4). The system is evaluated in a retrieval setting. At each turn of the dialog, the system has to select the correct response from a list of possible candidates.

In the original bAbI tasks, DB-Retrieval is bypassed by providing all possible system utterances with all combinations of information *pre*-retrieved from the DB in a large candidate response list. We extend the original testbed and propose a new testbed, which is closer to real-world restaurant reservation, by adding an actual external DB so that the system can also be tested on the ability to retrieve the required information from the DB. We evaluate our method on extended versions of task 1,2 and 4[4].

For our experiments, we use an end-to-end memory network similar to Bordes and Weston (2016), except that we encode sentences using an RNN, while they use BoW encoding. The encoded sentences, which are part of the dialog history, are stored in the memory and the query (last user utterance) embedding is used to attend over the memory to get relevant information from the memory. The generated internal state is used to select the candidate response, and is also given as input to the *DB-Retrieval Module* (h_ψ). The DB is used to identify the NEs along with their types (if a word is present in a NE-column in the DB it is a NE; the column where it appears gives its NE-type).[5]

The experiments are performed on two models:

- *W/O-NE-Table* model (the baseline model) - All input words including NEs are part of the vocabulary. For NEs, however, their embedding given to the sentence encoder RNN is the sum of the NE word embedding and the embedding associated with its NE-type.

[4] Task 3 requires to learn to sort. Bordes and Weston (2016) achieve close to 0% accuracy on it.Therefore, we decided to skip tasks 3 and 5 (task 5 includes task 3 dialogs) to focus on evaluating our proposed method.

[5] This simple method (based on exact match) though works for this dataset, is not very effective, as plural or abbreviated NEs will not match.

- *With-NE-Table* model (uses our proposed method) - When an NE is encountered in the dialog, the last hidden state of the RNN encoding the sentence is used as input to the *NE-Embedding Generation Module* (f_ϕ). The NE-Embedding generated is stored in the *NE-Table*. The generated NE-Embedding and the embedding associated with its NE-type are fed to the RNN.

Note that both the models have access to the information whether a given word is a NE and its NE-type. Supervision is provided for candidate response selection and all attention operations performed during *DB-Retrieval*, for both models.

3.4.1 Extended Dialog bAbI Tasks 1 and 2

In the original bAbI task 1, the conversation between the system and the user involves getting information necessary to issue an *api_call*. In task 2, the user can ask the system to update his/her preferences (cuisine, location etc.). The system has to take this into account and make an updated *api_call*. In our extended version, once the system determines that the next utterance is an *api_call*, the system also has to actually retrieve the restaurant details from the DB (rows) which match user preferences. The system is evaluated on having conversation with the user, issuing *api_call* and retrieving the correct information from DB. The DB is represented as a single table, where each row corresponds to a unique restaurant and columns correspond to attributes, e.g. cuisine, location etc.

Both *W/O-NE-Table* and *With-NE-Table* models, first select the four relevant (cuisine, location, price range and number of people) columns to represent each row (restaurant). The *W/O-NE-Table* model then selects the rows using attention over the row embeddings obtained through the combined (additive) representation of the four selected attributes. The *With-NE-Table* model splits the row selection into two simpler problems. For cuisine and location (which are NEs), one NE value each is retrieved from the *NE-Table* and an exact match is performed in the DB. The neural embeddings of the non-NE attributes (price range and number of people) are added to perform attention for selecting rows. The final retrieved rows are the intersection of the rows selected by NE column and non-NE column based selections.

The results for tasks 1 and 2 are shown in Table 3. The *With-NE-Table* model achieves close to 100% accuracy in both tasks, while *W/O-NE-Table* performs poorly. During DB retrieval, for the *With-NE-Table* model, two NEs are chosen from the *NE-Table* and exact matching is done over different cuisines and locations in the DB, but embeddings for these NEs are learned for *W/O-NE-Table*. This results in poor DB-Retrieval for *W/O-NE-Table* for less frequent/ OOV location/cuisine values. Both models perform well in *Per-dialog* accuracy as it does not involve DB retrieval[6]. The Per-Dialog accuracy is high for both models on the normal test set. However, for task 1 OOV-test set, *W/O-NE-Table* model is affected by OOV-NEs (90.3%), while *With-NE-Table* model performance is robust (99.0%).

Model	DB-Retrieval	Per-Dialog	Per-Dialog + DB-Retrieval
Task 1			
W/O-NE-Table	10.2 (7.0)	100 (90.3)	10.2 (6.7)
With-NE-Table	98.5 (99.0)	98.8 (99.0)	**97.3 (98.0)**
Task 2			
W/O-NE-Table	0.8 (1.0)	100 (100)	0.0 (0.1)
With-NE-Table	99.6 (99.8)	100 (99.9)	**99.2 (99.7)**
Task 4			
W/O-NE-Table	0.0 (0.0)	100 (100)	0.0 (0.0)
With-NE-Table	100 (100)	100 (100)	**100 (100)**

Table 3: Results for extended bAbI tasks 1, 2 and 4. % accuracy for Test and Test-OOV (given in parenthesis). *DB-Retrieval* : Retrieval accuracy for rows (task 1,2 - *all* restaurants matching user preferences) and a particular cell (task 4 - restaurant phone number/address). *Per-Dialog* : Percentage of dialogs where every dialog response is correct. Training details and hyperparameter values are provided in Appendix.

3.4.2 Extended Dialog bAbI Task 4

The original task 4 starts at the point where a user has decided a particular restaurant. The system is given information (location, phone number, address etc.) about *only* that restaurant as part of the dialog history and the user can ask for its phone number, address or both. For a given user request e.g. address, the task is to select the correct response with the restaurant's address from a list of candidate responses. These candidate responses have phone number and address information for all the restaurants mentioned in the DB.

In our extended version, even though the user

[6] The system responses in tasks 1/2/4 do not contain any NEs, but the system still needs to understand user utterances which might have NEs.

has decided a particular restaurant, its corresponding information is not provided as part of dialog history. This makes the task harder but more realistic. Now, the system needs to search for phone number/address for the restaurant from the full DB while in the original task, the phone number/address is already provided as part of dialog history. In the extended version, the NEs in candidate responses are replaced with their NE-type tags. For example, *Suvai_phone* is replaced with *NE_phone*. The system has to select the candidate with correct NE-type tag and then replace the tag with the actual NE-value retrieved from the DB, similar to Williams et al. (2017). This setting is closer to how a human agent would do this task.

For *With-NE-Table* model, the restaurant name that appears in the dialog would be stored in *NE-Table*. When the user asks for information such as phone number, the restaurant name stored in *NE-Table* is selected and used for retrieving its phone number from the DB. In *W/O-NE-Table* model, all input words (including NEs) are part of vocabulary and phone number is selected by neural embedding attention over all restaurants names.

The results for task 4 are shown in Table 3. We observe that both models perform well in Per-dialog accuracy. The *W/O-NE-Table* model fails in DB-retrieval (0%) because it needs to learn neural embeddings for all restaurant names, while *With-NE-Table* performs well (100%) as it uses our proposed method to generate NE-Embeddings on the fly and use the actual NE values later for exact value matching over restaurant names in the DB.

4 Related Work

NE in QA: Neelakantan et al. (2015) and Yin et al. (2015) transform a natural language query to a program that could run on DBs, but those approaches are only verified on small or synthetic DBs. Other papers dealing with large Knowledge Bases (KB) usually rely on entity linking techniques (Cucerzan, 2007; Guo et al., 2013), which links entity mentions in texts to KB queries. Recently, Liang et al. (2016) extended end-to-end neural methods to QA over KB, which could work for large KB and large number of NEs. However, their method still relies on entity linking to generate a short list of entities linked with text spans in the questions, in advance. Yin et al. (2015) propose 'Neural Enquirer', a neural network architecture similar to the neural retrieval mechanism used in this work, to execute natural language queries on DB. They keep the randomly initialized embeddings of the NEs fixed as a method to handle NEs and OOV words.

NE in Dialog: There has been a lot of interest in end-to-end training of dialog systems (Vinyals and Le, 2015; Serban et al., 2016; Lowe et al., 2015; Kadlec et al., 2015; Shang et al., 2015; Guo et al., 2017). Among recent work, Williams and Zweig (2016) use an LSTM model that learns to interact with APIs on behalf of the user; Dhingra et al. (2017) use reinforcement learning to build the KB look-up in task-oriented dialog systems. But the look-up actions are defined over each entity in the KB and is therefore hard to scale up. Most of these papers actually do not discuss the issue of handling NEs though they are present. Williams et al. (2017) propose Hybrid Code Networks and achieve state-of-the-art on Facebook bAbI dataset, but approach involves a developer writing domain-specific software components.

NE in Reading Comprehension and Others: For certain tasks such as Machine Translation and summarization, neural copying mechanisms (Gulcehre et al., 2016; Gu et al., 2016) have been proposed for handling OOV words. Our *NE-Table* method can be used along with such copying mechanisms for cases like dialog generation.

5 Conclusion and Future work

In this paper, we proposed a novel method for handling NEs in neural settings for NLP tasks. Our experiments on the CBT dataset illustrate that the models with *NE-Table* perform better than models without *NE-Table*, and clearly outperform the baseline models on the OOV test sets. We observe similar results for our experiments on the structured-QA task and goal-oriented bAbI dialog tasks. We also show that our method can be used for NEs in the external DB provided. Overall, these experiments show that the proposed method can be useful for various NLP tasks where it is beneficial to work with actual NE values, and/or it is hard to learn good neural embeddings for NEs.

In future, we are interested in testing the proposed method with retrieval mechanisms such as 'Neural Enquirer' (Yin et al., 2015), which can work with multiple tables. We are also interested in exploring the use of pre-trained embeddings: word2vec (Mikolov et al., 2013), ELMo (Peters et al., 2018) etc., to bootstrap our learned NE-embeddings. We are also interested in evaluating our proposed method on tasks that are more unstructured and requires more free-form generation, e.g. machine translation and dialog generation.

References

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the ACL.*

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 363–370.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR* abs/1603.06393. http://arxiv.org/abs/1603.06393.

C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaoxiao Guo, Tim Klinger, Clemens Rosenbaum, Joseph P Bigus, Murray Campbell, Ban Kawas, Kartik Talamadupula, Gerry Tesauro, and Satinder Singh. 2017. Learning to query, reason, and answer questions on ambiguous texts .

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

R. Kadlec, M. Schmid, and J. Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. In *Proc. of NIPS-15 Workshop on "Machine Learning for SLU and Interaction"*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020* .

R. Lowe, N. Pow, I. Serban, and J. Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proc. of SIGDIAL-2015*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834* .

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

I. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. of AAAI-2016*.

L. Shang, Z. Lu, and H. Li. 2015. Neural responding machine for short-text conversation. In *Proc. of ACL-2015*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *NIPS*.

O. Vinyals and Q. Le. 2015. A neural conversational model. *ICML, Workshop* .

J. D. Williams and G. Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* .

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965* .

A Model Training and Hyperparameter Details

A.1 Reading Comprehension - CBT

The hyperparameters used for baseline *W/O-NE-Table* models are as follows: hops = 1, embedding_size = 100, batch_size = 16. The *W/O-NE-Table (LSTM)* model uses an LSTM for encoding windows with hidden_units = 100. The entire model is trained using stochastic gradient descent (learning rate = 0.05), minimizing a standard cross-entropy loss between predicted answer $\hat{a}$ and the correct answer a. We use the same embedding matrix for encoding both story and the query.

For our *With-NE-Table* models, we use the same hyper-parameters as mentioned above. The *With-NE-Table (LSTM)* model uses separate LSTM cells for a) generating the context embedding for *NE-Embedding Generation Module* (f_ϕ) and b) encoding the windows.

A.2 Structured-QA

Both the *With-NE-Table* and *W/O-NE-Table* models use the following hyperparameters : embedding_size = 20, batch_size = 16. The RNN used for encoding the sentences has hidden_units = 20. The model is trained using Adam (learning rate = 0.01, epsilon = 1e-8). Supervision is provided for DB-Retrieval attentions and standard cross-entropy loss is used.

A.3 bAbI dialog tasks

Both the *With-NE-Table* and *W/O-NE-Table* models use the following hyperparameters : hops = 3, embedding_size = 40, batch_size = 16. The RNN used for encoding the sentences has hidden_units = 40. The model is trained using Adam (learning rate = 0.001, epsilon = 1e-8). Supervision is provided for both the candidate response selection and DB-Retrieval attentions and standard cross-entropy loss is used.

B Extended dialog bAbI tasks' examples

B.1 Original dialog bAbI task 1:

U: Hi !

S: Hello, what can I help you with today?

U: I'd like to book a table for six people in an expensive price range with British food.

S: I'm on it.

S: Where should it be?

U: In London.

S: Ok, let me look into some options for you.

S: api_call British, London, Six, Expensive

B.2 Extended dialog bAbI task 1:

U: Hi !

S: Hello, what can I help you with today?

U: I'd like to book a table for six people in an expensive price range with British food.

S: I'm on it.

S: Where should it be?

U: In London.

S: Ok, let me look into some options for you.

S: api_call

S (Retrieved Information):

{The_Place, british, london, ...}

{The_Fancy_Pub, british, london, ...}

B.3 Original dialog bAbI task 2:

U: Hi !

S: Hello, what can I help you with today?

U: I'd like to book a table for six people in an expensive price range with British food in London.

S: Ok, let me look into some options for you.

S: api_call British, London, Six, Expensive

U: Instead, could it be Indian cuisine.

S: Sure, is there anything else to update?

U: No.

S: Ok, let me look into some options for you.

S: api_call Indian, London, Six, Expensive

B.4 Extended dialog bAbI task 2:

U: Hi !

S: Hello, what can I help you with today?

U: I'd like to book a table for six people in an expensive price range with British food in London.

S: Ok, let me look into some options for you.

S: api_call

S (Retrieved Information):

{The_Place, british, london,...}
The_Fancy_Pub, british, london, ...}

U: Instead, could it be Indian cuisine.

S: Sure, is there anything else to update?

U: No.

S: Ok, let me look into some options for you.

S: api_call

S (Retrieved Information):

{Suvai, Indian, london, ...}
{Madras_Masala, Indian, london, ...}

B.5 Original dialog bAbI task 4:

DB Information given :
Name:The_Place, Cuisine:British,
Location:London, Price Range:Expensive,
Rating:7, Phone:The_Place_phone,
Address:The_Place_address, Availability:six

U: Hello

S: Hello, what can I help you with today?

U: I'd like to book a table at The_Place.

S: Great, let me do the reservation.

U: May I have the address of the restaurant.

S: Here it is, The_Place_address.

B.6 Extended dialog bAbI task 4:

DB Information given : Entire DB

U: Hello

S: Hello, what can I help you with today?

U: I'd like to book a table at The_Place.

S: Great, let me do the reservation.

U: May I have the address of the restaurant.

S: Here it is, NE_address.

S: The_Place_address *(*retrieved information)

C Multiple-attention based neural retrieval mechanism

Figure 4 shows the schematic of the entire retrieval process. In order to retrieve a particular cell from the table, the system needs to find the correct column and row corresponding to it. The *DB-Retrieval Module* (h_ψ) does that by generating 3 different attention key embeddings (vectors): Attention over Columns for Columns (*ACC*), Attention over Columns for Rows (*ACR*), Attention over Rows for Rows (*ARR*).

The column(s) that the final retrieved cell(s) belong to, are selected by matching *ACC* key embeddings with the neural embeddings of the column headings (Course Number, Instructor, Credits etc). A separate *ACC* key embedding is generated for every column heading and matched with its embeddings to provide attention scores for all the columns. For the example, *Who teaches EECS545?*, the system would want to retrieve the name of the *Instructor*. Therefore, the *Instructor* column heading alone will have high attention score and be selected. In our experiments, the attention scores are computed through dot products followed by a sigmoid operation, which allows for multiple selections.

Now that the column(s) are chosen, the system has to select row(s), so that it can get the cell(s) it is looking for. Each row in the table contains the values (EECS545, Machine Learning, Scott Mathew etc) of several attributes (Course Number, Course Name, Instructor etc). But we want to assign attention scores to the rows based on particular attributes that are of interest (*Course Number* in this example). The column/attribute headings that the system has to attend to for selecting these relevant attributes are obtained by matching *ACR* (Attention over Columns for Rows) key embeddings with the neural embeddings of the different column headings.

The last step in the database retrieval process is to select the relevant rows using the *ARR* (Attention over Rows for Rows) key embedding. *ARR* is split into two parts *ARR NE* and *ARR non-NE*. In a general scenario, *ACR* can select multiple columns to represent the rows. For each selected column that is a NE column, a separate NE-value is retrieved from the *NE-Table* using a separate *ARR NE* embedding for each of them. These NE values are used to do exact match search along the corresponding columns (in the NE row representations) to select the matching rows. For the non-NE columns that are selected by *ACR*, their neural embeddings are combined together along each row to get a fixed vector representation for each row in the DB (weighted sum of their embeddings, weighted by the corresponding column attention

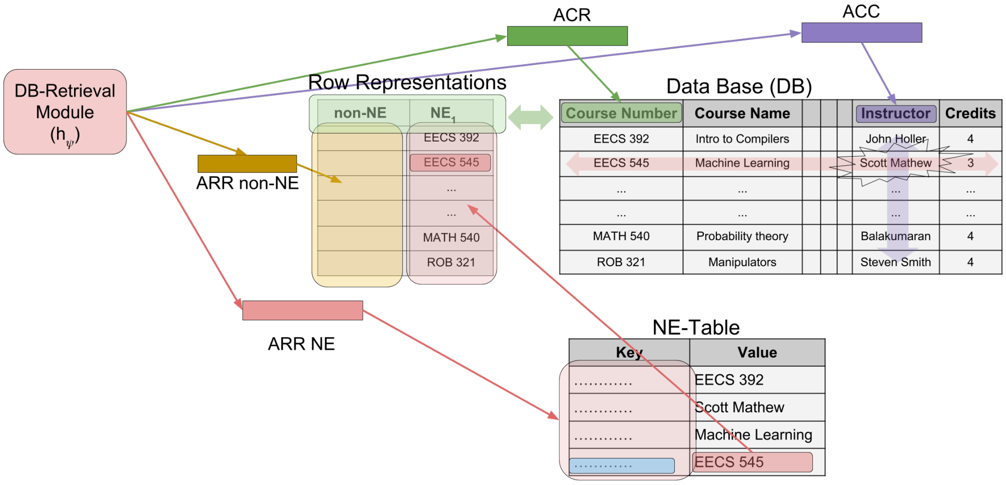


Figure 4: Multiple-attention based neural retrieval mechanism. The *DB-Retrieval Module* attends to the relevant rows and columns of the DB by generating attention key embeddings *ACC*, *ACR* and *ARR*.

scores). *ARR non-NE* is then used to match these representations for selecting rows. The intersection of the rows selected in the NE row representations and the non-NE row representations is the final set of selected rows.

In short, the dialog system can use neural embedding matching for non-NEs, exact value matching for NEs and therefore a combination of both to decide which rows to attend to. Depending on the number of columns and rows we match with, we select zero, one or more output cells. For our running example, *ARR NE* is used to match with the keys in the *NE-Table* to select the row corresponding to *EECS 545* and the value *EECS 545* is returned to do an exact match over the NE row representations (represented by the course number values). This gives us the row corresponding to *EECS 545* and hence the cell *Scott Mathew*. We could use our *NE-Table* idea with potentially many types of neural retrieval mechanisms to retrieve information from the DB. The multiple-attention based retrieval mechanism, described above, is only one such possible mechanism.

D Goal oriented dialog tasks: extended results

D.1 Extended results for tasks 1 and 2

The detailed results for task 1 and task 2 are shown in Table 4.

With-NE-Table: For issuing an *api_call* in tasks 1 and 2, four argument values are required - cuisine, location, price range and number of people. We consider cuisine and location to be NEs. So whenever cuisine and location names occur in the dialog, a NE key is generated on the fly and is stored in the *NE-Table* along with the NE values.

- ACC: For tasks 1 and 2, ACC is not required as we are interested in retrieving rows.
- ACR: ACR is used to select the columns required to represent the rows. These are four columns - NE columns (cuisine and location) and non-NE columns (price range and number of people)
- ARR-non-NE: Each row in the DB is represented by weighted vector (embedding) sum of its price range and number of people (embeddings). The model returns the relevant rows using attention on the non-NE columns embeddings.
- ARR-NE: The model attends over the *NE-Table* by matching (dot product) its generated key with the keys present in the *NE-Table* to retrieve NE values. The selected NE values are then matched (exact-match) with cuisine and location values in DB to retrieve the relevant rows.
- The final retrieved rows are the intersection of the rows selected by ARR-non-NE and ARR-NE.

W/O-NE-Table: *ACR* is used to attend to the four relevant columns. However, each row is rep-

Task	Model	ACR	ARR non-NE	ARR NE	DB-Retrieval	Per-response	Per-Dialog	Per-Dialog + DB-Retrieval
Task 1	W/O-NE-Table	100 (100)	9.0 (6.9)	-	10.2 (7)	100 (98.2)	100 (90.3)	10.2 (6.7)
	With-NE-Table	99.4 (98.1)	96.9 (96.7)	100,100 (100,100)	98.5 (99.0)	99.8 (99.8)	98.8 (99)	97.3 (98.0)
Task 2	W/O-NE-Table	100 (100)	8.6 (7.6)	-	0.8 (1.0)	100 (100)	100 (100)	0.0 (0.1)
	With-NE-Table	100 (100)	99.1 (99.8)	100,100 (100,100)	99.6 (99.8)	100 (100)	100 (100)	99.2 (99.7)

Table 4: Results for extended dialog bAbI task 1 and 2. Accuracy % for Test and Test-OOV (given in parenthesis). ARR non-NE columns are price and number of people. ARR NE columns are cuisine and location. DB-Retrieval %: Retrieval accuracy for rows (task 1,2) and a particular cell (task 4). Per-Dialog %: Percentage of dialogs where every dialog response is correct. Per-Dialog + DB-Retrieval %: Percentage of dialogs where every dialog response and information from DB retrieval are correct.

Model	ACR	ACC	ARR non-NE	ARR NE	DB-Retrieval	Per-response	Per-Dialog	Per-Dialog + DB-Retrieval
W/O-NE-Table	100 (100)	100 (100)	0.0 (0.0)	-	0.0 (0.0)	100 (100)	100 (100)	0.0 (0.0)
With-NE-Table	100 (100)	100 (100)	-	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)

Table 5: Results for extended dialog bAbI task 4. Accuracies in % for Test and Test Out-Of-Vocabulary (given in parenthesis). DB-Retrieval %: Retrieval accuracy for rows (task 1,2) and a particular cell (task 4). Per-Dialog %: Percentage of dialogs where every dialog response is correct. Per-Dialog + DB-Retrieval %: Percentage of dialogs where every dialog response and information from DB retrieval are correct.

resented by the combined neural embedding representation of all four attribute values, cuisine, location, price range and number of people. *ARR non-NE* is used to retrieve the relevant rows.

From Table 4, we can see that both the models perform well in selecting the relevant columns, but the model *W/O-NE-Table* performs poorly in retrieving the rows, while *With-NE-Table* performs very well. This results in *With-NE-Table* model achieving close to 100% accuracy in DB retrieval while *W/O-NE-Table* performs poorly.

This is because, in the *With-NE-Table* model, the task of retrieving rows is split into two simpler tasks. The NEs are chosen from the *NE-Table*, and then exact matching is used (which helps in handling OOV-NEs as well). The non-NEs, price range and number of people, have limited set of possible values (low, moderate or expensive for price range and 2,4,6 or 8 for number of people respectively). This allows the system to learn good neural embeddings for them and hence have high accuracy in *ARR non-NE*. Whereas in *W/O-NE-Table* model, *ARR non-NE* involves the neural representations of cuisine and location values as well, where a particular location and cuisine value will occur only a few number of times in the training dataset. In addition to that, new cuisine and location values can occur during the test time (Test OOV dataset, performance shown in parenthesis).

For the dialog part (which does not involve the DB retrieval aspect) of extended tasks 1 and 2, the system utterances do not have any NEs in them. However, the user utterances contain NEs (cuisine and location that the user is interested in) and so the system has to understand them in order to select the right system utterance. The accuracy in performing the dialog (by selecting responses from candidate set) is similar for both the models on the normal test set. However, in the OOV-test set, for task 1, where the system has to maintain the dialog state to track which attribute values have not been provided by the user yet, *W/O-NE-Table* model seems to get affected, while the *With-NE-Table* model is robust to that. While *W/O-NE-Table* gets a Per-Dialog accuracy of 90.3% in the OOV-test set, *With-NE-Table* is able to get 99%.

D.2 Extended results for task 4

Detailed results for task 4 are shown in Table 5.

With-NE-Table: In task 4, the user tells the system the restaurant in which he/she wants to book a table. The restaurant name, which is a NE, is stored in the *NE-Table* along with it's generated key. When the user asks for information about the restaurant such as, phone number, the NE restaurant name stored in the *NE-Table* is selected and used for retrieving its corresponding phone number from the DB. For this particular case, *ACC* attends over the column *Phone* and *ACR* attends over *Restaurant Name*. Since the column selected

Task	Model	Evaluation	Task 1	Task 2	Task 4
Original bAbI tasks	Baseline(MemN2N + match-type + RNN-encoding)	Per-Dialog	100 (100)	99.9 (50.6)	100 (100)
Extended bAbI tasks	With-NE-Table	Per-Dialog + DB-Retrieval	97.3 (98.0)	99.2 (99.7)	100 (100)

Table 6: Performance comparison of our model in the extended dialog bAbI tasks, with a baseline model in the original bAbI tasks. Accuracies in % for Test and Test Out-Of-Vocabulary (given in parenthesis).

by *ACR* is a NE column, the NE value (here the actual restaurant name given by the user) is retrieved using *ARR NE* from the *NE-Table*. The retrieved NE value is used to do an exact match over the DB column selected by *ACR* to select the rows. The cell that intersects the selected row and the column selected by *ACC* is returned as the retrieved information and used to replace the NE type tag in the output response.

W/O-NE-Table: Here, all input words (including NEs) are part of the vocabulary and for NEs, their embedding given to the sentence encoder is the sum of the NE word embedding and the embedding associated with its NE-type. The candidate response retrieval (dialog) is same as the above model and the column attentions are also similar. However, the models differ with respect to attention over rows. Since NEs are not treated special here, attention over rows happens through *ARR non-NE*. For this task, when *ACR* is selected correctly (restaurant name), each row will be represented by the neural embedding representation of its restaurant names. *ARR non-NE* generates a key to match these neural embeddings to attend to the row corresponding to the restaurant name mentioned by the user.

E Comparison with original dialog bAbI tasks

We choose the best model (MemN2N + match-type features) from (Bordes and Weston, 2016) (they use match-type features for dealing with entities) and update the baseline model by using RNN encoding for sentences (similar to *With-NE-Table*). Note that we achieve higher accuracy for our updated baseline model for original bAbI tasks than reported in (Bordes and Weston, 2016), which we attribute to the use of RNN for encoding sentences (they use BoW encoding).

For match-type features, (Bordes and Weston, 2016) add special words (*R_CUISINE*, *R_PHONE* etc.), for each KB entity type (cuisine, phone, etc.) to the vocabulary. The special word (e.g. *R_CUISINE*) is added to a candidate if a cuisine (e.g. Italian) appears in both dialog and the candidate. For each type, the corresponding type word is added to the candidate representation if a word is found that appears 1) as a KB entity of that type, 2) in the candidate, and 3) in the input or memory. For example, for a task 4 dialog with restaurant information about *RES1*, *only* one candidate *"here it is RES1_phone"* will be modified to *"here it is RES1_phone R_PHONE"*. Now, if the user query is for the restaurant's phone number, using match-type features essentially reduces the output search space for the model and allows it to attend to specific candidates better. Hence, match-type features can only work in a retrieval setting and will not work in a generative setting. Our *With-NE-Table* model will work in both retrieval and generative settings.

Table 6 compares the performance of the *With-NE-Table* model in the extended bAbI tasks with that of a baseline method on the original bAbI tasks. Note that extended dialog bAbI tasks require the dialog system to do strictly more work compared to the original dialog bAbI tasks. Though not a strictly fair comparison for our model, we observe that the performance of our *With-NE-Table* model in extended bAbI tasks is as good as the performance of updated baseline model in original bAbI tasks. In addition to that, for bAbI task 2 OOV test set, *With-NE-Table* model performance in the extended bAbI task, is actually much higher compared to the baseline model on the original bAbI task (99.7% vs 50.6%).

Enhancing Unsupervised Sentence Similarity Methods with Deep Contextualised Word Representations

Tharindu Ranasinghe, Constantin Orăsan and Ruslan Mitkov
Research Group in Computational Linguistics
University of Wolverhampton, UK
{t.d.ranasinghehettiarachchige, c.orasan, r.mitkov }@wlv.ac.uk

Abstract

Calculating Semantic Textual Similarity (STS) plays a significant role in many applications such as question answering, document summarisation, information retrieval and information extraction. All modern state of the art STS methods rely on word embeddings one way or another. The recently introduced contextualised word embeddings have proved more effective than standard word embeddings in many natural language processing tasks. This paper evaluates the impact of several contextualised word embeddings on unsupervised STS methods and compares it with the existing supervised/unsupervised STS methods for different datasets in different languages and different domains.

1 Introduction

Measuring Semantic Textual Similarity (STS) is calculating the degree of semantic equivalence between two snippets of text (Agirre et al., 2016). Earlier, STS tasks largely focused on similarity between short texts such as abstracts and product descriptions (Li et al., 2006; Mihalcea et al., 2006). Recently, STS tasks at the International Workshops on Semantic Evaluation (SemEval) focused on measuring STS between full sentence pairs. The introduction of competitive STS tasks led to the development of standard datasets like the SICK corpus (Bentivogli et al., 2016) and standardised the similarity score as a numerical value between 1 and 5 (Agirre et al., 2014).

Having a good STS metric is crucial for many natural language processing applications such as information retrieval (IR) (Majumder et al., 2016), text summarisation (Aliguliyev, 2009; Steinberger and Jezek, 2004), question answering (Mohler et al., 2011) and text classification (Rocchio, 1971). Semantic similarity also contributes to many semantic web applications like community extraction, ontology generation and entity disambiguation (Li et al., 2006), and it is also useful for Twitter search (Salton et al., 1997), where it is required to accurately measure semantic relatedness between concepts or entities (Xu et al., 2015). STS is not limited only to natural language processing. For example in Biomedical Informatics, it can be used to compare genes (Ferreira and Couto, 2010).

Given the growing importance of having a good STS metric and as a result of the SemEval workshops, researchers have proposed numerous STS methods. Most of the early approaches were based on traditional machine learning and involved heavy feature engineering (Béchara et al., 2015). With the advances of word embeddings, and as a result of the success neural networks have achieved in other fields, most of the methods proposed in recent years rely on neural architectures (Tai et al., 2015; Shao, 2017). Neural networks are preferred over traditional machine learning models as they generally tend to perform better than traditional machine learning models. They also do not rely on explicit linguistics features which have to be extracted before the ML model is learnt. Determining the best linguistic features for calculating STS is not an easy task as it requires a good understanding of the linguistic phenomenon and relies on researchers' intuition. In addition, calculating these features is usually not an easy task, especially for languages other than English. Therefore, in contrast to traditional ML methods, models based on neural networks can be easily applied to other languages.

However, the biggest challenge that the neural based architectures face when applied to STS tasks is the small size of datasets available to train them. As a result, in many cases the networks can-

Proceedings of Recent Advances in Natural Language Processing, pages 994–1003,
Varna, Bulgaria, Sep 2–4 2019.

not be trained properly. Given the amount of human labour required to produce datasets for STS, it is not possible to have high quality large training datasets. As a result researches working in the field have also considered unsupervised methods for STS. Recent unsupervised approaches use pretrained word/sentence embeddings directly for the similarity task without training a neural network model on them. Such approaches have used cosine similarity on sent2vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), Word Mover's Distance (Kusner et al., 2015), Doc2Vec (Le and Mikolov, 2014) and Smooth Inverse Frequency with GloVe vectors (Arora et al., 2017). While these approaches have produced decent results in the final rankings of shared tasks, they have also provided strong baselines for the STS task.

Word vectors are used to determine a representation of a sentence in approaches like Word Mover's Distance (Kusner et al., 2015) and Smooth Inverse Frequency (Arora et al., 2017). The main weakness of word vectors is that each word has the same unique vector regardless of the context it appears. For an example, the word ”play” has several meanings, but in standard word embeddings such as Glove (Pennington et al., 2014), FastText (Mikolov et al., 2018) or Word2Vec (Mikolov et al., 2013) each instance of the word has the same representation regardless of the meaning which is used. However, contextualised word embedding models such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) etc. generate embeddings for a word based on the context it appears, thus generating slightly different embeddings for each of its occurrence. The recent applications in areas such as question answering and textual entailment show that contextualised word embeddings perform better than the traditional word embeddings (Devlin et al., 2018).

This paper explores the performance of several contextualised word embeddings in three unsupervised STS methods - cosine similarity using average vectors, Word Mover's Distance (Kusner et al., 2015) and cosine similarity using Smooth Inverse Frequency (Arora et al., 2016). The rest of the paper is organised as follow. Section 2 contains information about the settings of the experiments carried out in this paper including the datasets employed here and the different contextualised word embedding models explored. Each of the contextualised word embedding models against each method are evaluated in Section 4. Further experiments are conducted on Spanish sentence similarity and Bio-medical sentence similarity to observe the portability of the model to other languages and domains in section 5. Section 6 would briefly describe the related work done for STS. The paper finishes with conclusions.

2 Settings of the Experiments

2.1 Data Sets

The experiments presented in this paper were carried out using several datasets which will be explained in next subsections. In order to prove the portability of the approaches, the proposed architectures were also tested on an English Biomedical STS dataset. In addition, the language independence of the method is tested by applying it to a Spanish STS dataset.

2.1.1 English-English STS Data Set

For the experiments carried out on English STS, we used the SICK dataset. (Bentivogli et al., 2016). The SICK data contains 9927 sentence pairs with a 5,000/4,927 training/test split which were employed in the SemEval tasks. Each pair is annotated with a relatedness score between 1 and 5, corresponding to the average relatedness judged by 10 different individuals. Table 1 shows a few examples from the SICK training dataset.

Sentence Pair	Similarity
1. A little girl is looking at a woman in costume. 2. A young girl is looking at a woman in costume.	4.7
1. A person is performing tricks on a motorcycle. 2. The performer is tricking a person on a motorcycle.	2.6
1. Someone is pouring ingredients into a pot. 2. A man is removing vegetables from a pot.	2.8
1. Nobody is pouring ingredients into a pot. 2. Someone is pouring ingredients into a pot.	3.5

Table 1: Example sentence pairs from the SICK training data

2.1.2 Spanish-Spanish STS Data Set

For the Spanish STS experiments we used the dataset provided for Spanish STS subtask in SemEval 2015 Task 2 (Agirre et al., 2015). The training set has 1250 sentence pairs annotated with a relatedness score between 0 and 4. There were two sources for test set - Spanish news and Spanish Wikipedia dump having 500 and 250 sentence pairs respectively. Both datasets were annotated with a relatedness score between 0 and 4. Table 2 shows few pairs of sentences with their similarity score. As can be seen, this dataset is significantly smaller than the English dataset presented in the previous section. The effect of this is discussed in more detail below.

Sentence Pair	Similarity
1. Ams, los misioneros apunten que los nmberos d’infectaos puen ser shasta dos o hasta cuatro veces ms grandess que los oficiales. 2. Los cadveres de personas fallecidas pueden ser hasta diez veces ms contagiosos que los infectados vivos.	0.6
1. Desde Colombia, el presidente Juan Manuel Santos dijo que convers por telfono con Humala sobre el tema y que entregara al detenido a las autoridades peruanas a ms tardar el viernes. 2. El presidente de Colombia, Juan Manuel Santos, haba anunciado horas antes que Orellana, que se encuentra detenido, ser entregado a las autoridades peruanas sentre hoy y maanas.	3.2
1. La polica abati a un canbal cuando devoraba a una mujer Matthew Williams, de 34 aos, fue sorprendido en la madrugada mordiendo el rostro de una joven a la que haba invitado a su hotel. 2. La polica de Gales del Sur mat a un canbal cuando se estaba comiendo la cara de una mujer de 22 aos en la habitacin de un hotel.	2
1. Ollanta Humala se rene maana con el Papa Francisco. 2. El Papa Francisco mantuvo hoy una audiencia privada con el presidente Ollanta Humala, en el Vaticano.	3

Table 2: Example sentence pairs from the Spanish STS training data

2.1.3 Bio-medical STS Data Set

In-order to see the performance of our baseline in a complete different domain we used the biomedical English STS dataset provided in Sogancioglu et al. (2017). The dataset comprises 100 sentence pairs, which were evaluated by five different human experts that judged their similarity and gave scores ranging from [0,4]. To represent the similarity between two sentences we took the average of these scores. Table 3 shows few examples in the dataset. A dataset as small as this one can not be used by to train a supervised ML method, requiring alternative approaches such as unsupervised methods.

Sentence Pair	Similarity
1. It has recently been shown that Craf is essential for Kras G12D-induced NSCLC. 2. It has recently become evident that Craf is essential for the onset of Kras-driven non-small cell lung cancer.	4
1. Up-regulation of miR-24 has been observed in a number of cancers, including OSCC. 2. In addition, miR-24 is one of the most abundant miRNAs in cervical cancer cells, and is reportedly up-regulated in solid stomach cancers.	3
1. These cells (herein termed TLM-HMECs) are immortal but do not proliferate in the absence of extracellular matrix (ECM) 2. HMECs expressing hTERT and SV40 LT (TLM-HMECs) were cultured in mammary epithelial growth medium (MEGM, Lonza)	1.4
1.The up-regulation of miR-146a was also detected in cervical cancer tissues. 2. Similarly to PLK1, Aurora-A activity is required for the enrichment or localisation of multiple centrosomal factors which have roles in maturation, including LATS2 and CDK5RAP2/Cnn.	0.2

Table 3: Example sentence pairs from the Bio-medical dataset

2.2 Contextualised Word Representations

In order to use words in machine learning models, words have to be represented with a numerical form. Over the years researches have used many word representations like bag of words, one hot encoded vectors etc. But the recent neural models like word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014) provide better representations to the words considering its context too. We call them *standard word representations* in this research. Their main weakness is that every word has a unique word embedding regardless of the context it appears. As an example the word ’bank’ in two sentences - “I am walking by the river bank” and “I deposited money to the bank” would have the same embeddings which can be confusing for machine learning models. The recent introduction of contextualised word representations solved this problem by providing vectors for words considering their context too. In this way the word ’bank’ in above sentences have two different embeddings. As a result, contextualised word embeddings perform better than standard word embeddings in many natural language processing tasks like question answering, textual entailment etc. (Devlin et al., 2018). The following contextualised words representation models were considered for the experiments.

2.2.1 ELMo

ELMo introduced by Peters et al. (2018) use bidirectional language model (biLM) to learn both word (e.g., syntax and semantics) and linguistic context. After pre-training, an internal state of vectors can be transferred to downstream natural language processing tasks. We used the ’original’ pre-trained model provided in Peters et al. (2018) which was trained on the 1 Billion Word Benchmark (Chelba et al., 2013), approximately 800M tokens of news crawl data from WMT 2011. Using the model we represented each word as a vector with a size of 3072 values.

2.2.2 BERT

BERT was introduced in Devlin et al. (2018). It is based on a bidirectional transformer architecture rather than a unidirectional transformer used in Open AI GPT (Radford et al., 2019). In contrast to ELMo which uses a shallow concatenation layer (Devlin et al., 2018), BERT employs a deep concatenation layer. As a result BERT is considered a very powerful embedding architecture. We used pre-trained 'bert-large-uncased' model and represented each word as a 4096 lengthened vector.

2.2.3 Stacked Embeddings

Stacked Embeddings are obtained by concatenating different embeddings. According to Akbik et al. (2019) stacking the embeddings can provide a powerful embeddings to represent words. We represent the stacked embeddings in section 4 with '+' between the used models. As an example if the model name says ELMo + BERT, it is a stacked embedding of ELMo and BERT. For ELMo + BERT model we used pre-trained 'bert-large-uncased' model and 'original' pre-trained ELMo model to represent each word as a 4096 + 3072 vector.

2.2.4 Flair

Flair is another type of popular contextualised word embeddings introduced in Akbik et al. (2018). It takes a different approach by using a character level language model rather than the word level language model used in ELMo and BERT. The recommended way to use Flair embeddings is to stack pre-trained 'news-forward' embeddings and pre-trained 'news-backward' embeddings with Glove (Pennington et al., 2014) word embeddings (Akbik et al., 2018). We used the stacked model to represent each word as a 4196 lengthened vector.

2.3 Standard Word Representations

In order to compare the results of contextualised word embeddings, we used a standard word representation model in each experiment as a baseline. In this research we used word2vec embeddings (Mikolov et al., 2013) pre-trained on Google news corpus. We represented each word as a 300 lengthened vector using this model.

3 Experiments

This section describes the actual methods used to calculate the STS score between a pair of sentences and their variants we used. Each experiment was conducted using all three contextualised word embedding models - ELMo, BERT and Flair and one standard word representation model - word2vec (Mikolov et al., 2013).

3.1 Cosine Similarity on Average Vectors

The first unsupervised STS method that we used to estimate the semantic similarity between a pair of sentences, takes the average of the word embeddings of all words in the two sentences, and calculates the cosine similarity between the resulting embeddings. This is a common way to acquire sentence embeddings from word embeddings. Obviously, this simple baseline leaves considerable room for variation. We have investigated the effects of ignoring stopwords and computing an average weighted by tf-idf in particular and reported them in the 4 section.

3.2 Word Mover's Distance

The second baseline that we have considered is Word Mover's Distance introduced by Kusner et al. (2015). Word Mover's Distance uses the word embeddings of the words in two texts to measure the minimum distance that the words in one text need to "travel" in semantic space to reach the words in the other text as shown in Figure 1. Kusner et al. (2015) says that this is a good approach than vector averaging since this technique keeps the word vectors as it is through out the operation. We have investigated the effects of considering/ ignoring stop words before calculating the word mover's distance.

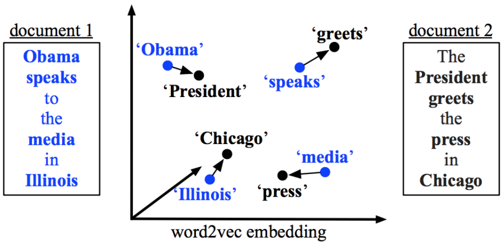


Figure 1: The Word Mover's Distance between two documents

3.3 Cosine Similarity Using Smooth Inverse Frequency

The third and the last unsupervised STS method we have considered is to acquire sentence embeddings using Smooth Inverse Frequency proposed by Arora et al. (2016) and then calculate the cosine similarity between those sentence embeddings. Semantically speaking, taking the average of the word embeddings in a sentence tends to give too much weight to words that are quite irrelevant. Smooth Inverse Frequency tries to solve this problem in two steps.

1. Weighting: Smooth Inverse Frequency takes the weighted average of the word embeddings in the sentence. Every word embedding is weighted by $\frac{a}{a+p(w)}$, where a is a parameter that is typically set to 0.001 and $p(w)$ is the estimated frequency of the word in a reference corpus.

2. Common component removal: After that, Smooth Inverse Frequency computes the principal component of the resulting embeddings for a set of sentences. It then subtracts their projections on first principal component from these sentence embeddings. This should remove variation related to frequency and syntax that is less relevant semantically.

As a result, Smooth Inverse Frequency downgrades unimportant words such as *but, just*, etc., and keeps the information that contributes most to the semantics of the sentence. After acquiring the sentence embeddings for a pair of sentences, the cosine similarity between those two vectors were taken to represent the similarity between them.

4 Evaluation on English SemEval Data

This section describes the evaluation results of English SemEval data for all the unsupervised STS methods we described above.

All the experiments were evaluated using the three evaluation metrics normally employed in the STS tasks: Pearson correlation (τ), Spearman correlation (ρ) and Mean Squared Error (MSE). Following sub-sections will discuss the results in detail.

4.1 Cosine Similarity on Average Vectors

Vector averaging results are shown in Table 4. Since we calculated the similarity as the cosine similarity between two vectors our predicted similarity lies between ∈ [0,1]. Since the GOLD standards are between ∈ [1,5] we re-scaled the predictions to be ∈ [1,5] in order to allow comparison.

Following variations were considered and reported in each sub-table.

1. All the word vectors were considered for averaging. Results are shown in table 4a.

2. All the word vectors except the vectors for stop words were considered for averaging. Table 4b shows the results.

3. All the word vectors were weighted from its tf-idf scores and considered averaging. Results are shown in table 4c

4. Stop words were removed first and remaining word vectors were weighted from its tf-idf scores and considered averaging. Table 4d shows the results.

As shown in table 4 the contextualised word vectors did not perform better than the standard word embeddings in all the variations. The only model that came close to word2vec performance was ELMo. All the contextualised word embedding models we considered have more than 3000 dimensions for the word representation which is significantly higher than the number of dimensions for the word representation we had for standard embeddings - 300. As the vector averaging model is highly dependent on the number of dimensions that a vector can have, the curse of dimensionality might be the reason for the poor performance of contextualised word embeddings.

4.2 Word Mover's Distance

The results for the Word Mover's Distance is shown in 5. Following variations were considered and reported in each sub-table.

1. Considering all the words to calculate the Word Mover's Distance. Results are shown in 5a

2. Removing stop words before calculating the Word Mover's Distance. Table 5b shows the results.

As depicted in table 5a contextualised word representations could not improve Word Mover's method too over standard word representations. Since the travelling distance is dependent on number of dimensions, the curse of dimensionality might be the reason for the poor performance of contextualised word representations in this scenario too.

Embedding	τ	ρ	MSE
Word2vec	**0.732**	**0.624**	**1.664**
ELMo	0.655	0.592	1.863
Flair	0.632	0.559	3.348
BERT	0.584	0.591	3.258
ELMo + BERT	0.654	0.612	2.789

(a) Averaging all the word vectors

Embedding	τ	ρ	MSE
Word2vec	**0.720**	**0.585**	**1.440**
ELMo	0.676	0.597	1.729
Flair	0.668	0.561	2.235
BERT	0.646	0.607	2.958
ELM0 + BERT	0.693	0.620	2.496

(b) Averaging all the word vectors removing stop words

Embedding	MSE	τ	ρ
Word2vec	**0.708**	**0.581**	**1.311**
ELMo	0.675	0.589	1.600
Flair	0.657	0.547	2.074
BERT	0.596	0.575	2.890
ELMo + BERT	0.661	0.594	2.387

(c) Averaging all the word vectors weighting them with tf-idf

Embedding	τ	ρ	MSE
Word2vec	**0.705**	**0.565**	**1.300**
ELMo	0.669	0.582	1.550
Flair	0.661	0.545	1.809
BERT	0.591	0.569	2.739
ELMo + BERT	0.656	0.587	2.250

(d) Averaging all the word vectors weighting them with tf-idf removing stop words

Table 4: Vector averaging results for SICK training set

Embedding	τ	ρ	MSE
Word2vec	**0.642**	**0.593**	**1.051**
ELMo	0.584	0.559	1.210
Flair	0.592	0.561	1.166
BERT	0.605	0.578	1.145
ELMo + BERT	0.595	0.568	1.189

(a) Considering all the word vectors

Embedding	τ	ρ	MSE
Word2vec	**0.636**	**0.573**	**1.156**
ELMo	0.600	0.549	1.416
Flair	0.615	0.557	1.254
BERT	0.639	0.580	1.177
ELMo + BERT	0.619	0.565	1.299

(b) Considering all the word vectors removing stop words

Table 5: Word moving distance results for SICK training set

4.3 Cosine Similarity Using Smooth Inverse Frequency

Table 6 shows the results for the Smooth Inverse Frequency method. As shown there, all the contextualised word representations have improved the results significantly over the standard word representations. Since the first principle component is removed in the process, curse of dimensionality has not affected this method. The stacked embeddings of ELMo and BERT provided the best results to the experiment. Also, it is important to notice that the Smooth Inverse Frequency using the stacked embeddings of ELMo and BERT showed the best results from all three methods for all three evaluation metrics.

As shown in the above tables, contextualised word embeddings did not improve the results of vector averaging and word movers distance. But contextualised word embeddings showed a great improvement over standard word embeddings in Smooth Inverse Frequency STS method which also provided the best results among the considered unsupervised STS methods.

Embedding	τ	ρ	MSE
Word2vec	0.734	0.632	0.604
ELMo	0.740	0.654	0.593
Flair	0.731	0.634	0.601
BERT	0.746	0.661	0.456
ELMo + BERT	**0.753**	**0.669**	**0.446**

Table 6: Smooth Inverse Frequency results for SICK training set

4.4 Further Experiments and Results

As shown in the above section Smooth Inverse Frequency with ELMo and BERT stacked contextualised word representations provided the best result. However, since we used the cosine similarity between two vectors, the predictions of our model are constrained to follow the cosine curve and are thus not suited for these evaluation metrics. For this reason, we applied a parametric regression step to obtain better-calibrated predictions. We trained a regression model on the SICK train data and predicted on the SICK test data. This calibration step served as a minor correction for our restrictively simple similarity function. However, this regression calibration improved the Pearson correlation by 0.01 for the SICK test set.

Our unsupervised method had 0.762 Pearson correlation score, whilst the best result in the International Workshop on Semantic Evaluation 2014 Task 1 had 0.828 Pearson correlation (Marelli et al., 2014). Our approach would be ranked on the ninth position from the top results out of 18 participants, and it is the best unsupervised STS method among the results. Our method even outperformed systems that rely on additional feature generation (e.g. dependency parses) or data augmentation schemes. As an example, our method is just above the UoW system which relied on 20 linguistics features fed in to a Support Vector Machine and obtained a 0.714 Pearson correlation (Gupta et al., 2014). Compared to these complex approaches our simple approach provides a strong baseline to STS tasks.

5 Portability of the Method to Other Languages and Domains

Our approach has the advantage that it does not rely on language dependent features and it does not need a training set as the approach is unsupervised. As a result, the approach is easily portable to other languages and domains given the availability of ELMo and BERT models in that particular language or domain. In order to observe how well the method performs in other languages and domains we applied it to Spanish STS dataset and Biomedical STS dataset described in section 3.

5.1 Spanish STS

We run all the unsupervised STS methods described in section 2 on the Spanish STS dataset explained in section 2.1.2. For the ELMo embeddings we used Spanish ELMo embeddings provided in Che et al. (2018), while for the BERT embeddings we used "BERT-Base, Multilingual Cased" [1] model which has been built on the top 100 languages with the largest Wikipedias which includes Spanish language too.

The predictions from the experiment were rescaled to lie $\in$ [0,4] as the GOLD standards. Organisers have used only one evaluation metric in this Spanish STS task: Pearson correlation (τ) against the predictions and GOLD standard. They have calculated Pearson correlation for each test set: Spanish news and Spanish wiki, separately and has taken the weighted average to give the final rankings in the leader board. We took the same procedure in order to evaluate our approach with the other approaches in the task. Also we applied parametric regression step we did to English-English STS experiment to obtain better-calibrated predictions. Parametric regression step improved the Pearson correlation by 0.01 for both Wikipedia and Newswire datasets.

From the experiments, Smooth Inverse Frequency with ELMo and BERT stacked embeddings gave the best results, similar to the English STS experiments we conducted. Our approach had 0.660 Pearson correlation for Wikipedia dataset, 0.547 Pearson correlation for Newswire dataset and 0.570 weighted mean from both of them. The best performing model that participated in SemEval 2015 task 2, had 0.705 Pearson correlation for Wikipedia, 0.683 for Newswire and 0.690 weighted mean (Agirre et al., 2015). Our approach would rank fifth out of 17 team in the final results, which is the best result for an unsupervised approach. As with the English model, this one also surpasses other complex supervised models. As an example RTM-DCU-1stST.tree uses a supervised machine learning algorithm with Referential Translation Machines(Biici and Way, 2014) and our fairly simple unsupervised approach outperform them by a significant margin. Comparing the results we can safely assume that our approach works well with Spanish language STS too.

5.2 Bio-Medical STS

In order to evaluate our approach in a different domain, we experimented it on Bio-medical STS dataset explained in 2.1.3. As in the previous experiments we applied all unsupervised approaches mentioned. We used ELMo embeddings trained on a biomedical domain corpora (e.g., PubMed abstracts, PMC full-text articles) (Peters et al.,

[1] https://github.com/google-research/bert

2018) and BioBERT: BERT embeddings trained on biomedical domain corpora (Lee et al., 2019). We did not apply Parametric regression step to this dataset since there was not enough data for the training. The predictions from the experiment were re-scaled to lie $\in$ [0,4] as the GOLD standards. Organisers have used only one evaluation metric in this Bio-medical STS task: Pearson correlation (τ) against the predictions and GOLD standard.

Same as English and Spanish experiments, Smooth Inverse Frequency with ELMo and BERT stacked embeddings performed best with this dataset too. It had 0.680 Pearson correlation, whilst the best performing method had 0.836 Pearson correlation. This would rank our approach seventh out of 22 teams in the final results of the task (Sogancioglu et al., 2017). It should be also noted that it outperforms many complex methods that sometimes uses external tools too. As an example, the UBSM-Path approach is based ontology based similarity which uses METAMAP (Aronson, 2001) for extracting medical concepts from text and our simple unsupervised approach outperform them by a significant margin. UBSM-Path only has 0.651 Pearson correlation. Comparing the results we can safely assume that our approach works well in bio medical domain too.

6 Related Work

Given that a good STS metric is required for a variety of natural language processing fields, researchers have proposed a large number of such metrics. Before the shift of interest in neural networks, most of the proposed methods relied heavily on feature engineering. With the introduction of word embedding models, researchers focused more on neural representation for this task.

There are two main approaches which employ neural representation models: supervised and unsupervised. Unsupervised approaches use pre-trained word/sentence embeddings directly for the similarity task without training a neural network model on them while supervised approaches uses a machine learning model trained to predict the similarity using word embeddings. ConvNet (He et al., 2015), Skip Thought vectors (Kiros et al., 2015), Dependency Tree-LSTM (Tai et al., 2015) and Siamese Neural Networks (Mueller and Thyagarajan, 2016) can be considered as the most successful architectures employed for calculating STS. These supervised approaches always suffer from less training data problem which is common in STS tasks. As a result the researches have also considered unsupervised approaches.

The three unsupervised STS methods explored in this paper: Cosine similarity on average vectors, Word Mover's Distance and Cosine similarity using Smooth Inverse Frequency are the most common unsupervised methods explored in STS tasks. Apart from them cosine similarity of the output from Infersent (Conneau et al., 2017), sent2vec (Pagliardini et al., 2018) and doc2vec (Le and Mikolov, 2014) have been used to represent the similarity between two sentences. All these approaches relies on pre-trained sentence embeddings.

7 Conclusions

This paper experimented three unsupervised STS methods namely cosine similarity using average vectors, Word Mover's Distance and cosine similarity using Smooth Inverse Frequency with contextualised word embeddings for calculating semantic similarity between pairs of texts and compared them with other unsupervised/ supervised approaches. Contextualised word embeddings could not improve cosine similarity using average vectors and Word Mover's Distance methods, but the results when using Smooth Inverse Frequency method were improved significantly with contextualised word embeddings, instead of standard word embeddings. Further more we learned that stacking ELMo and BERT provides a strong word representation rather than individual representations of ELMo and BERT. The results indicated that calculating cosine similarity using Smooth Inverse Frequency with stacked embeddings of ELMo and BERT is the best unsupervised method from the available approaches. Also, our approach finished on the top half of the final results list surpassing many complex and supervised approaches.

Our approach was also applied in the Spanish STS and Bio-medical STS tasks, where our simple unsupervised approach finished on the top half of the final result list in both cases. Therefore, given our results we can safely assume that regardless of the language or the domain cosine similarity using Smooth Inverse Frequency with stacked embeddings of ELMo and BERT will provide a simple but strong unsupervised method for STS tasks.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@NAACL-HLT*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@COLING*.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. page to appear.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*. pages 1638–1649.

Ramiz M. Aliguliyev. 2009. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Syst. Appl.* 36:7764–7772.

Alan R. Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. *Proceedings. AMIA Symposium* pages 17–21.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings .

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR 2017*.

Hanna Béchara, Hernani Costa, Shiva Taslimipoor, Rohit Gupta, Constantin Orasan, Gloria Corpas Pastor, and Ruslan Mitkov. 2015. Miniexperts: An svm approach for measuring semantic textual similarity. In *SemEval@NAACL-HLT*.

Luisa Bentivogli, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2016. Sick through the semeval glasses. lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Language Resources and Evaluation* 50:95–124.

Ergun Biici and Andy Way. 2014. Rtm-dcu: Referential translation machines for semantic similarity. In *SemEval@COLING*.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium, pages 55–64. http://www.aclweb.org/anthology/K18-2005.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

João D. Ferreira and Francisco M. Couto. 2010. Semantic similarity for automatic classification of chemical compounds. In *PLoS Computational Biology*.

Rohit Gupta, Hanna Béchara, Ismaïl El Maarouf, and Constantin Orasan. 2014. Uow: Nlp techniques developed at the university of wolverhampton for semantic similarity and textual entailment. In *SemEval@COLING*.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *ICML*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR* abs/1901.08746.

Yuhua Li, David McLean, Zuhair Bandar, James O'Shea, and Keeley A. Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering* 18:1138–1150.

Goutam Majumder, Partha Pakray, Alexander F. Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. *Computación y Sistemas* 20.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@COLING*.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Michael Mohler, Razvan C. Bunescu, and Rada Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *ACL*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners .

J. J. Rocchio. 1971. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, Englewood Cliffs, NJ: Prentice-Hall, pages 313–323.

Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic text structuring and summarization. *Inf. Process. Manage.* 33(2):193–207. https://doi.org/10.1016/S0306-4573(96)00062-3.

Yang Shao. 2017. Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *SemEval@ACL*.

Gizem Sogancioglu, Hakime Öztürk, and Arzucan Özgür. 2017. Biosses: a semantic sentence similarity estimation system for the biomedical domain. In *Bioinformatics*.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Wei Xu, Chris Callison-Burch, and William B. Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *SemEval@NAACL-HLT*.

Semantic Textual Similarity with Siamese Neural Networks

Tharindu Ranasinghe, Constantin Orăsan and Ruslan Mitkov
Research Group in Computational Linguistics
University of Wolverhampton, UK
{t.d.ranasinghehettiarachchige, c.orasan, r.mitkov }@wlv.ac.uk

Abstract

Calculating the Semantic Textual Similarity (STS) is an important research area in natural language processing which plays a significant role in many applications such as question answering, document summarisation, information retrieval and information extraction. This paper evaluates Siamese recurrent architectures, a special type of neural networks, which are used here to measure STS. Several variants of the architecture are compared with existing methods.

1 Introduction

Measuring Semantic Textual Similarity (STS) is the task of calculating the similarity between a pair of texts using both direct and indirect relationships between them (Rus et al., 2013). Originally, the work on STS largely focused on similarity between short texts such as abstracts and product descriptions (Li et al., 2006; Mihalcea et al., 2006). The introduction of the STS tasks at the International Workshops on Semantic Evaluation (SemEval) lead to an increase of the interest that the field received from the research community. The SemEval tasks also led to the development of standard datasets like the SICK corpus (Bentivogli et al., 2016) and standardised the similarity score as a numerical value between 1 and 5 (Agirre et al., 2014).

Having a good STS metric is very important in many natural language processing (NLP) applications. As an example, for certain types of question answering systems, having an accurate STS component is the key to success since the questions with similar meanings can be answered similarly (Majumder et al., 2016). STS is also important in translation memories retrieval and matching (Gupta et al., 2014b). Translation memories help translators by finding in the database they maintain previously translated sentences, which are similar to the one to be translated, and retrieving their translations. Hence, accurate STS methods are beneficial for translation memory.

Given the growing importance of having a good STS metric and as a result of the SemEval workshops, researchers have proposed numerous STS methods. Most of the early approaches were based on traditional machine learning and involved heavy feature engineering (Béchara et al., 2015). With the advances of word embeddings, and as a result of the success neural networks have achieved in other fields, most of the methods proposed in recent years rely on neural architectures (Tai et al., 2015; Shao, 2017). Neural networks are preferred over traditional machine learning models as they generally tend to perform better than traditional machine learning models. They also do not rely on linguistic features which means they can be easily applied to languages other than English. The architecture employed in this paper is a special class of neural networks called *Siamese neural networks*. These networks contain two or more identical sub-networks. The networks are identical in the sense that they have the same configuration with the same parameters and weights. In addition, parameter updating is mirrored across these sub-networks.

Siamese networks are popular among tasks that involve finding similarity or a relationship between two comparable things. They have been proven successful in tasks like signature verification (Bromley et al., 1993), face verification (Chopra et al., 2005), image similarity (Koch et al., 2015) and have been re-

Proceedings of Recent Advances in Natural Language Processing, pages 1004–1011,
Varna, Bulgaria, Sep 2–4 2019.

cently used successfully in sentence similarity (Neculoiu et al., 2016). Siamese architectures are good in these tasks because, when the inputs are of the same kind, it makes sense to use a similar model to process similar inputs. In this way the networks will have representation vectors with the same semantics, making them easier to compare pairs of sentences. Given that the weights are shared across sub networks there are fewer parameters to train, which in turn means they require less training data and less tendency to over-fit. Given the amount of human labour required to produce datasets for STS, Siamese neural networks can prove the ideal solution for the STS task.

This paper explores the performance of several architectures which use Siamese neural networks for STS. The rest of the paper is organized as follows. Section 2 briefly describes several approaches used to measure sentence similarity focusing more on Siamese neural networks. Section 3 contains information about the settings of the experiments carried out in this paper including the datasets employed here and the different architectures explored. The architectures are evaluated in Section 4. The paper finishes with conclusions.

2 Related Work

Given that a good STS metric is required for a variety of NLP fields, researchers have proposed a large number of such metrics. Before the shift of interest in neural networks, most of the proposed methods relied heavily on feature engineering. A typical example is (Gupta et al., 2014a) which employed 20 linguistic features fed into a support vector machine regressor. The top system (Zhao et al., 2014) in task 1 in SemEval 2014 has used seven types of features including text difference measures, common text similarity measures etc. (Zhao et al., 2014). Then they have fed it in to several learning algorithms like support vector machine regressor, Random Forest, Gradient boosting etc (Zhao et al., 2014). With the introduction of word embedding models, researchers focused more on neural representation for this task. Many of the leading teams in the STS task at Semeval 2017 used some kind of neural network architecture which employed word embeddings (Shao, 2017). As an example, Maharjan et al. (2017) used an ensemble of traditional machine learning models and deep learning models in their top performing system at Semeval 2017 STS task.

There are two main approaches which employ neural representation models: supervised and unsupervised. Unsupervised approaches use pretrained word/sentence embeddings directly for the similarity task without training a neural network model on them. Such approaches have used cosine similarity on sent2vec (Pagliardini et al., 2018), InferSent (Conneau et al., 2017), Doc2Vec (Le and Mikolov, 2014) and smooth inverse frequency with GloVe vectors (Arora et al., 2017).

Supervised approaches use neural networks to project word embeddings to fixed dimensional vectors which are trained to capture the semantic meaning of the sentence. Recently, many neural network architectures have been used to calculate sentence similarity. He et al. (2015) propose an elaborate convolutional network (ConvNet) variant which infers sentence similarity by integrating various differences across many convolutions at varying scales.

Kiros et al. (2015) propose the skip-thoughts model, which extends the skip-gram approach of word2vec from the word to sentence level. This model feeds each sentence into an Recurrent Neural Network (RNN) encoder-decoder with Gated Recurrent Unit (GRU) activations. They attempt to reconstruct the immediately preceding and following sentences. For the sentence similarity task, they obtain skip-thought vectors for sentence pairs. Then a separate classifier is trained using features derived from differences and products between skip-thought vectors for each pair of sentences.

Tai et al. (2015) propose Tree-LSTMs (Long short-term memory) which generalize the order-sensitive chain-structure of standard LSTMs to tree-structured network topologies. Each sentence is first converted into a parse tree using a separately trained parser, and the Tree-LSTM composes its hidden state at a given tree node from the corresponding word as well as the hidden states of all child nodes. The hope is that by reflecting syntactic properties of a sentence, the parse tree structured network can propagate necessary information more efficiently than a sequentially restricted architecture. The output from Tree-LSTM can be used for sentence similarity task as the same way as Kiros et al. (2015), where representations of the input sentences are now produced by Tree-LSTMs rather than skip-thoughts.

Our proposed model also represents sentences using neural networks whose inputs are word vectors learned separately from a large corpus. But unlike the models proposed by Kiros et al. (2015) and Tai et al. (2015) the sole target of our objective function is to calculate sentence similarity. In order to have an objective function that solely focus on similarity we need an architecture which is capable of handling two sentences parallelly. To do that we use a special kind of neural network architecture: Siamese neural network architecture.

Siamese recurrent neural networks have been recently used in STS tasks. The MALSTM architecture (Mueller and Thyagarajan, 2016) uses two identical LSTM networks trying to project zero padded word embeddings of a sentence to fixed sized 50 dimensional vectors using Manhattan distance as the similarity function between 2 sub networks. Mueller and Thyagarajan (2016) report that it performs better than other neural network models like Tree-LSTM (Tai et al., 2015). This inspired us to use this model and extend it. This research proposes new variants of the MALSTM architecture for predicting STS [1].

3 Settings of the Experiments

3.1 Data Sets

The experiments presented in this paper were carried out using the SICK dataset (Bentivogli et al., 2016) and SemEval 2017 Task 1 dataset (Cer et al., 2017) which we will refer as STS2017 dataset.

The SICK data contains 9927 sentence pairs with a 5,000/4,927 training/test split which were employed in the SemEval tasks. Each pair is annotated with a relatedness score between [1,5] corresponding to the average relatedness judged by 10 different individuals. In order to generate more training data we used thesaurus-based augmentation (Miller, 1992) and added 10,022 additional training examples. Evaluation was done with the SICK test data. Mueller and Thyagarajan (2016) uses the same thesaurus-based data augmentation in their research.

The STS2017 test datset had 250 sentence pairs annotated with a relatedness score between [1,5]. As the training data for the competition, participants were encouraged to make use of all existing data sets from prior STS evaluations including all previously released trial, training and evaluation data [2]. Once we combined all datasets from prior STS tasks we had 8277 sentence pairs for training.

3.2 Proposed Architectures

The basic structure of the Siamese neural network architecture used in our experiments is shown in Figure 1. It consists of an embedding layer which represents each sentence as a sequence of word vectors. This sequence of word vectors is fed into a Recurrent Neural Network (RNN) cell which learns a mapping from the space of variable length sequences of 300-dimensional vectors into a 50 dimensional vector. The sole error signal backpropagated during training, stems from the similarity between these 50 dimensional vectors, which can be also used as a sentence representation. Initially, the similarity function we used was based on Manhattan distance. To make sure that the prediction is between 0 and 1, we took the exponent of the negative Manhattan distance between 2 sentence representations. The similarity function was adopted from Mueller and Thyagarajan (2016). The proposed variants of our architecture are:

1. LSTM - Block A in Figure 1 contains a single LSTM cell. This is the architecture suggested by Mueller and Thyagarajan (2016)

2. Bi-directional LSTM - Block A in Figure 1 contains a single Bi-directional LSTM cell. Bi-directional LSTM tends to understand the context better than Unidirectional LSTM (Schuster and Paliwal, 1997).

3. GRU - Block A in Figure 1 contains a single GRU cell. GRUs have been shown

[1] The code is available on "https://github.com/TharinduDR/Siamese-Recurrent-Architectures"

[2] http://alt.qcri.org/semeval2017/task1/

Approach	τ	ρ	MSE
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jiménez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	NA	NA
Skip-thought+COCO (Kiros et al., 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai et al., 2015)	0.8676	0.8083	0.2532
ConvNet (He et al., 2015)	0.8686	0.8047	0.2606
Bi-directional LSTM†	0.8743	0.8251	0.2391
GRU + Capsule + Flatten†	0.8786	0.8286	0.2301
MALSTM (Baseline) (Mueller and Thyagarajan, 2016)	0.8822	0.8345	0.2286
LSTM: Adagrad†	0.8831	0.8364	0.2195
GRU + Attention†	0.8843	0.8372	0.2163
LSTM + Attention†	0.8886	0.8386	0.2142
Bi-directional GRU†	0.8896	0.8390	0.2125
GRU†	0.8901	0.8396	0.2112

Table 1: Pearson correlation (τ), Spearman correlation (ρ), and Mean Square Error (MSE) for the SICK test set.

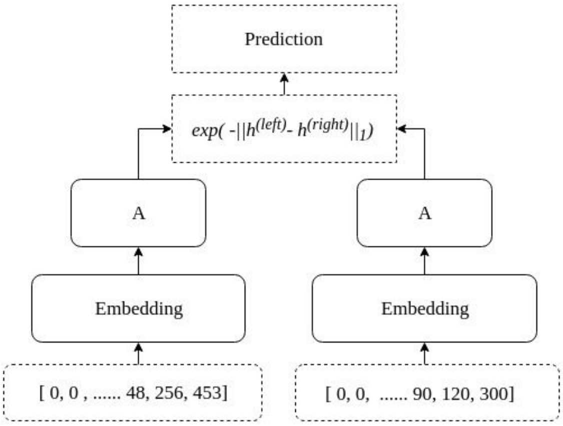


Figure 1: Basic structure of the Siamese neural network. Unit A is changed over the architectures.

to exhibit better performance on smaller datasets (Chung et al., 2014).

4. Bi-directional GRU - Block A in Figure 1 contains a single Bi-directional GRU cell. Bi-directional GRUs tend to understand the context better than Unidirectional GRUs (Vukotic et al., 2016).

5. LSTM + Attention - Block A in Figure 1 contains a single LSTM cell with self attention (Bahdanau et al., 2014).

6. GRU + Attention - Block A in Figure 1 contains a single GRU cell with self attention (Bahdanau et al., 2014).

7. GRU + Capsule + Flatten - Block A in Figure 1 contains a GRU followed by a capsule layer and a flatten layer. Dynamic routing used between capsules performs better than a traditional max-pooling layer (Sabour et al., 2017).

4 Evaluation Results

Table 1 shows the results obtained for the proposed architectures on the SICK test set. The table only reports the best results for each architecture [3]. The first group of results are top SemEval 2014 submissions and the second group are recent neural network methods (best result from each paper shown). † denotes the experiments we conducted in this research. All the models were evaluated using the three evaluation metrics normally employed in the STS tasks: Mean Square Error (MSE), Pearson correlation (τ) and Spearman correlation (ρ).

MALSTM (Mueller and Thyagarajan, 2016) is the baseline that we defined for this research. The baseline is the best result achieved by the architecture reported in Mueller and Thyagarajan (2016). Interestingly, we were able to beat the baseline using the same architecture using the Adagrad optimiser (Duchi et al., 2011) (LSTM:Adagrad).

The best result was obtained when block A in figure 1 contains a single GRU. As can be seen in the table 1, the proposed architecture outperformed both the benchmark and all the other architectures in all 3 evaluation metrics.

As can be seen in Table 1, the architectures with bidirectional GRU, LSTM with Attention and GRU with Attention also surpassed the benchmark. However uni-directional GRU out performed them on all 3 evaluation metrics.

We experimented with other similarity functions and other embedding models. Using Euclidean distance for the similarity function instead of Manhattan distance did not improve the results for the model because semantically different sentences could end up being represented by nearly identical vectors due to the vanishing gradients of the Euclidean distance (Chopra et al., 2005). Changing the embedding model to GloVe (Pennington et al., 2014), fastText (Mikolov et al., 2018) or concatenating them with word2vec model did not improve the results either. For this reason, none of these results are presented here.

The Siamese neural network with GRU was tested with a cyclical learning rate (Smith, 2015), which has the advantage of forcing the model to find another local minimum if the current minimum is not robust and makes the model generalize better to unseen data. However, neither cyclical learning rate nor reducing learning rate on plateau increased the performance further. We do not report these results too.

Approach	τ
FCICU (Hassan et al., 2017)	0.8280
BIT (Wu et al., 2017)	0.8400
ECNU (Tian et al., 2017)	0.8518
DT Team (Maharjan et al., 2017)	0.8536
Bi-directional LSTM†	0.8540
GRU + Capsule + Flatten†	0.8545
RTV	0.8547
MALSTM	0.8651
LSTM: Adagrad†	0.8692
GRU + Attention†	0.8725
LSTM + Attention†	0.8743
Bi-directional GRU†	0.8750
GRU†	0.8792

Table 2: Pearson correlation (τ) for STS2017 test set.

Table 2 shows the results obtained for STS2017 test dataset comparing our experiments with other top performing models in SemEval 2017 Task 1 (Cer et al., 2017). † denotes the experiments we conducted in this research. As SemEval 2017 Task 1 used Pearson correlation (τ) to evaluate the submissions, we evaluated our models using Pearson correlation (τ) too.

GRU based Siamese neural network performs better than existing systems for STS2017 dataset too, as it is shown in table 2.

4.1 Error Analysis

In order to understand better why the GRU based architecture performed better than the LSTM baseline, we compared sentences where the GRU architecture was better. Table 3 shows examples of such sentences from the SICK testset. Our analysis suggests that the GRU based architecture handles the additional words better than LSTM. Mueller and Thyagarajan (2016) report that their architecture does not perform well with active-passive equivalence. However, as shown in Table 4,

[3] These results are reported in Marelli et al. (2014)

Sentence 1	Sentence 2	GOLD	LSTM	GRU
The people are walking on the road beside a beautiful waterfall	The people are walking on the road beside a waterfall, which is beautiful	0.9750	0.5260	0.9569
The woman is frying a chop of breaded pork	The woman is frying a breaded pork chop	0.9250	0.5278	0.8561
A white dog is standing on the leaves on the ground	A dog, which is white, is standing on fallen leaves	0.9500	0.3611	0.7618
The man is erasing the other man's work from the board	The man is erasing the drawing on the board	0.7500	0.5128	0.7584

Table 3: Example sentence pairs from the SICK test data. LSTM denotes the baseline and GRU the best model

Sentence 1	Sentence 2	GOLD	LSTM	GRU
A man is mixing vegetables in a pot	Vegetables are being mixed in a pot by a man	0.9750	0.6684	0.8154
Carrots are being sliced by a woman	A woman is slicing carrots	1.0000	0.6739	0.7206
The elephant is being ridden by the woman	The woman is riding the elephant	0.9500	0.2249	0.5939

Table 4: Example active-passive sentence pairs from the SICK test data.

our architecture performs slightly better than the LSTM based architecture.

5 Conclusions

This paper evaluated several neural architectures based on Siamese recurrent neural network for calculating semantic similarity between pairs of texts. Most of these architectures fared better than the approach proposed in (Mueller and Thyagarajan, 2016). The variant with a GRU performed best, capitalising on GRU's ability to exhibit better performance on smaller datasets like the ones available for STS. Our architectures can be easily ported to other languages which have training data available, and we are currently experimenting with other languages.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@COLING*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Hanna Béchara, Hernani Costa, Shiva Taslimipoor, Rohit Gupta, Constantin Orasan, Gloria Corpas Pastor, and Ruslan Mitkov. 2015. Miniexperts: An svm approach for measuring semantic textual similarity. In *SemEval@NAACL-HLT*.

Luisa Bentivogli, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2016. Sick through the semeval glasses. lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Language Resources and Evaluation*, 50:95–124.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *SemEval@COLING*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *IJPRAI*, 7:669–688.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017.

Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:539–546 vol. 1.

Junyoung Chung, ÃĞaglar GülÃğehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Rohit Gupta, Hanna Béchara, Ismaïl El Maarouf, and Constantin Orasan. 2014a. Uow: Nlp techniques developed at the university of wolverhampton for semantic similarity and textual entailment. In *SemEval@COLING*.

Rohit Gupta, Hanna Bechara, and Constantin Orasan. 2014b. Intelligent translation memory matching and retrieval metric exploiting linguistic technology. *Proc. of Translating and the Computer*, 36:86–89.

Basma Hassan, Samir E. AbdelRahman, Reem Bahgat, and Ibrahim Farag. 2017. Fcicu at semeval-2017 task 1: Sense-based language independent semantic textual similarity approach. In *SemEval@ACL*.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*.

Sergio Jiménez, George Dueñas, Julia Baquero, and Alexander F. Gelbukh. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *SemEval@COLING*.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *SemEval@COLING*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Yuhua Li, David McLean, Zuhair Bandar, James O'Shea, and Keeley A. Crockett. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18:1138–1150.

Nabin Maharjan, Rajendra Banjade, Dipesh Gautam, Lasang Jimba Tamang, and Vasile Rus. 2017. Dt team at semeval-2017 task 1: Semantic similarity using alignments, sentence-level embeddings and gaussian mixture model output. In *SemEval@ACL*.

Goutam Majumder, Partha Pakray, Alexander F. Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. *Computación y Sistemas*, 20.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval@COLING*.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *ACL 2016*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Vasile Rus, Mihai C. Lintean, Rajendra Banjade, Nobal B. Niraula, and Dan Stefanescu. 2013. Semilar: The semantic similarity toolkit. In *ACL*.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *NIPS 2017*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Yang Shao. 2017. Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *SemEval@ACL*.

Leslie N. Smith. 2015. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *SemEval@ACL*.

Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2016. A step beyond local observations with a dialog aware bidirectional gru network for spoken language understanding. In *INTERSPEECH*.

Hao Wu, Heyan Huang, Ping Jian, Yuhang Guo, and Chao Su. 2017. Bit at semeval-2017 task 1: Using semantic information space to evaluate semantic textual similarity. In *SemEval@ACL*.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *SemEval@COLING*.

Analysing the Impact of Supervised Machine Learning on Automatic Term Extraction: HAMLET vs TermoStat

Ayla Rigouts Terryn[*], Patrick Drouin[**], Veronique Hoste[*] and Els Lefever[*]

[*]LT^3 Language and Translation Technology Team, Ghent University
Groot-Brittanniëlaan 45, 9000 Gent; name.surname@ugent.be
[**]Observatoire de Linguistique Sens-Texte, Université de Montréal
Succ. Centre-ville, Montréal, QC, H3C 3J7; patrick.drouin@umontreal.ca

Abstract

Traditional approaches to automatic term extraction do not rely on machine learning (ML) and select the top n ranked candidate terms or candidate terms above a certain predefined cut-off point, based on a limited number of linguistic and statistical clues. However, supervised ML approaches are gaining interest. Relatively little is known about the impact of these supervised methodologies; evaluations are often limited to precision, and sometimes recall and f1-scores, without information about the nature of the extracted candidate terms. Therefore, the current paper presents a detailed and elaborate analysis and comparison of a traditional, state-of-the-art system (TermoStat) and a new, supervised ML approach (HAMLET), using the results obtained for the same, manually annotated, Dutch corpus about dressage.

1 Introduction

Automatic term extraction (ATE), also known as automatic term recognition (ATR), has long been an established task within the field of natural language processing. It can be used both in its own right, to automatically obtain a list of candidate terms (cts) from a specialised corpus, or as a preprocessing step for other tasks, such as machine translation (Wolf et al., 2011). The traditional method for ATE is a hybrid approach, combining both linguistic and statistical information. In a first step, linguistic preprocessing is performed and a preliminary list of cts is produced based on part-of-speech (POS) patterns. Next, statistical metrics are applied to measure termhood (to what degree a term is related to the domain) and unithood for multi-word terms (whether the individual tokens combine to form a lexical unit) (Kageura and Umino, 1996). These metrics are used to sort the cts based on their likelihood to be actual terms. To filter the list, one can either determine a cut-off value or select the top n or top n percent of terms. As a final step, manual validation is required.

This has been a standard methodology for some time (Daille, 1994) and is still used by state-of-the-art systems such as TermoStat (Drouin, 2003) and TExSIS (Macken et al., 2013). However, the problem with these methodologies is determining the cut-off point (Lopes and Vieira, 2015) and combining multiple features (e.g., separate measures for termhood and unithood). It has become clear that multiple evidence (i.e. combining multiple features) is highly beneficial for ATE (Dobrov and Loukachevitch, 2011; Loukachevitch, 2012). Supervised machine learning (ML) methodologies are now being used in answer to these problems. By automatically learning an optimal combination of features and cut-off points, many features can be efficiently combined.

One of the biggest hurdles for the progress of ATE technologies has been the data acquisition bottleneck, both for evaluation and now also as training data. Manually annotating terms is a slow and arduous task, with notoriously low inter-annotator agreement due to the ambiguous nature of terms. This lack of agreement on the basic characteristics of terms is also reflected in the different methodologies of various ATE research, e.g., min./max. length and frequency, POS patterns and degree of specialisation. As a result, the supervised methodologies that have been developed are extremely difficult to compare (both to each other and to non-ML systems) and qualitative analyses that go beyond calculating precision (how many of the extracted cts are true terms), recall (how many of the true terms are extracted) and f1-scores (weighted average of precision and recall) are rare.

The construction of a diverse and extensive dataset for ATE (Rigouts Terryn et al., 2019) pro-

Proceedings of Recent Advances in Natural Language Processing, pages 1012–1021,
Varna, Bulgaria, Sep 2–4 2019.

vided an opportunity to (1) develop a supervised ML approach for ATE (HAMLET) and (2) perform a detailed evaluation of this system compared to a traditional tool without ML: TermoStat (Drouin, 2003). These specific systems were chosen because they both allow extraction of single- and multi-word terms (swts and mwts) and are not restricted to only nouns and noun phrases, but instead also allow verbs, adjectives and adverbs to be extracted. Moreover, their methodology is similar, so the research can focus on one main difference: the fact that HAMLET uses supervised ML to combine different features, rather than relying on manually set filters and thresholds like TermoStat. This is important to better understand the impact of the methodology. Are the same terms found with both methodologies? Do they make similar mistakes? Is it possible to see the impact of the training data? The analysis is performed by a terminologist, in her native language (Dutch) and on a subject for which she is a domain specialist (equitation - dressage).

2 Related Research

Some of the original supervised approaches to ATE start appearing in the early 2000s. Vivaldi and Rodríguez (2001) claim to be the first to combine different methodologies for term extraction into a single system. Based on two manually annotated Spanish corpora in the medical domain, four different strategies are combined. The first strategy is to use EuroWordNet (EWN) (Vossen, 1998) to determine whether a word belongs to the medical domain. Next, Greek and Latin word forms are detected. Context is analysed as well, focusing on prime term candidates, i.e. those that are validated with EWN as medical terms. Finally, three unithood measures help to find relevant multi-word terms. Combining these four techniques leads to better results than using any one of them separately. The system is only tested on the Spanish medical domain; performance may vary significantly depending on EWN coverage of the corpus and relevance of the Latin and Greek words. Later research does test on multiple domains, for instance, an evolutionary algorithm based on the optimisation of the Receiver Operating Characteristics curve for the extraction of mwts (Azé et al., 2005), tested on the domains of biology and HR; or a system for both swts and mwts (Yuan et al., 2017), elaborately evaluated with different algorithms, using undersampling to obtain more balanced data, and cross-domain training/testing on four domains.

In 2016, neural network word embeddings are applied to ATE for the first time (Amjadian et al., 2016), first as a filter on an existing tool (TermoStat), later on also as a full ATE pipeline (Amjadian et al., 2018). The success of multiple features for ATE has been proven repeatedly (Dobrov and Loukachevitch, 2011; Loukachevitch, 2012; Nokel, Michael et al., 2012) and aside from the original binary classification approach of cts, sequence labelling approaches are also gaining interest (Judea et al., 2014; Kucza et al., 2018). Additionally, There has been an increased interest in more nuanced term labels (Ljubei et al., 2019; Hätty and Schulte im Walde, 2018), even though binary classification is still the norm.

Unsupervised and semi-supervised approaches are starting to appear as well, which is interesting considering the time and effort associated with constructing good gold standard data. Judea, Schütze and Brügmann (2014) use the specific layout of patents to generate training data. Cts are extracted based on their POS pattern and filtered with an elaborate stopword list. When these cts were preceded by a figure reference in patents, 95% of them were true terms. Since these terms could be identified with high precision, they were used as training data to detect other terms without figure references. Another strategy is fault-tolerant learning, which has been used for Chinese ATE (Yang et al., 2011). Two sets of seed terms are extracted from the same, unlabelled dataset, with two different termhood metrics methods. By comparing the results of the two classifiers and retraining on only the best results (for n iterations), a system can be trained without any labelled training data. Human annotation is only used for evaluation, where an approximation of precision is calculated by randomly sampling and annotating 10% of the extracted cts. Patry and Langlais (2005) take an unusual approach regarding the difficulty of obtaining data and ask users to provide an annotated corpus. This added effort on the part of the user would be rewarded in the form of a customised tool, considering the user's own definition of the ambiguous concept of a term. They also cite two of the most common problems for ATE: the lack of a common benchmark for evaluation and the difficulty extracting hapax terms, especially considering that these make up 75% of the terms in their test corpus.

Despite the increasing research interest, research on the impact of ML approaches on ATE is limited (Amjadian et al., 2018; Nokel, Michael et al., 2012). Comparative evaluations are highly problematic for several reasons. First, established benchmarks such as the GENIA corpus (Kim et al., 2003) and the ACL RD-TEC (Qasemizadeh and Schumann, 2016) are rare and often only available in a single language and domain. Second, reported evaluation scores (usually precision, recall and f1-score) differ greatly depending on the strictness of the evaluation (e.g., whether or not partial matches are approved). Third, the difficulty of the task varies considerably depending on the ct selection. For instance, limiting POS patterns and frequency thresholds can result in a more balanced data set and narrower search space. Finally, results are rarely discussed beyond reporting the scores, which may result in a distorted image, given the ambiguous nature of terms, as will be discussed further on. Therefore, while researchers regularly mention the suspected impact of methodology, term definitions, language and domain, little is known about how these factors influence the actual results. The research presented in this paper presents an elaborate and qualitative evaluation and comparison of two tools and will focus on the difference between a supervised ML approach and a traditional approach.

3 Data and Tools

3.1 Data

The dataset is described in detail in (Rigouts Terryn et al., 2019). The Dutch corpus on dressage was chosen as the evaluation corpus. The annotation scheme is based on lexicon-specificity (whether a term belongs to general language or only the vocabulary of experts) and domain-specificity (how relevant the term is to the given domain). Terms are annotated with three different labels: Specific Terms (which are both domain-specific and lexicon-specific), Common Terms (which are domain-specific but not lexicon-specific) and Out-Of-Domain (OOD) Terms (which are not domain specific but are lexicon-specific). Named Entities are annotated as well. In this corpus of around 55k tokens (64 documents), this resulted in 1326 different manual annotations (excluding Split Terms).

3.2 TermoStat

TermoStat is a hybrid term extractor developed by Drouin (2003) which is still continuously updated. It is currently available in French, English, Spanish, Italian, and Portuguese, with beta versions for German, Catalan, Korean, Chinese and Dutch. It is customisable in the sense that users can choose to extract swts, mwts, or both and can also select which POS (nouns, adjectives, adverbs and/or verbs) should be extracted. TermoStat selects cts based on their POS pattern and filters and sorts these cts with the Specificity score, a measure that takes into account the relative frequency of a ct in the specialised corpus, compared to that in a general reference corpus to calculate termhood.

3.3 HAMLET

HAMLET stands for Hybrid Adaptable Machine Learning approach to Extract Terminology and is a supervised methodology for ATE based on the data described in (Rigouts Terryn et al., 2019). HAMLET's architecture is inspired by traditional hybrid systems such as TermoStat. First, cts are extracted based on their POS pattern. However, rather than a predefined list, the patterns are obtained from the annotated training corpus. Since there were no restrictions on which POS could be annotated, this results in an extensive list. Moreover, incorrect patterns due to POS-tagging errors are included as well. This may result in a lot of noise but could also increase recall if similar tagging mistakes are made on terms in the test corpus.

Next, a series of features are calculated for each ct. There are six different feature groups: morphological/shape (e.g., term length, capitalisation, special characters), frequency (e.g., relative frequencies in specialised corpus, newspaper corpus and Wikipedia corpus), statistical (e.g., various termhood and unithood measures), related cts (e.g., information about terms with same lemma or normalised form), linguistic (e.g., POS pattern) and corpus features (e.g., domain of corpus of origin). There are 152 distinct features in total. In contrast to most other term extractors, no restrictions are placed on term length or frequency.

This information is fed to a binary decision tree classifier in Scikit-learn (Pedregosa et al., 2011). Hyperparameter optimisation with grid search is performed in 5 folds on the training data. All values are scaled to a value between 0 and 1. For the experiment discussed in the current contribution, HAMLET was trained on the Dutch corpora about heart failure and wind energy and tested on the Dutch corpus about dressage. Irrelevant features (with the same value for all instances) are discarded, leaving 136 features in this case. The data is highly imbalanced, with fewer than 10% positive instances (similar distribution in train and test sets). While other algorithms were able to reach better scores (e.g., a random forest classifier obtained an f1-score of 61% on the same dataset), only the decision tree model is discussed, because it offers both decent performance and is easy to interpret. Future research will devote more attention to the differences between algorithms for this task.

4 Experiments and Comparisons

4.1 Candidate Terms and Part-of-Speech Patterns

The gold standard data (test data) contains 1326 unique annotations: 985 Specific Terms, 190 Common Terms, 45 OOD Terms and 106 Named Entities. For this experiment, HAMLET was trained to find all annotation types, which is the configuration that lead to the best results for TermoStat. However, HAMLET could also be trained on specific combinations of these labels to customise the results for different applications. Out of the 1326 annotations which were considered true terms, only two could not be found because the annotations were made below token-level and were therefore never selected as a ct by either extractor: *promotie* (promotion, i.e. moving to a higher level of competition) and *k* (one of the letters indicating a certain position in the riding arena). Another portion could not be found due to their POS pattern. This is always a problem for non-ML extractors, since it is nearly impossible to manually define all possible patterns, especially considering that POS taggers can make mistakes. However, the supervised system has similar troubles. HAMLET's preprocessing can only select terms for which the POS pattern occurred in the training corpora (the two Dutch corpora on heart failure and wind energy). In this case, there are 216 different patterns in the training data, but the test corpus still contains terms with 63 patterns that are not in the training data. This illustrates how domain-specific terminology can be. Dressage terminology contains many terms that start with a preposition. For instance, there are 85 annotations of the preposition+determiner+noun pattern, e.g., *aan het been* (responsive to a rider's leg aids). Patterns including verbs are common in dressage as well, e.g., *vierkant halthouden* (stopping the horse so all four hoofs form a rectangle). Due to the absence of such patterns in the training data, 104 terms were not extracted by HAMLET, while 11 of these were found by TermoStat.

Across all 3 languages and 4 domains in the complete dataset, a total of 1345 distinct POS patterns are identified (419 in Dutch in all four domains), meaning that these types of errors are greatly reduced when HAMLET is trained on a larger portion of the data, though that also leads to more noise. This emphasises the importance of diverse datasets to train robust term extractors and to evaluate extractors in multiple domains.

4.2 Decision Tree

The decision tree (of depth 8) that was created based on the training data of Dutch corpora on heart failure and wind energy uses 64 out of the 152 distinct features. All feature categories are represented, except corpus features. In other experiments involving more domains and languages, corpus features are regularly used, but in this setting, with only two different domains in the training data, they did not appear to be informative. Statistical features are used most often (66 nodes, using 17 distinct features), followed by linguistic features (35 nodes, 16 features), related ct features (28 nodes, 9 features), morphological/shape features (25 nodes, 9 features), and frequency features (16 nodes, 11 features).

The most discriminating feature (first node in the decision tree) is Vintar's termhood score (Vintar, 2010), calculated for the original, unlemmatised ct, compared to a reference corpus of newspaper articles. This is also the feature that, following domain consensus, is used most often (10 times and 8 times, respectively). The most frequently used features in the other categories are: number of characters (morphological/shape feature used 7 times), number of cts that contain the current ct (related feature used 6 times), the presence of either a preposition or a noun (linguistic features, both used 4 times). The frequency features are all used 0-2 times and none stand out. A possible explanation for the comparative irrelevance of frequency features, is that frequency is most informative when already incorporated into termhood or unithood measures and that many frequency features are strongly correlated.

A feature indicating presence in a list of stopwords is not used, even though lists of stopwords are generally very useful for ATE. This may be related to the limited list used for Dutch (414 tokens) or the way it is currently implemented (only complete matches are counted). This analysis shows how the statistical termhood and unithood features are indeed most useful for ATE, but that there are many other informative features as well, in a range of different categories.

4.3 Precision, Recall and F1-scores

HAMLET extracts 1352 cts with a precision of 55.03%, a recall of 56.11% and an f1-score of 55.56%. TermoStat extracts many more cts (4671) and has a much lower precision of only 18.18% but a higher recall at 64.03%, resulting in an f1-score of 28.31%. This is where the supervised ML component becomes immediately apparent: HAMLET is trained to optimise f1-score, whereas the cut-off point for TermoStat had to be set manually based on a limited set of experiments. Figures 1 and 2 show the precision, recall and f1-score curves for HAMLET and TermoStat. In this case, HAMLET did not print the predicted label of the classifier, but the predicted probability of label 1, i.e. the predicted probability that the ct is a true term. In Figure 1, only terms with a probability higher than 50% (up until rank 1352) were labelled as terms by HAMLET. However, for the sake of comparison with TermoStat, the graph was calculated supposing that all 4671 highest ranked cts were predicted as terms. As can be seen in the graph, the decision boundary is very close to the highest possible f1-score. According to this ranking, that would have been 57.05%, if HAMLET had extracted the highest ranked 1619 cts instead of the first 1352. The TermoStat results in Figure 2 show a different trend. Here, the ideal cut-off point would have been after the 1307th highest ranked term (Specificity of 16.06), which would have resulted in an f1-score of 42.61%. Instead, 3362 more terms were extracted, causing a large drop in f1-score.

Another notable peculiarity in these curves is that the TermoStat curves are smoother and follow a more predictable pattern: precision starts high and decreases gradually, recall increases but starts to slowly flatten out. The recall curve for HAMLET follows this pattern and even reaches over 80% at rank 4671, where TermoStat's recall is still only at 64%. However, HAMLET's precision curve is far from smooth in the beginning, with the highest precision only around rank 285. These fluctuations are due to two factors. First, precision curves are very susceptible to small changes at the start, when it is calculated for few examples. Second, surprisingly, HAMLET's predicted probability that a ct is a true term does not always correspond with the reality. For instance, 13 cts were given a 100% probability and only 7 of these were actual terms. So, while the predicted true term probability for these cts was 100%, the actual precision was only 54%. One of the false positives should have been in the gold standard and was missed by the annotators. Two were parts of terms and the remaining three were very common words: *bovenstaande* (above), *moet* (has to), and *werd* (became). Further research is needed to explain this behaviour and compare results with other algorithms and corpora.

4.4 Term Labels

While the extractors only performed binary classification, the gold standard does contain more detailed labels (Specific Terms, Common Terms, OOD Terms and Named Entities, see section 3.1). It was already established that TermoStat extracts many more terms, resulting in a lower precision but also a higher recall. An additional analysis can show whether both tools extract the same term types based on the more fine-grained labels. Regarding these labels, two hypotheses were formulated. First, we expect HAMLET to be better than TermoStat at extracting Named Entities and maybe also OOD Terms, since these were included in the training data, while TermoStat's Specificity score is designed mostly to detect domain-specific terms, i.e. Specific and Common Terms. TermoStat may still extract Named Entities and OOD Terms, since they share many characteristics with the other two categories, but the hypothesis is that it will extract comparatively fewer than HAMLET. This hypothesis was partly confirmed by the results. Even though HAMLET extracts fewer terms in total, it extracts more Named Entities than TermoStat (63 versus 43) and a larger percentage of all HAMLET's extractions are Named Entities (5% versus 1%). For OOD Terms the hypothesis could not be confirmed. since the difference was too small. This may be due, at least

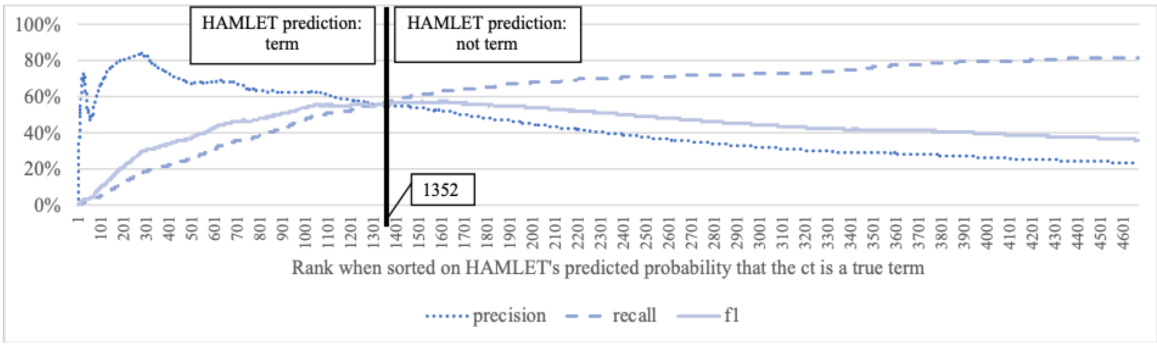


Figure 1: Precision, recall and f1-curves of HAMLET, including the 4671 highest ranked cts based on predicted probability that the ct is a true term; black line indicates boundary between cts that were predicted to be terms and those that were predicted not to be terms.

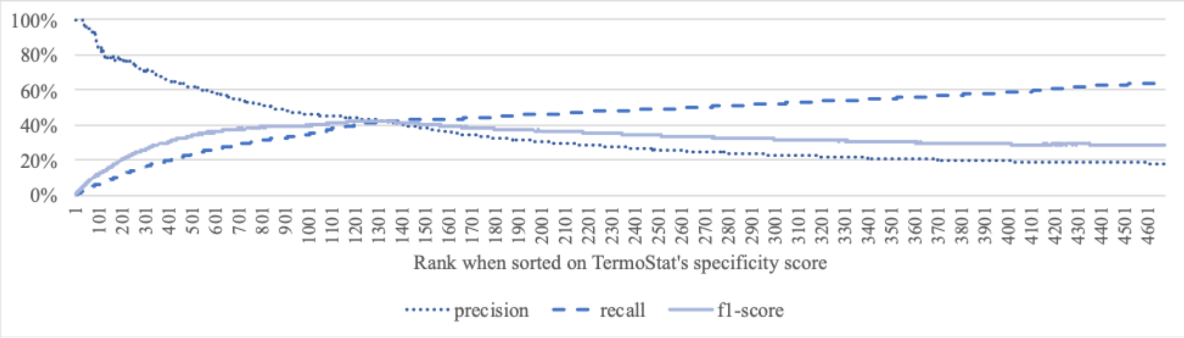


Figure 2: Precision, recall and f1-score curves of TermoStat for all 4669 extracted terms, ranked based on specificity score

in part, to the annotation. Since the corpus subject was dressage (a subdomain of equitation), rather than equitation as a whole, many terms that are specific to other branches of equitation were annotated as OOD Terms. These are nearly all terms related to other equitation disciplines, such as *gymkhana* (same in English) or *voltige* (equestrian vaulting). Had the annotation been slightly less strict about the domain-specificity, at least 27 of the 34 OOD Term annotations would have been Specific Terms. This illustrates how a subjective decision about whether or not to include a certain group of terms, can have a large impact on the results.

The second hypothesis concerns Specific Terms: we expect HAMLET to outperform TermoStat for Specific Terms. TermoStat relies heavily on a single termhood measure, which means it has the typical drawback of being very sensitive to frequency, leading to low recall on rare terms. HAMLET combines many more features, which may mean that it is less sensitive to frequency. This is important for Specific Terms, since they are often rare. The average relative frequency of Specific Terms versus Common Terms in the domain-specific corpus, calculated by HAMLET is 0.0001268 versus 0.0003642 (similar for document frequency). Again, the hypothesis could only partially be confirmed. HAMLET extracts fewer Specific terms than TermoStat (540 versus 626), though this is similar when considering the comparative difference in total number of extracted terms. However, HAMLET does extract more hapax terms (291 versus 241 by TermoStat), despite extracting fewer terms in total, confirming the part of the hypothesis about HAMLET's improved ability to extract rare terms.

4.5 Agreement between HAMLET and TermoStat

The agreement between HAMLET and TermoStat is very low, with a Cohen's Kappa score of only 0.162. Part of the disagreement is due to the much higher number of non-terms extracted by TermoStat, but even agreement on true terms is only slightly more elevated (0.28). These numbers indicate that the two tools have different strengths and weaknesses. In previous sections, two main strengths of the supervised approach were already discussed: it is better at optimising for f1-score and it is better at extracting rare terms. The variety of features also visibly results in other improvements. For instance, there is a feature indicating the presence of a dash at the end of a ct. HAMLET has incorporated this feature into the decision tree and extracts only 3 wrong cts that begin or end with a dash. This is not included in TermoStat's preprocessing, resulting in 43 wrong extractions. This does not always explain the results, as illustrated by the fact that the feature indicating the presence of digits in a ct is never used, but HAMLET still correctly extracts 10 out of 21 gold standard terms with digits, whereas TermoStat does not recognise any. Another notable result is that TermoStat extracts 90 cts that begin or end with an article (compared to only 5 such error extracted by HAMLET). These types of mistakes were actually expected from HAMLET, rather than TermoStat, since HAMLET selects cts based on a list of POS patterns that is not manually validated and includes wrong patterns. The fact that only TermoStat makes this error, indicates that the former may have learnt to exclude such cts, while the latter may include wrong patterns, due to its susceptibility to human error. Another possibility is that the POS tagger used by TermoStat is less accurate, resulting in more such errors.

There are also disadvantages to the supervised method, specifically due to the differences between training and test data. For instance, single letters, indicating certain positions in a dressage arena, can be terms. This is not be the case in most other domains, so a supervised system may learn rules that obstruct the extraction of single-character terms. HAMLET only extracts 3 out of 10 single-character terms in the gold standard, while TermoStat extracts 6. This is an illustration of how domain-dependent term characteristics can be and how this could impact supervised systems. Furthermore, HAMLET's lower sensitivity to frequency is not only an advantage but can also backfire. A few seemingly obvious terms with very high frequencies are not extracted, e.g., *hulpen* (aids) and *hand* (meaning both literally hand, but also the direction the horse is going in the arena). Even *paarden* (horses) received a 0% probability of being a term by HAMLET. A final category of terms both extractors struggle with, are those that are also part of general language and only become terms in this context. An example is *pijp*, which usually means pipe, but, in the context of dressage, refers to a part of a horse's leg. At least half of the terms that were not found by either tool concern terms that are also part of general language.

	H = 1	**H = 0**	**SUM**
TS = 1	852	3819	4671
TS = 0	500	9360	9860
SUM	1352	13179	14530

Table 1: Agreement between TermoStat (TS) and HAMLET (H); =0.162

The described differences illustrate various strengths and weaknesses of both approaches and inspires a few suggestions for improvement. TermoStat's approach could benefit from more elaborate preprocessing (e.g., removing cts ending in a dash) and an evaluation of the POS patterns. The supervised approach is clearly influenced by the domain-dependence of term characteristics and could benefit from in-domain training data or training data in more domains. The two approaches are at least partly complementary and a combination of the output results in a recall of 77.45%, which is high, considering the strictness of the evaluation and the gold standard.

4.6 Agreement Between Tools and Gold Standard

Even though the gold standard was rigorously annotated, there is always the possibility of human error and the ambiguous nature of terms, which means that these annotations are not the only possible correct annotations. Therefore, it is worth looking at the ATE results in more detail. Are there any terms that should have, or could have been annotated among the false positives? Or the opposite: terms which could or should not have been annotated among the false negatives? Are the mistakes made by the tools understandable or undeniably wrong? In an attempt to answer these questions, HAMLET's results were analysed in more detail.

Only a single annotation was found to be undeniably wrong: *veel* (many) was mistakenly annotated as a term. However, there were 76 others which were labelled: should (not) have been annotated, including terms such as *uitzwaaien* (wrong positioning of the horse's hindquarters, mostly during a turn), and *verruim* (specific way of lengthening the horse's stride; other forms of this verb were annotated correctly). Looking at the 608 false positives, at least 217 of them could have been terms, which would increase precision to over 70%. It implies that there is at least some logic in the errors and that, overall, HAMLET does appear to have learnt informative general characteristics of terms. However, this analysis should also be interpreted as a cautionary tale regarding ATE evaluation. When evaluating a list of already extracted cts, annotators are biased to evaluate favourably. Therefore, results compared to a predetermined gold standard may tend to be worse than results based on the annotation of the ATE output. Any comparisons between such results should be interpreted with due caution.

Nevertheless, it is encouraging to see logical patterns in the ATE results. For instance, many cts were extracted related to body parts (of both horse and rider). These were not always consistently annotated but are still logical terms in the field of dressage, which is a sport where the positioning of horse and rider are crucial. At least 171 cts extracted by HAMLET were related to body parts or bodily functions (e.g., *schuimproductie*, the production of foam in the horse's mouth). They were actually extracted by HAMLET more consistently than they were labelled by the human annotator. Also encouraging was the fact that, despite only very limited information about term variation, HAMLET often makes the same decision for related terms, such as terms with different full forms sharing the same lemma. Still, TermoStat's strategy of grouping terms with the same lemma is more effective and should be considered as an option to improve HAMLET.

One last item to mention here is that HAMLET is still susceptible to classic ATE errors, such as wrongly extracting parts of terms, combinations of different terms, or very frequent terms in combination with a non-term. For instance, 35 false positives contain the word *paard* or *paarden* (horse(s)), but in combinations that are not terms, e.g., *paard gaat* (horse goes), *paard niet* (horse not), and *paard symmetrisch* (horse symmetrical). These are typical errors because such combinations are much more frequent in the domain-specific corpus than in reference corpora, so they get high termhood values. Even though HAMLET still makes these mistakes, there is a marked improvement compared to TermoStat, which relies more heavily on termhood statistics. For instance, TermoStat wrongly extracts 320 cts that contain *paard(en)*, compared to only 35 for HAMLET. This further illustrates the positive effect of multiple features to limit frequency-related errors.

5 Conclusions and Future Research

The research described in this paper presents an elaborate evaluation of a supervised ML approach to automatic term extraction (HAMLET), compared to a traditional system without training data (TermoStat). As expected, the supervised system obtains higher f1-scores by combining features with various types of information and optimising f1-score. A closer look at the results confirms that the system has clearly learnt informative general characteristics of terms. It is less reliant on frequency, leading to fewer mistakes on rare terms or frequent non-terms. However, the supervised system also has a distinct weakness, namely its domain-dependence, since it was trained on out-of-domain data. This emphasises the need for annotated data, though there are also indications that very little training data could suffice (Amjadian et al., 2018). Nevertheless, annotated data remains critical for a nuanced evaluation.

Current versions of HAMLET can already obtain an average f1-score of 53%, using cross-validation on all domains and languages combined. Preliminary results already show the impact of factors such as algorithm, language, domain, term definition, and in-domain training data, with f1-scores of up to 66% depending on the combination. Precision and recall are not always as balanced as for the presented use-case, and results vary greatly per corpus. Future research will concentrate on further exploring the robustness of HAMLET, with more contrasting results for different configurations and data. Aside from the binary classifier, a sequence labelling approach, which is further removed from the original methodology, will also be explored and will provide further material for comparison.

6 Acknowledgements

This research has been carried out as part of a PhD fellowship on the EXTRACT project, funded by the Research Foundation Flanders.

References

Ehsan Amjadian, Diana Inkpen, T.Sima Paribakht, and Farahnaz Faez. 2016. Local-Global Vectors to Improve Unigram Terminology Extraction. In *Proceedings of the 5th International Workshop on Computational Terminology*. Osaka, Japan, pages 2–11.

Ehsan Amjadian, Diana Zaiu Inkpen, T. Sima Paribakht, and Farahnaz Faez. 2018. Distributed specificity for automatic terminology extraction. *Terminology* 24(1):23–40. https://doi.org/10.1075/term.00012.amj.

Jérôme Azé, Mathieu Roche, Yves Kodratoff, and Michèle Sebag. 2005. Preference Learning in Terminology Extraction: A ROC-based approach. In *Proceeedings of Applied Stochastic Models and Data Analysis*. Brest, France, pages 209–2019. ArXiv: cs/0512050. http://arxiv.org/abs/cs/0512050.

Béatrice Daille. 1994. Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In J. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Aproaches to Language*, MIT Press, Massachusetts, pages 49–66.

Boris Dobrov and Natalia Loukachevitch. 2011. Multiple evidence for term extraction in broad domains. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*. Hissar, Bulgaria, pages 710–715.

Patrick Drouin. 2003. Term Extraction Using Non-Technical Corpora as a Point of Leverage. *Terminology* 9(1):99–115.

Anna Hätty and Sabine Schulte im Walde. 2018. Fine-Grained Termhood Prediction for German Compound Terms Using Neural Networks. In *Proceedings of the Joint Workshop on,Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*. Sante Fe, New Mexico, USA, pages 62–73.

Alex Judea, Hinrich Schütze, and Sören Brügmann. 2014. Unsupervised training set generation for automatic acquisition of technical terminology in patents. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical Papers*. Dublin, Ireland, pages 290–300.

Kyo Kageura and Bin Umino. 1996. Methods of automatic term recognition. *Terminology* 3(2):259–289.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(1):180–182.

Maren Kucza, Jan Niehues, Thomas Zenkel, Alex Waibel, and Sebastian Stüker. 2018. Term Extraction via Neural Sequence Labeling a Comparative Evaluation of Strategies Using Recurrent Neural Networks. In *Interspeech 2018*. ISCA, Hyderabad, India, pages 2072–2076. https://doi.org/10.21437/Interspeech.2018-2017.

Nikola Ljubei, Darja Fier, and Toma Erjavec. 2019. KAS-term: Extracting Slovene Terms from Doctoral Theses via Supervised Machine Learning. *arXiv:1906.02053 [cs]* ArXiv: 1906.02053. http://arxiv.org/abs/1906.02053.

Lucelene Lopes and Renata Vieira. 2015. Evaluation of cutoff policies for term extraction. *Journal of the Brazilian Computer Society* 21(1). https://doi.org/10.1186/s13173-015-0025-0.

Natalia Loukachevitch. 2012. Automatic Term Recognition Needs Multiple Evidence. In *Proceedings of LREC 2012*. ELRA, Istanbul, Turkey, pages 2401–2407.

Lieve Macken, Els Lefever, and Véronique Hoste. 2013. TExSIS: Bilingual Terminology Extraction from Parallel Corpora Using Chunk-based Alignment. *Terminology* 19(1):1–30.

Nokel, Michael, Bolshakova, E.i., and Loukachevitch, Natalia. 2012. Combining multiple features for single-word term extraction. In *Proceedings of Dialog 2012*. pages 490–501.

Alexandre Patry and Philippe Langlais. 2005. Corpus-Based Terminology Extraction. In *Terminology and Content Development - Proceedings of the 7th International Conference on Terminology and Knowledge Engineering*. Copenhagen, Denmark, pages 313–321.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. 2011. Scikit-learn: Machine Learning in Python. *MACHINE LEARNING IN PYTHON* (12):2825–2830.

Behrang Qasemizadeh and Anne-Kathrin Schumann. 2016. The ACL RD-TEC 2.0: A Language Resource for Evaluating Term Extraction and Entity Recognition Methods. In *Proceedings of LREC 2016*. ELRA, Portoro, Slovenia, pages 1862–1868.

Ayla Rigouts Terryn, Véronique Hoste, Joost Buysschaert, Robert Vander Stichele, Elise Van Campen, and Els Lefever. 2019. Validating multilingual hybrid automatic term extraction for search engine optimisation: the use case of EBM-GUIDELINES. *Argentinian Journal of Applied Linguistics* 7(1):93–108.

Jorge Vivaldi and Horacio Rodríguez. 2001. Improving term extraction by combining different techniques. *Terminology* 7(1):31–48. https://doi.org/10.1075/term.7.1.04viv.

Piek Vossen, editor. 1998. *EuroWordNet: A multilingual database with lexical semantic networks*. Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-017-1491-4.

Petra Wolf, Ulrike Bernardini, Christian Federmann, and Hunsicker Sabine. 2011. From statistical term extraction to hybrid machine translation. In Mikel L. Forcada, Heidi Depraetere, and Vincent Vandeghinste, editors, *Proceedings of the 15th Conference of the European Association for Machine Translation*. Leuven, Belgium, pages 225–232.

Yuhang Yang, Hao Yu, Yao Meng, Yingliang Lu, and Yingju Xia. 2011. Fault-Tolerant Learning for Term Extraction. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*. Sendai, Japan, pages 321–330.

Yu Yuan, Jie Gao, and Yue Zhang. 2017. Supervised Learning for Robust Term Extraction. In *The proceedings of 2017 International Conference on Asian Language Processing (IALP)*. IEEE.

Distant Supervision for Sentiment Attitude Extraction

Nicolay Rusnachenko[1], **Natalia Loukachevitch**[1,2], **Elena Tutubalina**[3,4]
[1] Bauman Moscow State Technical University, Moscow, Russia
[2] Lomonosov Moscow State University, Moscow, Russia
[3] Kazan Federal University, Kazan, Russia
[4] Samsung-PDMI Joint AI Center, PDMI RAS, St. Petersburg, Russia
kolyarus@yandex.ru, louk_nat@mail.ru, elvtutubalina@kpfu.ru

Abstract

News articles often convey attitudes between the mentioned subjects, which is essential for understanding the described situation. In this paper, we describe a new approach to distant supervision for extracting sentiment attitudes between named entities mentioned in texts. Two factors (pair-based and frame-based) were used to automatically label an extensive news collection, dubbed as RuAttitudes. The latter became a basis for adaptation and training convolutional architectures, including piecewise max pooling and full use of information across different sentences. The results show that models, trained with RuAttitudes, outperform ones that were trained with only supervised learning approach and achieve 13.4% increase in F1-score on RuSentRel collection.[1]

1 Introduction

Relation extraction nowadays remains one of the popular tasks in the natural language processing domain. The relation types to be extracted from texts may vary and result in different tasks: semantic classification of relations between a pair of common nominals (Hendrickx et al., 2009), source-target sentiment relation extraction (Ellis et al., 2014), opinion expression towards entities and events (Deng and Wiebe, 2015a), attitude extraction between mentioned named entities (Rusnachenko and Loukachevitch, 2018), etc.

Dealing with one of these tasks, the greatest difficulty one encounters is the complexity of the sentence structure. As for analytical articles, the idea expressed by the author could be conveyed in different variants, which is a feature of natural languages.

When relation extraction is performed automatically using machine learning approaches, this complexity results in a lack of training examples. One technique that helps to accomplish this task is *distant supervision* (DS), initially proposed by (Mintz et al., 2009). It assumes to extract and label data by relying on assumptions based on a prepared knowledge base. Although many methods have been proposed in such domains as sentiment analysis and relation extraction (Turney, 2002; Zeng et al., 2015), the domain of sentiment attitude extraction remains understudied.

This paper describes a new approach to distant supervision for extracting sentiment attitudes between named entities mentioned in texts. It is worth noting that DS faces the problem of wrong labels, which becomes a reason of noisy labeled data. To address the shortcomings of noisy labeling, in this paper we exploit two primary sources of automatic annotation:

- Prior knowledge about current attitudes between political entities (figures);
- Sentiment frames that define attitudes between participants of a situation.

The obtained corpus annotated with attitudes was used to train convolutional neural networks (CNNs), adapted for relation extraction and full use of information across multiple sentences. Our key contributions in this work are two-fold:

- We propose a workflow of automatic sentiment attitudes extraction, which exploits news title simplicity to perform annotation;
- We conduct extensive experiments on RuSentRel (Loukachevitch and Rusnachenko, 2018) and the results demonstrate that CNNs trained on two types of training data achieve F1-score increase by 13.4% over models that do not employ obtained corpus in training.

[1] The code is available on https://github.com/nicolay-r/attitudes-extraction-ds

Proceedings of Recent Advances in Natural Language Processing, pages 1022–1030,
Varna, Bulgaria, Sep 2–4 2019.

2 Related Work

The task of attitude recognition toward named entities or events, including opinion holder identification from full texts did not attract much attention. In 2014, the TAC evaluation conference in Knowledge Base Population (KBP) track included so-called sentiment track (Ellis et al., 2014). The task was to find all the cases where a query entity (sentiment holder) holds a positive or negative sentiment about another entity (sentiment target). Thus, this task was formulated as a query-based retrieval of entity-sentiment from relevant documents and focused only on query entities[2].

MPQA 3.0 (Deng and Wiebe, 2015b) is a corpus of analytical articles with annotated opinion expressions (towards entities and events). The annotation is sentence-based. For example, in the sentence «When the Imam issued the fatwa against Salman Rushdie for insulting the Prophet ...», Imam is negative to Salman Rushdie but is positive to the Prophet.

In paper (Choi et al., 2016), authors studied the approach to the recovery of the documents attitudes between subjects mentioned in the text. The approach considers such features as frequency of a named entity in the text, relatedness between entities, direct-indirect speech, etc. The best quality of opinion extraction obtained in this work was only about 36% F-measure, which shows that the necessity of improving extraction of attitudes at the document level is significant and this problem has not been sufficiently studied.

A corpus of analytical articles, obtained from authoritative foreign sources and translated into Russian has been invented in (Loukachevitch and Rusnachenko, 2018). The collected articles contain both the author's opinion on the subject matter of the article and a large number of attitudes mentioned between the participants of the described situations. Authors experiment with automatic attitudes extraction within the developed corpus. In comparison with (Choi et al., 2016) where documents much smaller and written in English, authors mentioned the closest F-measure and conclude that the task still remains complicated.

Each attitude may be considered in terms of related article context, or sentence. The sentence consists of words which could be gathered and treated as an *embedding*, where each word represents a feature vector. Convolving embedded sentence representation by a set of different filters, in paper (Zeng et al., 2014) authors implemented and trained the Convolutional Neural Network (CNN) model for the relation classification task. Being applied for the SemEval-2010 Task 8 (Hendrickx et al., 2009), the obtained model significantly outperformed the results of other participants.

This idea was developed further in terms of *max-pooling* operation (Zeng et al., 2015). This is an operation, which is applied to the convolved by filters data and extracts the maximal values within each convolution. However, for the relation classification task, original max-pooling reduces information extremely rapid and blurs significant relation aspects. Authors proposed to treat each convolution in parts. The division into parts depends on attitude entities: *inner* (between entities), and *outer*. This approach resulted in an advanced architecture model and was dubbed as "Piecewise Convolutional Neural Network" (PCNN).

3 Resources

This section describes resources (collections and lexicons) that were used for the dataset annotation.

3.1 RuSentRel Collection

In our experiments, we use the RuSentRel corpus[3] consisted of analytical articles from Internet-portal `inosmi.ru` (Loukachevitch and Rusnachenko, 2018) devoted to international relations. In this corpus, the manual annotation of the sentiment attitudes towards mentioned named entities had been carried out at the document level. The annotation is subdivided into two subtypes:

- The author's relation to mentioned named entities;
- The relation of subjects expressed as named entities to other named entities.

An analytical document can refer to an entity with several variants of naming (*Vladimir Putin – Putin*), synonyms (*Russia – Russian Federation*), or lemma variants generated from different wordforms. For correct inference of attitudes between named entities in the whole document, the corpus is provided with a list of variant names for the same entity found in the corpus.

[2] https://tac.nist.gov/2014/KBP/Sentiment/index.html

[3] https://github.com/nicolay-r/RuSentRel/tree/v1.1

In this paper, we utilize RuSentRel corpus in experiments for the proposed approach. Table 1 describes the corpus statistics.

Parameter	Value
Number of documents	73
Total opinion pairs	1361
Sentences (avg./doc.)	105.75
Opinion pairs (avg./doc.)	18.64
Positive opinion pairs (avg./doc.)	8.71
Negative opinion pairs (avg./doc.)	9.93
Avg. dist. between named entities within a sentence in words	10.2

Table 1: Attitude statistics of RuSentRel-v1.1 corpus.

3.2 RuSentiFrames Lexicon

The RuSentiFrames[4] lexicon describes sentiments and connotations conveyed with a predicate in a verbal or nominal form (Rashkin et al., 2016; Klenner and Amsler, 2016). The structure of the frames includes the set of predicate-specific roles and frame dimensions.

For role designation, the approach of PropBank (Palmer et al., 2005) is used. In this approach, individual verb's semantic arguments are numbered, beginning with zero. For a particular verb, Arg0 is generally the argument exhibiting features of a Prototypical Agent (Dowty, 1991), while Arg1 is a Prototypical Patient or Theme.

In the main part of the frame, the following dimensions are described:

- the attitude of the author of the text towards mentioned participants;
- positive or negative sentiment between participants;
- positive or negative effects to participants;
- positive or negative mental states of participants related to the described situation.

All assertions are provided with the score of confidence, which currently has two values: 1, if this assertion is true almost always, or 0.7, the assertion is considered in default. We do not describe assertions about neutral sentiment, effect or state of participants.

Type of lexical unit	Number
Verbs	2 794
Nouns	822
Phrases	2 401
Other	49
Unique entries	6 036
Total entries	6 412

Table 2: Quantitative characteristics of the RuSentiFrames entries.

The created frames are associated not only with a single entry but with a "family" of related words and expressions, which have the same attitudes. The following lexical units can be associated with a sentiment frame: single words, idioms, light verb constructions, and some other multiword expressions.

Currently, RuSentiFrames contains 277 frames with 6,412 associated *frame entries*. Table 2 shows the distribution of the RuSentiFrames entries according to parts of speech (POS) and other characteristics. Let us consider frame "Одобрить" (Approve) presented in Example 1.

Example 1: Frame "Одобрить" (Approve)

```
"roles": {"a0": "who approves",
          "a1": "what is approved"}
"polarity": {["a0", "a1", "pos", 1.0],
             ["a1", "a0", "pos", 0.7]},
"effect": {["a1", "pos", 1.0]},
"state": {["a0", "pos", 1.0],
          ["a1", "pos", 1.0]}
```

Nowadays, the lexicon is under development. For the proposed distant supervision approach, we utilize only the dimension of attitudes towards Prototypical Patient conveyed by Prototypical Agent. Table 3 provides related statistics.

Effect	Sentiment	Number
A0 → A1	pos	2 252
A0 → A1	neg	2 802

Table 3: The distribution of RuSentiFrames text entries according to attitudes.

3.3 News Collection

The collection to be used for sentiment attitude extraction consists of Russian articles and news of major news sources, specialized political sites, and

[4] https://github.com/nicolay-r/RuSentiFrames/tree/v1.0

Russian sites of world known news agencies published in 2017.

Each article is separated into the title and the contents. The collection statistics presented in Table 4.

Parameter	Value
Number of documents	2.8×10^6
Sentences (avg./doc.)	13.24

Table 4: News collection statistics.

4 Automatic Forming of Training Collection for Sentiment Attitude Extraction

This section discusses two different methods of sentiment attitude annotation: pair-based, and frame-based. Both methods apply to the title as it provides the main idea of the article and usually has a relatively simple sentence structure. Figure 1 illustrates the collection development flow. Further subsections describe the flow components in detail.

4.1 Text Processing and Named Entity Recognition

For attitude extraction, it is necessary to parse a text. This process involves the *tokenization* to demarcate text string into words and punctuation signs. Numbers and URL-links are considered as non-meaningful and masked.

Each attitude is based on a pair of named entities. For named entity recognition (NER, Figure 1), we utilize the following resources:

1. The pre-trained neural network model[5], which is state-of-the-art for the Russian language (Burtsev et al., 2018);

2. The list of named entities from RuSentRel corpus, organized in *authorized objects*.

The list of authorized objects is necessary to avoid accidental misses from the NER model.

The text of news articles may refer to an entity in several naming variants (Putin – Vladimir Putin), and synonyms (EU – Europe). To match named entity synonyms, in this paper we utilize both stemming[6] and list of synonyms, provided along with the RuSentRel corpus.

4.2 Pair-Based Annotation

This attitude annotation method utilizes the preassigned attitudes organized in a *list of pairs* (Figure 1).

Given a processed title with labeled named entities set E, we select pairs $\{\langle e_i, e_j\rangle \mid e_i, e_j \in E\}$, suitable for sentiment attitudes role. For relevant pairs filtering, the following restrictions should be met:

1. The presence of synonymous attitude in a given attitudes list;

2. All the entities appeared between pair endings should be authorized objects;

However, in a specific sentence, the supposed relation between countries can be false. For example, in the sentence "Зрители смогут увидеть показательные выступления спортсменов - чемпионов России и Европы" (Spectators will be able to see demonstrations of athletes - champions of Russia and Europe), prior negative relations between Russia and Europe are not mentioned. Therefore we need an additional factor to provide the quality of the annotation, and the RuSentiFrames lexicon can be used as such a factor.

4.3 Frame-Based Annotation

This attitude annotation method utilizes frame entries from the RuSentiFrames lexicon. Given a processed title with labeled named entities set E, an entry pair with $e_i, e_j \in E$, where e_i appears before e_j, considered as sentiment attitude when all the following criteria are met:

- All the frame entries between e_i and e_j have polarity;

- All the entities that appeared between e_i and e_j should be authorized.

We assign a positive sentiment score when all the polarities of inner frame entries have a positive sentiment. Otherwise, it assigns the negative sentiment score. We also consider frame entry polarity as *inverted*, when it is used with "не" (not) particle.

4.4 Attitude Filtering

To combine the annotation methods described above (attitudes filter, Figure 1), we intersect the annotations and separate the intersection into the

[5] https://github.com/deepmipt/ner

[6] https://tech.yandex.ru/mystem/

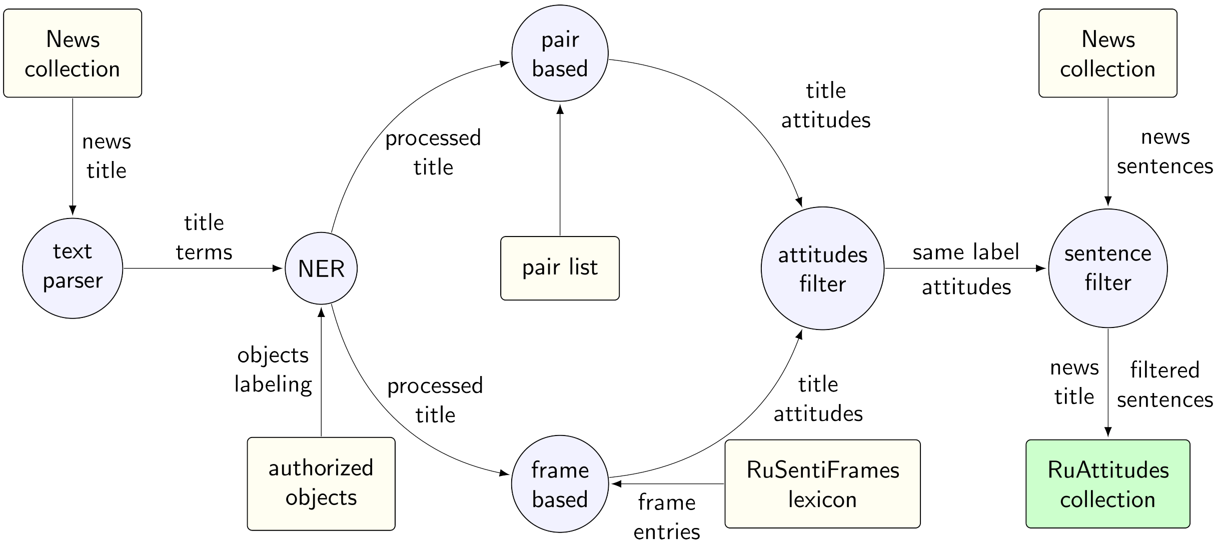


Figure 1: Training collection development flow

following sets: (i) with the same polarity and (ii) with different polarity according to both sources.

In the case of the non-empty set with the same polarity (SAME), at last step we utilize *sentence filter* (Figure 1). Given a set of processed news sentences, we select those which contain at least a single entity pair, presented in the SAME set.

Corpus	doc. level attitudes	texts count	titles and sentences
Pair-Based	60 788	52 377	136 496
Frame-Based	55 566	43 383	104 205
Intersection	22 589	20 885	50 958
Different	7 929	7 435	17 939
Same RuAttitudes	14 660	13 450	33 019

Table 5: The statistics of automated annotation of texts and sentences.

Finally, the workflow (Figure 1) is applied to the news collection (Section 3.3), and we obtain the **RuAttitudes**[7] dataset, automatically labeled with sentiment attitudes between named entities. Table 5 provides statistics separately for each step. Evaluated accuracy of randomly selected sentences from different texts presented in Table 6.

5 Convolutional Neural Networks for Attitude Classification

For automatic sentiment attitude classification, the following CNN-based architectures were used:

Corpus	Accuracy
Pair-Based	67.0
Frame-Based	62.5
RuAttitudes	89.0

Table 6: Accuracy of attitude annotation in the generated collections.

- Classic CNN (Zeng et al., 2014);
- Piecewise-CNN (Zeng et al., 2015);

To predict the attitude polarity, both models utilize sentences in the input. Given a context (a set of sentences) with a mentioned attitude in it, the output of models is the class label. Two different approaches to training were considered: single-sentence training and multi-sentence training.

5.1 Sentence Embedding

We use *sentence embedding* to present sentences in model input. This is a matrix with rows related to words or token (for example, punctuation marks) embeddings.

For words, we look up for vectors in precomputed and publicly available Word2Vec model[8] based on news articles with *window size* of $w = 20$. For tokens, we utilize a set of predefined types (size of 17), where each type is a randomly initialized vector of the same size as word vector.

Each word and token vectors have been additionally expanded with the following features:

[7] https://github.com/nicolay-r/RuAttitudes/tree/v1.0

[8] http://rusvectores.org/static/models/rusvectores2/news_mystem_skipgram_1000_20_2015.bin.gz

- Distance embedding (Rusnachenko and Loukachevitch, 2018) – is vectorized distance in words from entities e_i and e_j of an entry pair $\langle e_i, e_j \rangle$ to a given word or token;

- Part-of-speech (POS) tags embedding; we use "unknown" tag in case of tokens.

For features, we use randomly initialized vectors. Table 7 provides parameter values of each embedding described above.

Type	Parameters	Values
POS	v_{size}	5
Distance	v_{size}	5
Tokens	$\langle size, l_t \rangle$	$\langle 17, 10^3 \rangle$
Words	$\langle size, l_w, w \rangle$	$\langle 147 \cdot 10^3, 10^3, 20 \rangle$

Table 7: Embedding parameters, where v_{size} is the size of embedding vectors.

5.2 Single Sentence Training

This training process assumes to predict a sentiment label by a single sentence. Given an attitude context, we consider that each sentence should be labeled with an attitude sentiment.

We utilize training process described in (Rusnachenko and Loukachevitch, 2018). The input organized in minibatches, which yields n *bags*.

Each bag has a set of m sentences $\{s_1, \ldots, s_m\}$, where $s_j = \langle e_s, y \rangle$ includes sentence embedding e_s and related label $y \in \mathbb{R}^c$. The training process is iterative, and iteration includes the following steps:

1. Composing a minibatch $I = \{b_1, \ldots, b_n\}$ where $b_i = \{s_1, \ldots, s_m\}$;

2. Performing a forward propagation through the network; the result is a vector $\{o_k\}_{k=1}^q$, where $o_k \in \mathbb{R}^c$, and $q = n \cdot m$;

3. Computing cross entropy loss for output:

$$l_k = \sum_{j=1}^{c} \log p(y_i | o_{k,j}; \theta), \; k \in \overline{1 \ldots q} \quad (1)$$

4. Composing cost vector $\{cost_i\}_{i=1}^n$, where $cost_i = \max \left[l_{i \cdot g} \; \ldots \; l_{(i+1) \cdot g} \right)$ is a maximal loss within i'th bag;

5. Using *cost* to update hidden variables set;

5.3 Multiple Sentence Training

This training process assumes to predict a sentiment label for sentences set. This process refers to a single sentence case described in Section 5.2 with the following modifications:

- Each minibatch presented as a sequence of n bags $b_i = \langle E_s, y \rangle$, where E_s is a set of embedded sentences (Section 5.2);

- Result output vector $\{o_l\}_{l=1}^n$ obtained by an application of max-pooling operation over separately convolved context sentences (Jiang et al., 2016);

6 Datasets and Experimential Setup

We consider the problem of sentiment attitudes classification as a two-class task at the document level. We conduct three different experiments using the following datasets:

1. RA – RuAttitudes dataset, described in this paper (Section 4);

2. RSR – RuSentRel based dataset with sentence-level attitude labeling (Section 3.1);

3. RSR+RA – a combination of RSR and RA datasets;

By default, the RuSentRel dataset provides a document level attitude labeling. This labeling was used to complete RSR and therefore treat RuSentRel in the same format as RuAttitudes. We consider sentence-level attitudes as bidirectional pairs of named entities. We filter sentences from RuSentRel with sentiment pairs according to the following rule. For each sentence among the all documents we check, whether at least a single pair has the labeled attitude, presented in document attitude annotation.

Statistical comparison of the RSR and RuAttitudes datasets is presented in Table 8. We use 10-fold cross-validation (CV) in RA experiment. In other experiments, the 3-fold CV has been chosen due to a small number of documents in RSR dataset. In experiments with RSR+RA, the cross-validation procedure applies to RSR; the RuAttitudes dataset combines with each training block of RSR. It is worth noting that RuSentRel sentences have a larger amount of opinions per sentences (2.26), mainly due to the nature of their content – these are analytical articles, while RuAttitudes has been based on news reports. This fact makes experiments with the RSR dataset significantly challenging.

For models to be trained, we apply *named entity masking* for pairs due to: (i) omit entity-related feature dependency, and (ii) prevent models from learning the distribution of the latter.

Parameter	RuAttitudes	RSR
Documents	20 855	73
k-fold cross-validation	10	3
Opinions on sentence level	35 125	2 879
Negative opinion pairs	26 904	1 602
Positive opinion pairs	8 221	1 277
Avg. opinions per sentence	1.06	**2.26**
Avg. sentences per opinion	2.40	2.57

Table 8: Comparison of RuAttitudes and RuSentRel based (RSR) datasets for experiments.

Description	Parameters	Values
Minibatch	$\langle n, m\rangle$	$\langle 8, 3\rangle$
Optimiser	$\langle lr, \rho, \epsilon\rangle$	$\left\langle 0.1, 0.95, 10^{-6}\right\rangle$
Terms	k	50
Window size	w	3
Filters count	c	300
Dropout	ρ	0.9

Table 9: Predefined training parameters.

Table 9 illustrates model parameter values. Each minibatch has n bags. As for the sentence count per bag parameter, we select $m = 3$ to cover the average sentence count per opinion (Table 8). To translate the labels onto the document level, we utilize *average* function across all the sentences (sentence sets) of a given attitude. All the sentences were limited by k words, including tokens. Each word is considered in a lemmatized[9] form. The convolutional window size and the filters count were chosen according to (Zeng et al., 2015). We use the *adadelta* optimizer with parameters according to (Zeiler, 2012).

Several baselines were also added in experiments: **baseline_neg** – all pairs of named entities are labeled as negative; **baseline_rand** – pairs are labeled randomly according to the sentiment distribution in the training collection;

We measure average values of accuracy every five epochs. The training process terminates if one of the following conditions are met: (i) average epoch accuracy reaches 99%; (ii) the training error exceeds the prior related value (except RSR experiment). The latter exception is due to an unstable training process, that might be caused by the relatively small training set passed per a single epoch.

For this task, we adopt macroaveraged (over documents): F1-score (F_1^{PN}), precision (π^{PN}), and recall (ρ^{PN}).

7 Result Analysis and Discussion

Table 10 provides the results of both evaluated baselines and models for each dataset. For RuAttitudes dataset, where the negative class significantly exceeds positive, and most documents lack positive attitudes, both baselines show high values.

For a certain model and related experiments α and β, let $C_{\alpha,\beta}$ is a set of labeling contradictions of two experiments. All the pairs of contradictions could be then treated as correctly or wrongly labeled by each experiment. We let $C^*_{\alpha,\beta}$ as a subset of contradictions where β corrects the errors of α. This subset yields of only those pairs, where the sentiment class has been correctly defined in β. Table 11 provides the comparison statistics between RSR and RSR+RA experiments, separately for each model, where $|L|$ is an opinion pairs count (Table 1). While the contradiction represents 34% of total opinion pairs, with an average 61% of corrections and 38% of wrong labeling, we can conclude the average result error corrections are 7% in case of RSR+RA experiment (Table 10).

The contribution of RuAttitudes corpus in RSR+RA experiment could be considered in terms of frequencies of following entry types E: nouns, verbs, frames. Due to the task considered as sentiment attitudes classification, it is significant to separate statistics by positive and negative classes. To define a sentence class, we utilize sentiment of attitude that appears in it. We utilize semantic orientation (SO) function (Turney, 2002) to reveal a discrepancy in entries between sentiment classes:

$$SO(e) = PMI(e, pos) - PMI(e, neg) \quad (2)$$

where $PMI(e, c)$ is a *pointwise mutual information* of entry e and sentiment class c. For each entry type separately, we utilize Formula 2 towards RuAttitudes to complete a set of entries, bounded with positive ($SO(e) > 0$) and negative ($SO(e) < 0$) classes. We order these sets by descending of $|SO(e)|$ to select k most distinctive entries and complete RA^c_k subsets, $c \in \{pos, neg\}$.

[9] https://tech.yandex.ru/mystem/

Models	RA			RSR			RSR+RA		
	F_1^{PN}	π^{PN}	ρ^{PN}	F_1^{PN}	π^{PN}	ρ^{PN}	F_1^{PN}	π^{PN}	ρ^{PN}
baseline-neg	0.83	0.78	0.89	0.39	0.31	0.54	0.39	0.31	0.54
baseline-rand	0.72	0.75	0.71	0.49	0.51	0.48	0.49	0.51	0.48
CNN	0.91	0.88	0.94	0.52	0.52	0.55	0.63	0.62	0.66
PCNN	0.93	0.91	0.96	0.59	0.58	0.61	0.67	0.66	0.69
MI-CNN	0.91	0.88	0.94	0.57	0.56	0.60	0.62	0.60	0.65
MI-PCNN	**0.94**	0.92	0.96	**0.62**	0.60	0.64	**0.68**	0.67	0.70

Table 10: Result of single sentence (CNN, PCNN) and multiple sentence (MI-CNN, MI-PCNN) trained models in following experiments: RA – RA results trained on RA; RSR – RSR results trained on RSR; RSR+RA – RSR results trained on RSR+RA.

Model	$\lvert C_{2,3} \rvert$	$\lvert L \rvert / \lvert C_{2,3} \rvert$	$\lvert C^*_{2,3} \rvert / \lvert C_{2,3} \rvert$
CNN	468	0.34	0.63
PCNN	428	0.31	0.62
MI-CNN	488	0.36	0.58
MI-PCNN	442	0.35	0.60

Table 11: Contradiction statistics between experiments RSR (2) and RSR+RA (3).

E	Entry Value	$tf(e)$
N_{pos}	«поддержка» (support)	0.20
	«помощь» (help)	0.02
	«переговоры» (negotiations)	0.24
V_{pos}	«поддерживать» (to support)	**0.55**
	«начинать» (to start)	0.26
	«предлагать» (to suggest)	0.10
F_{pos}	«помочь» (to help)	0.23
	«начать» (to begin)	**0.68**
	«договориться» (to agree)	**0.98**
N_{neg}	«санкция» (sanction)	**0.78**
	«борьба» (fight)	**0.50**
	«отношение» (relation)	**0.96**
V_{neg}	«обвинять» (to blame)	0.05
	«вводить» (to introduce)	0.06
	«продлять» (to extend)	0.00
F_{neg}	«наказать» (to punish)	**0.59**
	«обвинить» (to blame)	0.06
	«бороться» (to fight)	0.44

Table 12: List of $k = 3$ most distinctive nouns (N), verbs (V), and frames (F) of RA_k^c with related frequencies in $C^*_{2,3}$; E is an entry type.

To assess how RuAttitudes effects on error corrections, we provide statistic of entries both appears in $C^*_{2,3}$ and RA_k^c. For each entry $e \in RA_k^c$ we calculate $tf(e, C^*_{2,3})$ – is an averaged (among all models) normalized term frequency of entry e in $C^*_{2,3}$. Table 12 lists three ($k = 3$) most distinctive entries by each entry type, where entries with $tf(e, C^*_{2,3}) > 0.5$ are bolded. It is possible to investigate that $C^*_{2,3}$ mostly saturated with positively bounded frames of RA_k^{pos} and negatively bounded nouns of RA_k^{neg}.

Conclusion

This paper proposes an approach to the automatic development of a train collection for the sentiment attitude extraction task in the news domain. The combination of two different techniques was used to provide the double-check and keep labeled results of the common intersection. The first proposed technique obtains contexts by a manually implemented list of pairs of named entities with sentiment scores. The other technique, on the contrary, extracts relations from contexts using sentiment frames. The latter became possible due to the assumption of title simplicity.

Sentiment attitude extraction was considered as a two-class classification task. This result analysis demonstrates the model classification improvements achieve 13.4% increase in F_1^{PN} when the latter being trained with the developed collection.

In further work, we plan to address the shortcomings in the following directions: to emphasize the difference between sentiment and non-sentiment relations and to reduce noisy labeling of the existed approach.

Acknowledgments

The reported study was funded by RFBR according to the research project № 19-37-50001. The development of Russian sentiment frames is supported by the RFBR research project № 16-29-09606.

References

Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, et al. 2018. Deeppavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*. pages 122–127.

Eunsol Choi, Hannah Rashkin, Luke Zettlemoyer, and Yejin Choi. 2016. Document-level sentiment inference with social, faction, and discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 333–343.

Lingjia Deng and Janyce Wiebe. 2015a. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1323–1328.

Lingjia Deng and Janyce Wiebe. 2015b. Mpqa 3.0: An entity/event-level sentiment corpus. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* pages 1323–1328.

David Dowty. 1991. Thematic proto-roles and argument selection. *language* 67(3):547–619.

Joe Ellis, Jeremy Getman, and Stephanie M Strassel. 2014. Overview of linguistic resources for the tac kbp 2014 evaluations: Planning, execution, and results. In *Proceedings of TAC KBP 2014 Workshop, National Institute of Standards and Technology*. pages 17–18.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. Association for Computational Linguistics, pages 94–99.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. 2016. Relation extraction with multi-instance multi-label convolutional neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 1471–1480.

Manfred Klenner and Michael Amsler. 2016. Sentiframes: A resource for verb-centered german sentiment inference. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 2888–2891.

Natalia Loukachevitch and Nicolay Rusnachenko. 2018. Extracting sentiment attitudes from analytical texts. *Proceedings of International Conference on Computational Linguistics and Intellectual Technologies Dialogue-2018 (arXiv:1808.08932)* pages 459–468.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pages 1003–1011.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation frames: A data-driven investigation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pages 311–321.

Nicolay Rusnachenko and Natalia Loukachevitch. 2018. Neural network approach for extracting aggregated opinions from analytical articles. In *International Conference on Data Analytics and Management in Data Intensive Domains*. Springer, pages 167–179.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 417–424.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. pages 2335–2344.

Self-Attentional Models Application in Task-Oriented Dialogue Generation Systems

Mansour Saffar Mehrjardi, Amine Trablesi, Osmar R. Zaïane
Department of Computing Science, University of Alberta
{saffarme,atrabels,zaiane}@ualberta.ca

Abstract

Self-attentional models are a new paradigm for sequence modelling tasks which differ from common sequence modelling methods, such as recurrence-based and convolution-based sequence learning, in the way that their architecture is only based on the attention mechanism. Self-attentional models have been used in the creation of the state-of-the-art models in many NLP tasks such as neural machine translation, but their usage has not been explored for the task of training end-to-end task-oriented dialogue generation systems yet. In this study, we apply these models on the three different datasets for training task-oriented chatbots. Our finding shows that self-attentional models can be exploited to create end-to-end task-oriented chatbots which not only achieve higher evaluation scores compared to recurrence-based models, but also do so more efficiently.

1 Introduction

Task-oriented chatbots are a type of dialogue generation system which tries to help the users accomplish specific tasks, such as booking a restaurant table or buying movie tickets, in a continuous and uninterrupted conversational interface and usually in as few steps as possible. The development of such systems falls into the Conversational AI domain which is the science of developing agents which are able to communicate with humans in a natural way (Ram et al., 2018). Digital assistants such as Apple's Siri, Google Assistant, Amazon Alexa, and Alibaba's AliMe are examples of successful chatbots developed by giant companies to engage with their customers.

There are mainly two different ways to create a task-oriented chatbot which are either using set of hand-crafted and carefully-designed rules or use corpus-based method in which the chatbot can be trained with a relatively large corpus of conversational data. Given the abundance of dialogue data, the latter method seems to be a better and a more general approach for developing task-oriented chatbots. The corpus-based method also falls into two main chatbot design architectures which are pipelined and end-to-end architectures (Chen et al., 2017). End-to-end chatbots are usually neural networks based (Shang et al., 2015; Dodge et al., 2015; Wen et al., 2016; Eric and Manning, 2017a) and thus can be adapted to new domains by training on relevant dialogue datasets for that specific domain. Furthermore, all sequence modelling methods can also be used in training end-to-end task-oriented chatbots. A sequence modelling method receives a sequence as input and predicts another sequence as output. For example in the case of machine translation the input could be a sequence of words in a given language and the output would be a sentence in a second language. In a dialogue system, an utterance is the input and the predicted sequence of words would be the corresponding response.

Self-attentional models are a new paradigm for sequence modelling tasks which differ from common sequence modelling methods, such as recurrence-based and convolution-based sequence learning, in the way that their architecture is only based on the attention mechanism. The Transformer (Vaswani et al., 2017) and Universal Transformer (Dehghani et al., 2018) models are the first models that entirely rely on the self-attention mechanism for both encoder and decoder, and that is why they are also referred to as a self-attentional models. The Transformer models has produced state-of-the-art results in the task neural machine

Proceedings of Recent Advances in Natural Language Processing, pages 1031–1040,
Varna, Bulgaria, Sep 2–4 2019.

translation (Vaswani et al., 2017) and this encouraged us to further investigate this model for the task of training task-oriented chatbots. While in the Transformer model there is no recurrence, it turns out that the recurrence used in RNN models is essential for some tasks in NLP including language understanding tasks and thus the Transformer fails to generalize in those tasks (Dehghani et al., 2018). We also investigate the usage of the Universal Transformer for this task to see how it compares to the Transformer model.

We focus on self-attentional sequence modelling for this study and intend to provide an answer for one specific question which is:

- How effective are self-attentional models for training end-to-end task-oriented chatbots?

Our contribution in this study is as follows:

- We train end-to-end task-oriented chatbots using both self-attentional models and common recurrence-based models used in sequence modelling tasks and compare and analyze the results using different evaluation metrics on three different datasets.

- We provide insight into how effective are self-attentional models for this task and benchmark the time performance of these models against the recurrence-based sequence modelling methods.

- We try to quantify the effectiveness of self-attention mechanism in self-attentional models and compare its effect to recurrence-based models for the task of training end-to-end task-oriented chatbots.

2 Related Work

2.1 Task-Oriented Chatbots Architectures

End-to-end architectures are among the most used architectures for research in the field of conversational AI. The advantage of using an end-to-end architecture is that one does not need to explicitly train different components for language understanding and dialogue management and then concatenate them together. Network-based end-to-end task-oriented chatbots as in (Wen et al., 2016; Bordes et al., 2016) try to model the learning task as a policy learning method in which the model learns to output a proper response given the current state of the dialogue. As discussed before, all encoder-decoder sequence modelling methods can be used for training end-to-end chatbots. Eric and Manning (2017a) use the copy mechanism augmentation on simple recurrent neural sequence modelling and achieve good results in training end-to-end task-oriented chatbots (Gu et al., 2016).

Another popular method for training chatbots is based on memory networks. Memory networks augment the neural networks with task-specific memories which the model can learn to read and write. Memory networks have been used in (Bordes et al., 2016) for training task-oriented agents in which they store dialogue context in the memory module, and then the model uses it to select a system response (also stored in the memory module) from a set of candidates. A variation of Key-value memory networks (Miller et al., 2016) has been used in (Eric and Manning, 2017b) for the training task-oriented chatbots which stores the knowledge base in the form of triplets (which is (subject,relation,object) such as (yoga,time,3pm)) in the key-value memory network and then the model tries to select the most relevant entity from the memory and create a relevant response. This approach makes the interaction with the knowledge base smoother compared to other models.

Another approach for training end-to-end task-oriented dialogue systems tries to model the task-oriented dialogue generation in a reinforcement learning approach in which the current state of the conversation is passed to some sequence learning network, and this network decides the action which the chatbot should act upon. End-to-end LSTM based model (Williams and Zweig, 2016), and the Hybrid Code Networks (Williams et al., 2017) can use both supervised and reinforcement learning approaches for training task-oriented chatbots.

2.2 Sequence Modelling Methods

Sequence modelling methods usually fall into recurrence-based, convolution-based, and self-attentional-based methods. In recurrence-based sequence modeling, the words are fed into the model in a sequential way, and the model learns the dependencies between the tokens given the context from the past (and the future in case of bidirectional Recurrent Neural Networks (RNNs)) (Goodfellow et al., 2016). RNNs and their variations such as Long Short-term Memory

(LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Units (GRU) (Cho et al., 2014) are the most widely used recurrence-based models used in sequence modelling tasks. Convolution-based sequence modelling methods rely on Convolutional Neural Networks (CNN) (LeCun et al., 1998) which are mostly used for vision tasks but can also be used for handling sequential data. In CNN-based sequence modelling, multiple CNN layers are stacked on top of each other to give the model the ability to learn long-range dependencies. The stacking of layers in CNNs for sequence modeling allows the model to grow its receptive field, or in other words context size, and thus can model complex dependencies between different sections of the input sequence (Gehring et al., 2017; Yu and Koltun, 2015). WaveNet (2016), used in audio synthesis, and ByteNet (2016), used in machine translation tasks, are examples of models trained using convolution-based sequence modelling.

3 Models

We compare the most commonly used recurrence-based models for sequence modelling and contrast them with Transformer and Universal Transformer models. The models that we train are:

3.1 LSTM and Bi-Directional LSTM

Long Short-term Memory (LSTM) networks are a special kind of RNN networks which can learn long-term dependencies (Hochreiter and Schmidhuber, 1997). RNN models suffer from the vanishing gradient problem (Bengio et al., 1994) which makes it hard for RNN models to learn long-term dependencies. The LSTM model tackles this problem by defining a gating mechanism which introduces input, output and forget gates, and the model has the ability to decide how much of the previous information it needs to keep and how much of the new information it needs to integrate and thus this mechanism helps the model keep track of long-term dependencies.

Bi-directional LSTMs (Schuster and Paliwal, 1997) are a variation of LSTMs which proved to give better results for some NLP tasks (Graves and Schmidhuber, 2005). The idea behind a Bi-directional LSTM is to give the network (while training) the ability to not only look at past tokens, like LSTM does, but to future tokens, so the model has access to information both form the past and future. In the case of a task-oriented dialogue generation systems, in some cases, the information needed so that the model learns the dependencies between the tokens, comes from the tokens that are ahead of the current index, and if the model is able to take future tokens into accounts it can learn more efficiently.

3.2 Transformer

As discussed before, Transformer is the first model that entirely relies on the self-attention mechanism for both the encoder and the decoder. The Transformer uses the self-attention mechanism to learn a representation of a sentence by relating different positions of that sentence. Like many of the sequence modelling methods, Transformer follows the encoder-decoder architecture in which the input is given to the encoder and the results of the encoder is passed to the decoder to create the output sequence. The difference between Transformer (which is a self-attentional model) and other sequence models (such as recurrence-based and convolution-based) is that the encoder and decoder architecture is only based on the self-attention mechanism. The Transformer also uses multi-head attention which intends to give the model the ability to look at different representations of the different positions of both the input (encoder self-attention), output (decoder self-attention) and also between input and output (encoder-decoder attention) (Vaswani et al., 2017). It has been used in a variety of NLP tasks such as mathematical language understanding [110], language modeling (Dai et al., 2018), machine translation (Vaswani et al., 2017), question answering (Devlin et al., 2018), and text summarization (Liu et al., 2018).

3.3 Universal Transformer

The Universal Transformer model is an encoder-decoder-based sequence-to-sequence model which applies recurrence to the representation of each of the positions of the input and output sequences. The main difference between the RNN recurrence and the Universal Transformer recurrence is that the recurrence used in the Universal Transformer is applied on consecutive representation vectors of each token in the sequence (i.e., over depth) whereas in the RNN models this recurrence is applied on positions of the tokens in the sequence. A variation of the Universal Transformer, called Adaptive Universal

Transformer, applies the Adaptive Computation Time (ACT) (Graves, 2013) technique on the Universal Transformer model which makes the model train faster since it saves computation time and also in some cases can increase the model accuracy. The ACT allows the Universal Transformer model to use different recurrence time steps for different tokens.

We know, based on reported evidence that transformers are potent in NLP tasks like translation and question answering. Our aim is to assess the applicability and effectiveness of transformers and universal-transformers in the domain of task-oriented conversational agents. In the next section, we report on experiments to investigate the usage of self-attentional models performance against the aforementioned models for the task of training end-to-end task-oriented chatbots.

4 Experiments

We run our experiments on Tesla 960M Graphical Processing Unit (GPU). We evaluated the models using the aforementioned metrics and also applied early stopping (with delta set to 0.1 for 600 training steps).

4.1 Datasets

We use three different datasets for training the models. We use the Dialogue State Tracking Competition 2 (DSTC2) dataset (Williams et al., 2013) which is the most widely used dataset for research on task-oriented chatbots. We also used two other datasets recently open-sourced by Google Research (Shah et al., 2018) which are M2M-sim-M (dataset in movie domain) and M2M-sim-R (dataset in restaurant domain)[1]. M2M stands for Machines Talking to Machines which refers to the framework with which these two datasets were created. In this framework, dialogues are created via dialogue self-play and later augmented via crowdsourcing. We trained on our models on different datasets in order to make sure the results are not corpus-biased. Table 1 shows the statistics of these three datasets which we will use to train and evaluate the models.

Dataset	Num. of Slots	Train	Dev	Test
DSTC2	8	1618	1117	500
M2M-R	9	1116	349	775
M2M-M	5	384	120	264

Table 1: Statistics of DSTC2, M2M-R, and M2M-M Datasets

The M2M dataset has more diversity in both language and dialogue flow compared to the the commonly used DSTC2 dataset which makes it appealing for the task of creating task-oriented chatbots. This is also the reason that we decided to use M2M dataset in our experiments to see how well models can handle a more diversed dataset.

4.1.1 Dataset Preparation

We followed the data preparation process used for feeding the conversation history into the encoder-decoder as in (Eric and Manning, 2017a). Consider a sample dialogue D in the corpus which consists of a number of turns exchanged between the user and the system. D can be represented as $(u_1, s_1), (u_2, s_2), ..., (u_k, s_k)$ where k is the number of turns in this dialogue. At each time step in the conversation, we encode the conversation turns up to that time step, which is the context of the dialogue so far, and the system response after that time step will be used as the target. For example, given we are processing the conversation at time step i, the context of the conversation so far would be $(u_1, s_1, u_2, s_2, ..., u_i)$ and the model has to learn to output (s_i) as the target.

4.2 Training

We used the tensor2tensor library (Vaswani et al., 2018) in our experiments for training and evaluation of sequence modeling methods. We use Adam optimizer (Kingma and Ba, 2014) for training the models. We set $\beta_1 = 0.9$, $\beta_2 = 0.997$, and $\epsilon = 1e - 9$ for the Adam optimizer and started with learning rate of 0.2 with noam learning rate decay schema (Vaswani et al., 2017). In order to avoid overfitting, we use dropout (Srivastava et al., 2014) with dropout chosen from [0.7-0.9] range. We also conducted early stopping (Goodfellow et al., 2016) to avoid overfitting in our experiments as the regularization methods. We set the batch size to 4096, hidden size to 128, and the embedding size to 128 for all the models. We also used grid search for hyperparameter tuning for all of the trained models. Details of our training and hyperparameter tuning and the code for reproducing the results can be found in the *chatbot-exp*

[1] https://github.com/google-research-datasets/simulated-dialogue

Dataset Split	Model	BLEU	Per Turn. Acc	Per Diag. Acc	Entity F1
test	LSTM (bs=1)	5.75	17.70	0.0	5.63
	LSTM + Attention (bs=2)	30.84	18.08	0.15	32.16
	Bi-LSTM (bs=2)	30.38	18.04	0.0	24.34
	Bi-LSTM + Attention (bs=2)	38.64	26.04	0.62	43.52
	Transformer (bs=2)	**51.83**	**39.02**	**1.7**	**64.20**
	UT (bs=2)	44.93	36.62	1.08	57.98
	UT + ACT (bs=2)	39.40	30.00	0.15	61.49
development	LSTM	16.13	10.33	0.0	6.54
	LSTM + Attention	31.05	18.68	0.31	32.59
	Bi-LSTM	30.92	19.07	0.31	25.91
	Bi-LSTM + Attention	39.12	27.28	**0.96**	44.15
	Transformer	**54.18**	**41.09**	0.62	**66.02**
	UT	47.95	39.01	0.31	61.27
	UT + ACT	39.27	29.30	0.31	62.50

Table 2: Evaluation of Models on DSTC2 dataset for both test and development datasets (bs: shows the best beam size in inference; UT: Universal Transformers)

github repository[2].

4.3 Inference

In the inference time, there are mainly two methods for decoding which are greedy and beam search (Freitag and Al-Onaizan, 2017). Beam search has been proved to be an essential part in generative NLP task such as neural machine translation (Wu et al., 2016). In the case of dialogue generation systems, beam search could help alleviate the problem of having many possible valid outputs which do not match with the target but are valid and sensible outputs. Consider the case in which a task-oriented chatbot, trained for a restaurant reservation task, in response to the user utterance *"Persian food"*, generates the response *"what time and day would you like the reservation for?"* but the target defined for the system is *"would you like a fancy restaurant?"*. The response generated by the chatbot is a valid response which asks the user about other possible entities but does not match with the defined target.

We try to alleviate this problem in inference time by applying the beam search technique with a different beam size $\alpha \in \{1, 2, 4\}$ and pick the best result based on the BLEU score. Note that when $\alpha = 1$, we are using the original greedy search method for the generation task.

4.4 Evaluation Measures

BLEU: We use the Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) metric which is commonly used in machine translation tasks. The BLEU metric can be used to evaluate dialogue generation models as in (Eric and Manning, 2017a; Li et al., 2015). The BLEU metric is a word-overlap metric which computes the co-occurrence of N-grams in the reference and the generated response and also applies the brevity penalty which tries to penalize far too short responses which are usually not desired in task-oriented chatbots. We compute the BLEU score using all generated responses of our systems.

Per-turn Accuracy: Per-turn accuracy measures the similarity of the system generated response versus the target response. Eric and Manning (2017a) used this metric to evaluate their systems in which they considered their response to be correct if all tokens in the system generated response matched the corresponding token in the target response. This metric is a little bit harsh, and the results may be low since all the tokens in the generated response have to be exactly in the same position as in the target response.

Per-Dialogue Accuracy: We calculate per-dialogue accuracy as used in (Bordes et al., 2016; Eric and Manning, 2017a). For this metric, we consider all the system generated responses and compare them to the target responses. A dialogue is considered to be true if all the turns in the system generated responses match the corresponding turns in the target responses. Note that this is a very strict metric in which all the utterances in the

[2] https://github.com/msaffarm/chatbot-exp

Dataset Split	Model	BLEU	Per Turn. Acc	Per Diag. Acc	Entity F1
M2M-R (test)	LSTM(bs=2)	6.00	**2.3**	0.0	7.99
	LSTM+Att.(bs=1)	7.9	1.84	0.0	16.77
	Bi-LSTM(bs=1)	8.15	1.8	0.0	19.61
	Bi-LSTM+Att.(bs=1)	8.3	0.97	0.0	24.12
	Transformer(bs=1)	**10.28**	1.76	0.0	**36.92**
	UT(bs=2)	9.15	1.88	0.0	25.44
	UT+ACT(bs=2)	8.54	1.43	0.0	23.12
M2M-M (test)	LSTM(bs=4)	7.7	**3.36**	0.0	31.07
	LSTM+Att.(bs=2)	8.3	3.27	0.0	31.18
	Bi-LSTM(bs=2)	9.6	2.09	0.0	28.09
	Bi-LSTM+Att.(bs=2)	10.62	2.54	0.0	32.43
	Transformer(bs=1)	**11.95**	2.36	0.0	**39.89**
	UT(bs=2)	10.87	3.15	0.0	34.15
	UT+ACT(bs=2)	10.48	2.46	0.0	32.76

Table 3: Evaluation of models on M2M restaurant (M2M-R) and movie (M2M-M) dataset for test datasets (bs: The best beam size in inference; UT: Universal Transformers)

dialogue should be the same as the target and in the right order.

F1-Entity Score: Datasets used in task-oriented chores have a set of entities which represent user preferences. For example, in the restaurant domain chatbots common entities are meal, restaurant name, date, time and the number of people (these are usually the required entities which are crucial for making reservations, but there could be optional entities such as location or rating). Each target response has a set of entities which the system asks or informs the user about. Our models have to be able to discern these specific entities and inject them into the generated response. To evaluate our models we could use named-entity recognition evaluation metrics (Jiang et al., 2016). The F1 score is the most commonly used metric used for the evaluation of named-entity recognition models which is the harmonic average of precision and recall of the model. We calculate this metric by micro-averaging over all the system generated responses.

5 Results and Discussion

5.1 Comparison of Models

The results of running the experiments for the aforementioned models is shown in Table 2 for the DSTC2 dataset and in Table 3 for the M2M datasets. The bold numbers show the best performing model in each of the evaluation metrics. As discussed before, for each model we use different beam sizes (bs) in inference time and report the best one. Our findings in Table 2 show that self-attentional models outperform common recurrence-based sequence modelling methods in the BLEU, Per-turn accuracy, and entity F1 score. The reduction in the evalution numbers for the M2M dataset and in our investigation of the trained model we found that this considerable reduction is due to the fact that the diversity of M2M dataset is considerably more compared to DSTC2 dataset while the traning corpus size is smaller.

5.2 Time Performance Comparison

Table 4 shows the time performance of the models trained on DSTC2 dataset. Note that in order to get a fair time performance comparison, we trained the models with the same batch size (4096) and on the same GPU. These numbers are for the best performing model (in terms of evaluation loss and selected using the early stopping method) for each of the sequence modelling methods. Time to Convergence (T2C) shows the approximate time that the model was trained to converge. We also show the loss in the development set for that specific checkpoint.

5.3 Effect of (Self-)Attention Mechanism

As discussed before in Section 3.2, self-attentional models rely on the self-attention mechanism for sequence modelling. Recurrence-based models such as LSTM and Bi-LSTM can also be augmented in order to increase their performance, as evident in Table 2 which shows the increase in the performance of both LSTM and Bi-LSTM

Model	T2C (sec)	Dev Loss
LSTM	1100	0.89
LSTM+Att	1305	0.62
Bi-LSTM	1865	0.60
Bi-LSTM+Att	2120	0.49
Transformer	**612**	**0.31**
UT	1939	0.36
UT+ACT	665	0.33

Table 4: Comparison of convergence performance of the models

when augmented with an attention mechanism. This leads to the question whether we can increase the performance of recurrence-based models by adding multiple attention heads, similar to the multi-head self-attention mechanism used in self-attentional models, and outperform the self-attentional models.

To investigate this question, we ran a number of experiments in which we added multiple attention heads on top of Bi-LSTM model and also tried a different number of self-attention heads in self-attentional models in order to compare their performance for this specific task. Table 6 shows the results of these experiments. Note that the models in Table 6 are actually the best models that we found in our experiments on DSTC2 dataset and we only changed one parameter for each of them, i.e. the number of attention heads in the recurrence-based models and the number of self-attention heads in the self-attentional models, keeping all other parameters unchanged. We also report the results of models with beam size of 2 in inference time. We increased the number of attention heads in the Bi-LSTM model up to 64 heads to see its performance change. Note that increasing the number of attention heads makes the training time intractable and time consuming while the model size would increase significantly as shown in Table 5. Furthermore, by observing the results of the Bi-LSTM+Att model in Table 6 (both test and development set) we can see that Bi-LSTM performance decreases and thus there is no need to increase the attention heads further.

Our findings in Table 6 show that the self-attention mechanism can outperform recurrence-based models even if the recurrence-based models have multiple attention heads. The Bi-LSTM model with 64 attention heads cannot beat the best Trasnformer model with NH=4 and also its results are very close to the Transformer model with NH=1. This observation clearly depicts the power of self-attentional based models and demonstrates that the attention mechanism used in self-attentional models as the backbone for learning, outperforms recurrence-based models even if they are augmented with multiple attention heads.

Model	T2C (sec)	Dev Loss
Bi-LSTM+Att.[NH=1]	2120	0.49
Bi-LSTM+Att.[NH=4]	3098	0.47
Bi-LSTM+Att.[NH=8]	3530	0.44
Bi-LSTM+Att.[NH=16]	3856	0.44
Bi-LSTM+Att.[NH=32]	7320	0.36
Bi-LSTM+Att.[NH=64]	**9874**	**0.38**
Transformer[NH=1]	**375**	**0.33**
Transformer[NH=4]	**612**	**0.31**
Transformer[NH=8]	476	0.31

Table 5: Comparison of convergence performance of the models

6 Conclusion and Future Work

We have determined that Transformers and Universal-Transformers are indeed effective at generating appropriate responses in task-oriented chatbot systems. In actuality, their performance is even better than the typically used deep learning architectures. Our findings in Table 2 show that self-attentional models outperform common recurrence-based sequence modelling methods in the BLEU, Per-turn accuracy, and entity F1 score. The results of the Transformer model beats all other models in all of the evaluation metrics. Also, comparing the result of LSTM and LSTM with attention mechanism as well as the Bi-LSTM with Bi-LSTM with attention mechanism, it can be observed in the results that adding the attention mechanism can increase the performance of the models. Comparing the results of self-attentional models shows that the Transformer model outperforms the other self-attentional models, while the Universal Transformer model gives reasonably good results.

In future work, it would be interesting to compare the performance of self-attentional models (specifically the winning Transformer model) against other end-to-end architectures such as the Memory Augmented Networks.

Dataset Split	Model	BLEU	Per-Turn Acc	Per-Diag Acc	Entity F1
test	Bi-LSTM+Att.[NH=1]	38.64	26.04	0.62	43.52
	Bi-LSTM+Att.[NH=4]	42.23	29.01	0.92	48.06
	Bi-LSTM+Att.[NH=8]	42.61	28.18	0.77	49.90
	Bi-LSTM+Att.[NH=16]	**43.11**	**30.34**	**0.61**	**50.87**
	Bi-LSTM+Att.[NH=32]	**48.62**	**36.46**	**1.85**	**59.8**
	Bi-LSTM+Att.[NH=64]	**47.33**	**33.17**	**1.23**	**56.49**
	Transformer[NH=1]	**45.90**	**36.64**	**1.7**	**57.55**
	Transformer[NH=4]	**51.83**	**39.02**	**1.7**	**64.20**
	Transformer[NH=8]	51.37	39.45	3.24	62.38
	UT[NH=1]	43.02	31.20	1.54	60.10
	UT[NH=8]	48.17	35.76	2.93	61.56
	UT+ACT[NH=1]	34.98	25.66	0.46	51.32
	UT+ACT[NH=8]	36.29	24.97	0.31	55.27
development	Bi-LSTM+Att.[NH=1]	39.12	27.28	0.96	44.15
	Bi-LSTM+Att.[NH=4]	40.47	27.64	0.93	48.10
	Bi-LSTM+Att. [NH=8]	42.78	28.36	0.31	50.05
	Bi-LSTM+Att.[NH=16]	**42.88**	**30.36**	**0.93**	**52.09**
	Bi-LSTM+Att.[NH=32]	**49.36**	**38.24**	**0.61**	**61.26**
	Bi-LSTM+Att.[NH=64]	**47.28**	**33.12**	**0.93**	**56.86**
	Transformer[NH=1]	**47.86**	**38.33**	**1.85**	**60.37**
	Transformer[NH=4]	**54.18**	**41.09**	**0.62**	**66.02**
	Transformer[NH=8]	51.54	39.42	1.54	63.56
	UT[NH=1]	43.01	32.12	1.58	60.42
	UT[NH=8]	47.89	35.57	1.23	61.33
	UT+ACT[NH=1]	35.74	26.46	0.31	52.71
	UT+ACT[NH=8]	38.95	27.10	0.31	57.02

Table 6: Evaluation of effect of self-attention mechanism using DSTC2 dataset (Att: Attetnion mechanism; UT: Universal Transformers; ACT: Adaptive Computation Time; NH: Number of attention heads)

References

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter* 19(2):25–35.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2018. Transformer-xl: Language modeling with longer-term dependency .

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* .

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931* .

Mihail Eric and Christopher D Manning. 2017a. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024* .

Mihail Eric and Christopher D Manning. 2017b. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414* .

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806* .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pages 1243–1252.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Ridong Jiang, Rafael E Banchs, and Haizhou Li. 2016. Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*. pages 21–27.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198* .

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604* .

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *SSW* 125.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416* .

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. pages 5998–6008.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* .

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* .

Whom to Learn From?
Graph- vs. Text-based Word Embeddings

Małgorzata Salawa[1,2], **António Branco**[1], **Ruben Branco**[1], **João Rodrigues**[1] and **Chakaveh Saedi**[1,3]

[1]*University of Lisbon*
NLX-Natural Language and Speech Group, Faculdade de Ciências, 1749-016 Lisboa, Portugal
[2]*AGH University of Science and Technology*
Faculty of Computer Science, Electronics and Telecommunications, 30-001 Kraków, Poland
[3]*Macquarie University*
Department of Computing, Sydney, NSW, 2109, Australia

Abstract

Vectorial representations of meaning can be supported by empirical data from diverse sources and obtained with diverse embedding approaches. This paper aims at screening this experimental space and reports on an assessment of word embeddings supported (i) by data in raw texts vs. in lexical graphs, (ii) by lexical information encoded in association- vs. inference-based graphs, and obtained (iii) by edge reconstruction- vs. matrix factorisation vs. random walk-based graph embedding methods. The results observed with these experiments indicate that the best solutions with graph-based word embeddings are very competitive, consistently outperforming mainstream text-based ones.

1 Introduction

As neural networks are becoming a central technology in natural language processing, interest on distributional semantics, with its vector space models of meaning, has been a driving factor for research on natural language semantics. When focusing on the meaning of words under this approach, information on lexical semantics has been sought to be encoded into appropriate vectorial representations, also known as word embeddings. The source for this information has consisted mostly of large collections of raw text, and thus ultimately on the frequencies of co-occurrence of words with other neighbouring words in certain windows of context, (Mikolov et al., 2013a; Pennington et al., 2014; Mikolov et al., 2018) among others. A few research trends have been gaining momentum concerning the application of neural networks to natural language technology, and *a fortiori* in what concerns distributional semantics. On the one hand, there has been a growing interest in the linguistic information that may be ultimately encoded in vectorial representations (Belinkov et al., 2017; Conneau et al., 2018), also relating to their eventual ”universality”, in view of possibly transferring these representations from one language processing task or application to another (Shi et al., 2016; Cífka and Bojar, 2018).

On the other hand, growing attention has been devoted to sources of information for word embeddings other than what may be conveyed and extracted from co-occurrences in text. This includes information that is encoded in sophisticated lexical collections of data that are carefully crafted and densely loaded with accurate information on lexical semantics (Goikoetxea et al., 2015; Saedi et al., 2018).

The results reported in the present paper lies at the intersection of those research goals. In particular, we aim here to gain a better insight into these two sources of lexical information, and the quality of the resulting word embeddings, by assessing how graph-based word embeddings compare to mainstream text-based ones. To pursue this objective, we explore an experimental space that takes into account lexical semantic networks of essentially different types as well as different sorts of methods, with different strengths, to convert those graphs into embeddings. In the experimental space that will be explored here, text-based embeddings will be represented by top performing solutions from the literature.

In the next Sections 2 and 3, the lexical graphs and the graph embeddings techniques used are introduced. Each one of the following Sections 4 and 5 will indicate how each graph was handled and what was the outcome of applying graph embedding techniques to them.

Section 6 is devoted to ponder on the lessons that can be learned from the results obtained. Finally, in the last two Sections 7 and 8 the related work will be taken into account and the conclusions of this paper will be presented.

Proceedings of Recent Advances in Natural Language Processing, pages 1041–1051,
Varna, Bulgaria, Sep 2–4 2019.

2 Lexical graphs

How to represent the meaning of words has been at the core of research on lexical semantics. Besides distributional semantics (Harris, 1954; Osgood et al., 1957) that word embeddings adhere to, two other broad families of approaches have emerged, namely those advocating that lexical semantics is better represented as a semantic network (Quillan, 1966) or as a feature-based model (Minsky, 1975; Bobrow and Norman, 1975).

In a nutshell, in an inference-based semantic network, a lexical unit, typically a word, is recorded as a node in a graph while the semantic relations among words, such as hyponymy or synonymy, etc., are recorded as labelled edges among the nodes of the graph — with the inference being ensured by the relation that happen to be transitive. Feature-based models representing lexical semantics, in turn, resort to a hash table that stores the lexical units as keys, and the semantically related units as the respective values.

The motivation for these two families of lexical representation is to be found in their different suitability and success in explaining a wide range of empirical phenomena, in terms of how these are manifest in ordinary language usage and how they are elicited in laboratory experimentation. These phenomena are related to the acquisition, storage and retrieval of lexical knowledge (e.g. the spread activation effect (Meyer and Schvaneveldt, 1971), the fan effect (Anderson, 1974), among many others) and to how this knowledge interacts with other cognitive faculties or tasks, including categorization (Estes, 1994), reasoning (Rips, 1975), problem solving (Holyoak and Koh, 1987), learning (Ross, 1984), etc. Feature-based models seek to respond primarily to our outstanding ability as speakers of associating concepts with other concepts, while inference-based ones seek to respond to the also outstanding ability to reason on the basis of semantic relations among concepts.

In the scope of the formal and computational modelling of lexical semantics, these approaches have inspired a number of initiatives to build repositories of lexical knowledge. Prominent examples of such repositories are, for semantic networks, WordNet (Fellbaum, 1998) and for feature-based models, Small World of Words (SWOW) (De Deyne et al., 2013). Interestingly, to achieve the highest quality, repositories of different types typically resort to different empirical sources of primary data. For instance, WordNet is constructed on the basis of lexical intuitions systematically handled by human experts, while the information encoded in Small World of Words are the associations between concepts evoked and collected from laypersons.

Even when motivated in the first place by (psycho-)linguistic research goals, these repositories of lexical knowledge have been extraordinarily important for language technology. They have been instrumental for major advances in language processing tasks and applications such as word sense disambiguation, part-of-speech tagging, named entity recognition, sentiment analysis (e.g. Li and Jurafsky (2015)), parsing (e.g. Socher et al. (2013)), textual entailment (e.g. Baroni et al. (2012)), discourse analysis (e.g. Ji and Eisenstein (2014)), among many others.[1]

In our experiments, we resort to these two major representatives of inference- and feature-based lexical networks, namely WordNet[2] and SWOW[3].

3 Graph embedding methods

As for methods to convert graphs into embedding, we are resorting also to one outstanding representative per major family of techniques.

Following the recent comprehensive survey by (Cai et al., 2017), graph embeddings methods divide into those that represent a whole graph as a single vector and those that output a vector for each node in the graph. For our experiments, we are interested in the latter, for which there are three major families of approaches, viz. based on edge reconstruction, on matrix factorisation and on random walks. Each of these techniques has its advantages and drawbacks, capturing the information encoded in the graph with different emphasis.

Graph embeddings techniques based on edge reconstruction operate on graphs represented by edge lists. An edge is a triple $\langle lhs, rel, rhs \rangle$, where lhs (left-hand side) and rhs (right-hand) are nodes connected by a relation of type rel. The system is trained to recognise triples that are feasible (present in the graph) from the infeasible ones.

The objective function optimised in the model

[1] For the vast number of applications of WordNet, see http://lit.csci.unt.edu/~wordnet

[2] Princeton's WordNet 3.0 is the version used here, obtained from http://wordnet.princeton.edu/ in February 2019.

[3] From http://github.com/SimonDeDeyne/SWOWEN-2018 in March 2019.

is either maximising the edge reconstruction probability or minimising the edge reconstruction loss. The latter can be further divided into distance-based loss and margin-based ranking loss. Since most of the existing knowledge graph embedding methods choose to optimise margin based ranking loss (Cai et al., 2017), we choose a method from this subgroup as a representative of the edge reconstruction models, namely Semantic Matching Energy (SME) from (Bordes et al., 2014).

Edge reconstruction methods support a relatively efficient training, but ensures optimisation using only local information between nodes close to each other.

Another family of graph embedding methods is based, in turn, on graphs represented by matrices. This is perhaps the family of techniques with the largest number of instances, which in many cases result from slight variants from one another in the tricks used to weight and condense the nodes in the matrix.

As a representative of the matrix factorisation methods for graph embedding, we use the so-called Katz index (Newman, 2010, Eq. (7.63)) as this is the technique used in previous works on WordNet (Saedi et al., 2018) and on SWOW (De Deyne et al., 2018).

This method starts by creating a matrix with all of the possible semantic relations between all the words, resulting in an adjacency matrix M. Then it populates each cell M_{ij} of the matrix resorting to a lexical semantic graph G. Each cell M_{ij} is set to 1 if and only if there is a direct edge between nodes including the two words $word_i$ and $word_j$ the cell represents. If there is no edge between the two words, that cell is set to 0. For all nodes not directly connected, that is connected through other nodes in between, the representation of their affinity strength is obtained by following the cumulative iteration:

$$M_G^n = I + \alpha M + \alpha^2 M^2 + \cdots + \alpha^n M^n \quad (1)$$

M^n is the matrix where every two words, $word_i$ and $word_j$, are transitively related by n edges. I represents the identity matrix and α is used as a decay factor for longer paths.

The iteration converges into the matrix M_G, obtained by an inverse matrix operation:

$$M_G = \sum_{e=0}^{\infty} (\alpha M)^e = (I - \alpha M)^{-1} \quad (2)$$

Matrix factorisation inverts the trade off found with edge reconstruction methods. Differently from the latter, it is able to take into account the affinity between nodes at the global level of the graph, but at the cost of a large time and space consumption though.

A third family of graph embedding methods is based on a ”text” generated from graphs, where the word embeddings are obtained from some deep learning technique used over that text. This is an ”artifical” text that results from concatenating the words in the nodes that are visited in a random walk through the edges in the graph.

Starting at a random node in the graph, at each iteration, this technique randomly chooses a neighbour node (with a probability α) to be the starting point of the next iteration or stopping the walk (with a probability 1 - α) .

Improving over the matrix factorisation and the edge reconstruction approaches, the random walk technique is effective and accommodates global information on the nodes. However, as it only considers the local context within a path at each iteration, that makes it hard to find an optimal sampling strategy.

In the next Sections, we report on the application of these three different graph embedding techniques, with their different advantages and drawbacks, over the two lexical networks, from two distinct lexical semantic families, thus encoding lexical knowledge from quite distinct primary sources of empirical data. This leads to different word embeddings that encode and emphasise different shades of lexical information, thus contributing to an encompassing and discriminating experimental space.

4 Inference-based graph embeddings

This section describes the conversion of WordNet to word embeddings under each of the three graph embedding techniques.

4.1 Edge reconstruction

In models based on edge reconstruction, the objective is to rank a true triplet $\langle lhs, rel, rhs \rangle$ over a false triplet $\langle lhs', rel, rhs' \rangle$ that does not exist in the graph. Under the SME technique (Bordes et al., 2014) we are following here, this is achieved by designing an energy function $f_{rel}(lhs, rhs)$, interpreted as a distance between the nodes lhs and rhs in the context of relation rel, where the energy value should be lower for feasible triplets and higher for infeasible ones. SME seeks to minimise the margin-based ranking loss, defined as:

$$O_{rank} = \min \sum_{\substack{\langle lhs,rel,rhs\rangle \in S \\ \langle lhs',rel,rhs'\rangle \notin S}} \max\Big(0, \gamma + f_{rel}(lhs, rhs) - f_{rel}(lhs', rhs')\Big) \tag{3}$$

where γ is the margin size (set by default to 1).

The SME function is designed as a neural network that first combines the nodes separately with the relation type, putting the combinations of $\langle lhs, rel\rangle$ and $\langle rel, rhs\rangle$ in a common space, where they can be *matched*. The matching is performed using a dot product of the resulting vectors. The combination function comes in two flavours: *linear* and *bilinear*. We opted for the former here given its lower complexity.

The triples were generated in the following manner: for each word w_{lhs} in the vocabulary and for each synset s_{lhs} this word belongs to, a triple is generated for each word w_{rhs} in each synset s_{rhs} (that w_{rhs} belongs to), such that there exists a relation rel between synsets s_{lhs} and s_{rhs}, and both w_{lhs} and w_{rhs} are in the vocabulary.

rel is one of the semantic relations used in WordNet.[4] Three of these relation types, namely *antonym, derivationally related form*, and *pertainym* exist not between synsets, but directly between the word forms (lemmas). These were also taken into account to generate triples.[5]

For training, we used a publicly available implementation of SME.[6] The models were trained for 500 epochs, with evaluation at every 10 epochs, a learning rate of 0.01 and 200 batches. The remaining parameters were left the same as the default ones used in Bordes et al. (2014). The model with the best performance on the validation set was picked.

Since the the edge reconstruction based methods are retaining the local neighbourhood only, we experimented also with extending the data sets by generating relations resulting from the concatenation of two simple relations. The data sets created in this way, however, suffer from an exponential growth in size. Due to resource limitations, we managed to conduct the experiments on a 15k vocabulary only, which gave significant boost in the performance of the model on the evaluation tasks. Further exploration of this path could be beneficial, but needs to be left for future work.

For a fair comparison with other methods, the data used for training the models is based though on the same 60k vocabulary as in the matrix factorisation based method (see details in Section 4.2), and thus eventually restricted to 1-hop relations. The vocabulary was selected with the same procedure as in Saedi et al. (2018). Also for the sake of comparison with the other experiments with text-based embeddings available from the literature (see details in Section 6), we chose vectors of dimension 300. Since there is a random element in the system (the initialisation of the neural network), we trained three models using different seeds for the random number generator and averaged the results.

4.2 Matrix factorisation

For matrix factorisation, we started by building an adjacency matrix from WordNet 3.0, which produced a square matrix of a size above 155k.

Tests with different weights for each type of relation — namely hyponymy and hyperonymy weighing the most, — showed that symmetrical weights performed the best. Also the parameters in equation 2 and other options to tackle computational complexity were empirically determined in and taken from Saedi et al. (2018).

The matrix inversion raises substantial challenges in terms of the memory footprint. To cope with this issue, we resorted to sub-graphs of WordNet of manageable size, and we will be using here a vocabulary with 60k words. To mitigate the impact of this downsizing, we sorted the words by the decreasing number of outgoing edges in the graph and picked the 60k top ones.

Another parameter to consider is the decay value (α) in equation 2, which discounts the strength of a connection if the nodes are far away from each other in the graph. Several values for α were experimented with, with 0.75 performing the best, which is also the value for α we used here.

After going through the procedure in equation 2, a Positive Point-wise Mutual Information transformation (PMI+) was applied to reduce the frequency bias, followed by an L2-norm to normalise each line of M_G, and finally, a Principal Compo-

[4] For a list of relation types, see http://wordnet.princeton.edu/documentation/wninput5wn.

[5] To extract the data from the WordNet 3.0 files, we used the NLTK library, available at www.nltk.org/_modules/nltk/corpus/reader/wordnet.html.

[6] http://github.com/glorotxa/SME.

nent Analysis (PCA) was applied to reduce the dimension of the vectors.

This procedure was evaluated with different vector sizes by Saedi et al. (2018), namely 100, 300, 850, 1000 and 3000, with 850 performing the best. For the sake of comparability with the other models we resort to, namely the text-based ones, we set a vector size of 300 for the matrix factorisation embedding technique.

4.3 Random walk

The random walk was based on UKB (Agirre and Soroa, 2009; Agirre et al., 2014; Goikoetxea et al., 2015), which performs a random walk through edges on graphs and in each step writes a word in the node into an artificial text. With the resulting corpus, a two-layer neural network model (Skip-Gram) (Mikolov et al., 2013b) was trained to predict for each vocabulary word its neighbouring words, thus generating in one of the layers the resulting word embedding vectors.

We restricted the original technique to use only the information from the graph and to ignore the glosses. The random walk was applied to the same WordNet graph (60k vocabulary) described in the Sections 4.1 and 4.2.

We discarded the three lemma-lemma relations not supported by UKB, namely antonym, derivationally related form, pertainym.

To create the artificial corpus, we used the default UKB random walk parameters[7] and to obtain the word embeddings, we used the default Gensim's (Řehůřek and Sojka, 2010) Skip-Gram implementation, with a vector dimension of 300.

4.4 Results

The assessment of the word embeddings obtained from the conversion of lexical graphs use the same tasks used for this purpose when the embeddings are obtained from corpora. These tasks consist in predicting the semantic similarity and the semantic relatedness between words in pairs and in seeking to match the gold scores assigned by humans to those test pairs. The cosine between the vectors of the words in a pair is mapped into the scale used for the gold scores.

For semantic similarity, we resorted to the test sets SimLex-999 (with 999 pairs) (Hill et al., 2016), WordSim-353-Similarity (203 pairs) (Agirre et al., 2009) and RG1965 (65) (Rubenstein and Goodenough, 1965). For semantic relatedness, WordSim-353-Relatedness (252) (Agirre et al., 2009), MEN (3000) (Bruni et al., 2012) and MTURK-771 (771) (Halawi et al., 2012) were used.

The results with WordNet embeddings are displayed in Table 1.[8]

	Edge	Factor.	Walk
Similarity			
Simlex-999	39.63±1.55	49.90	**50.93**±0.15
WordSim-353	54.93±2.31	50.80	**67.40**±0.30
RG1965	57.70±4.84	57.00	**77.50**±0.95
Relatedness			
WordSim-353	26.20±4.10	**30.90**	28.43±0.76
MEN	39.67±2.55	45.00	**52.17**±0.70
MTurk-771	42.40±1.25	52.80	**52.90**±0.50

Table 1: Performance of WordNet embeddings (columns) over test sets (rows) in terms of Spearman's Correlation Coefficient (higher is better), with deviation from averaging over three runs indicated where relevant. Bold denotes best results.

5 Feature-based graph embeddings

This section describes the conversion of SWOW to word embeddings under each of the three graph embedding techniques.

5.1 Edge reconstruction

The data for the application of the SME method was generated on the basis of the associative strength among words, described in detail in De Deyne et al. (2018). The vocabulary was restricted to their 12 216 cue words.

The relations were generated with the support of the associative strength files that were generated by using the publicly available implementation.[9] The strength file is generated for three association types separately (*R1, R2, R3*), which induced the three relation types taken into by the SME method with SWOW.

We used the same implementation and methodology as in Section 4.1. We empirically chose a smaller interval between the evaluations (every 5

[7] http://github.com/asoroa/ukb/

[8] The coverage of the test sets is the following: 100% of Simlex-999; 100% WordSim-353 S; 98.0% RG1965; 97.6% WordSim-353 R; 83.4% MEN; 99.9% MTurk-771.

[9] http://github.com/SimonDeDeyne/SWOWEN-2018

epochs instead of 10) and a lower learning rate (0.001 instead of 0.01) for a better training quality. The validation and test sets each made up for around 5% of the data set. For the sake of comparison, we again chose the vector size of 300.

Similarly, as in Section 4.1, we trained three models and average the results.

5.2 Matrix factorisation

We follow the same methodology, data and implementation in De Deyne et al. (2018). The data set contained 12 216 cue words, a shorter vocabulary and matrix than the one selected from WordNet.

The data is pre-processed before generating the adjacency matrix, where the cue words and responses are spell-checked, and the adjustment of capitalisation and americanisms takes place. From the cue-response data, only 100 participants for each cue are considered. Since each participant responded with three associated tokens, this associates each cue with 300 word instances.

The adjacency matrix was then created similarly to the matrix factorisation of WordNet in section 4.2, yielding a square matrix A_G, with every word displayed in the rows and in the columns. The cell A_{Gij} contains the associative strength of word i with word j, obtained from the frequency with which word j is responded when word i is cued.

The adjacency matrix is factorised using the same parameters as described in Section 4.2, namely with the decay factor α set at 0.75, and with a vector dimension of 300. Due to the small, 12k vocabulary available here, no extraction of a subset was necessary as it formed a data set computationally manageable.

The processing of the output matrix is also the same as in section 4.2, with an application of PMI+ to reduce frequency bias and PCA for dimension reduction.

5.3 Random walk

The random walk used the same technique as used for the inference-based graph, in Section 4.3.

The SWOW data set described in the previous Sections 5.1 and 5.2 was converted into a graph input for UKB. Each word in the vocabulary was considered a node. Each relation from a SWOW cue word to the associated word was considered as a relation between nodes. With the resulting graph, we created the artificial corpus by using the default UKB random walk parameters. To obtain the word embeddings, we used the default Gensim's Skip-Gram implementation (Řehůřek and Sojka, 2010) with vectors of dimension 300.

5.4 Results

The results with SWOW embeddings are displayed in Table 2.[10]

	Edge	Factor.	Walk
Similarity			
Simlex-999	54.13±6.20	67.80	**69.33**±0.06
WordSim-353	77.07±4.76	**85.00**	84.53±0.06
RG1965	83.50±4.50	**92.90**	90.23±0.49
Relatedness			
WordSim-353	70.70±3.68	**79.30**	77.73±0.23
MEN	78.50±3.90	**87.20**	84.27±0.06
MTurk-771	74.77±4.21	80.90	**81.10**±0.17

Table 2: Performance of SWOW embeddings (columns) over test sets (rows) in terms of Spearman's Correlation Coefficient (higher is better).

6 Discussion

The results in the Sections above were obtained with word embeddings whose source of information are specifically designed and carefully curated lexical collections whose primary empirical source of data are human lexical intuitions elicited and gathered under a tightly controlled experimental protocol. This range of results should be enlarged with results obtained also with word embeddings that have, as the source of lexical information, collections of raw texts that were produced with purposes other than to serve specifically for word embeddings.

6.1 Text-based embeddings

For this purpose, we resort to mainstream text-based word embeddings. For a fair comparison, we focus on embeddings that rely solely on lexical information, thus not possibly enhanced with supra-lexical information, like for instance Dependency word embeddings (Levy and Goldberg, 2014), etc. The three word embeddings selected are Glove (Pennington et al., 2014),[11] word2vec

[10] The coverage of the test sets is the following: 99.6% of Simlex-999; 83.1% WordSim-353 S; 90.6% RG1965; 87.3% WordSim-353 R; 89.4% MEN; 93.2% MTurk-771.

[11] These embeddings have Vectors of dimension 300 trained over 840B Token text. They were obtained from

(Mikolov et al., 2013a),[12] and fastText (Mikolov et al., 2018).[13]

The evaluation results for these text-based word embeddings are displayed in Table 3.

	Glove	word2vec	fastText
Similarity			
Simlex-999	37.52	43.61	**49.24**
WordSim-353	62.98	74.08	**79.74**
RG1965	65.77	74.77	**81.31**
Relatedness			
WordSim-353	57.09	60.97	**71.33**
MEN	67.65	69.89	**80.87**
MTurk-771	63.07	65.69	**76.13**

Table 3: Performance of text-based embeddings (columns) over test sets (rows) in terms of Spearman's Correlation Coefficient (higher is better).

The word embeddings trained with a 600B token collection of texts, fastText, outperforms the other ones trained with 100B (word2vec) and 840B (Glove) token collections.

6.2 Analysis

The experimental space explored gave rise to the range of results displayed in Tables 1 to 3. We discuss in turn the observed impact of different graph embedding techniques, different lexical graphs, and different sources of lexical information.

The edge reconstruction technique consistently delivers the worst results across all lexical graphs and test sets. The top position, in turn, is shared by the random walk and matrix factorisation methods. While the former originates the best results with WordNet for most test sets, the latter does so with SWOW.

A possible explanation for this contrast may lie in that the systematic and exhaustive structuring of WordNet with regards the semantic knowledge pertaining to a given node of the graph may mitigate (more than SWOW does) the known drawback of the random walk in terms of not ensuring an optimal sampling strategy.[14]

In the reverse direction, a factor that may be favouring matrix factorisation with SWOW may lie in that the systematic coverage of all the paths within the graph ensured by that technique may mitigate (more than random walk does) the less systematic nature of the lexical knowledge encoded in an association-based graph, like SWOW.

In what concerns comparison among lexical graphs, in turn, SWOW stands out as supporting results consistently far better for every test set than the ones supported by WordNet, with a range of deltas that go from 20% (15 points) with RG1965, to 159% (48 points) with WordSim-353 Relatedness. It is also interesting to note that the largest deltas are observed with data sets that test semantic relatedness, with deltas from 53% (28 points) with MTurk-771 to 159% (48 points) with WordSim-353, than with data sets for semantic similarity, with deltas from 20% (15 points) with RG1965, to 36% (18 points) with Simlex-999. This seems to indicate that the lexical knowledge necessary to solve the semantic tasks embodied in these test sets is better encoded in SWOW than in (a subset of) WordNet.

We look now into the impact of different sources of empirical data that inform word embeddings. While the best scores of text-based consistently outperform the best scores of WordNet embeddings, they are though consistently outperformed by the best scores of SWOW embeddings, with a range of deltas that go from 7% (5 points) with MTurk-771, to 41% (20 points) with Simlex-999. It is also interesting to note that the largest deltas are observed this time with data sets that test similarity, with deltas from 7% (5 points) with WordSim-353 Similarity to 41% (20 points) with Simlex-999, than with data sets for relatedness, with deltas from 8% (6 points) with MEN, to 11% (8 points) with WordSim-353 Relatedness.

As usual, this type of results needs to be taken with a prudent grain of salt. The kind of individual scores registered above depend on the size of the supporting data sets, be they graph- of text-based embeddings, and are expected to improve as the data sets get larger. Nevertheless, the patterns ob-

http://nlp.stanford.edu/projects/glove/ in February 2019.

[12] These embeddings have Vectors of dimension 300 trained over 100B Token text. They were obtained from http://code.google.com/archive/p/word2vec/ on 22/04/2019.

[13] These embeddings have Vectors of dimension 300 trained over 600B Token text. They were obtained from http://fasttext.cc/docs/en/english-vectors.html on 22/04/2019.

[14] It is of note that the random walk graph embedding technique is not limited by the excessive memory footprint of the matrix factorisation method, and it is thus probably even better suited to take advantage of the full information and strength of WordNet.

served with this experimental space seems to provide a clear indication that graph-based embeddings are very competitive, with the best scoring solutions consistently outperforming mainstream text-based ones by a substantial margin.

It is of note that this is obtained with data sets of a much smaller size (12k) than the ones used for text-based embeddings (600B) — whose collection can be obtained with quite affordable costs in the case of SWOW, the graph that is informing the top-performing embeddings.

7 Related Work

There have been some publications pioneering the issue of obtaining word embeddings from lexical semantic networks. Each has focused though on a particular graph embedding technique or in a particular lexical graph, and thus a systematic study of graph embeddings under comparable settings was not undertaken, and *a fortiori* a comparative assessment of their strengths with regards text-based ones is also lacking.

The application of Katz index for matrix factorisation was undertaken by De Deyne et al. (2016) over SWOW and by Saedi et al. (2018) over a WordNet subset. These are the results from previous works that we follow more closely here.

The graph embedding SME technique based on edge reconstruction was pioneered by Bordes et al. (2014), who applied it to a small WordNet subset restricted to 1-hop relations, which we expanded in the experiments reported here.

The random walk methods for graph embeddings were experimented with by Goikoetxea et al. (2015) over full WordNet. This however does not represent a ”purely” graph-based approach given the raw text in the glosses was also used. In our implementation here, the embeddings were based solely on the information in the graph.

In this connection, it is worthy of note the work by Hughes and Ramage (2007), which resorts also to random graph walks over WordNet. Differently, from the goal here, its goal was to obtain word-specific stationary probability distributions — such that the semantic affinity of two words is based on the similarity of their probability distributions —, rather than to obtain vectorial representations for words.

It is also worth mentioning that the task of determining the semantic similarity between two words can be performed not only on the basis of the distance of their respective vectors in a semantic space, but also on the basis of the distance of the respective concepts in the semantic network itself. There has been a research tradition on this issue whose major proposals include (Jiang and Conrath, 1997; Lin, 1998; Leacock and Chodorow, 1998; Hirst and St-Onge, 1998; Resnik, 1999) a.o., which received nice comparative assessments in (Ferlez and Gams, 2004) and (Budanitsky and Hirst, 2006). The focus of the present paper, though, is rather on vectorial representations and semantic distances based on them.

8 Conclusions

This paper reports on the insights gained on word embeddings with an experimental space that systemically explored empirical data from radically different sources (raw texts vs. lexical graphs), lexical information encoded in graphs from essentially different paradigms of lexical semantics (association- vs. inference-based), and methods to obtain vectorial representations of the nodes in graphs from each major family of graph embedding techniques (edge reconstruction- vs. matrix factorisation vs. random walk-based). Following mainstream practice, the resulting embeddings were evaluated for semantic similarity and relatedness prediction tasks.[15]

The results obtained permit to observe a clear pattern indicating that the best scoring solutions with graph embeddings are very competitive, consistently outperforming mainstream text-based ones by a substantial margin. They indicate also that the graphs that are informing the top-performing word embeddings are of a type that can be obtained with quite affordable costs, as they belong to the family of feature-based lexical graphs, which can be collected from lexical associations evoked from laypersons.

In future work, it will be interesting to study how the distinct performance of word embeddings that are informed by different empirical data and embedding methods may have an equally distinctive impact into downstream tasks that take pretrained word embeddings as input.

[15] The code and data sets used in this paper can be found at `https://github.com/nlx-group/Graph-vs.-Text-based-Embeddings`.

Acknowledgements

The research results presented here were supported by PORTULAN CLARIN Infrastructure for the Science and Technology of Language, by the National Infrastructure for Distributed Computing (INCD) of Portugal, and by the ANI/3279/2016 grant.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies Conference (NAACL-HLT2009)*, pages 19–27. Association for Computational Linguistics.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Eneko Agirre and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41. Association for Computational Linguistics.

John Robert Anderson. 1974. Retrieval of propositional information from long-term memory. *Cognitive Psychology*, 6(4):451–474.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL2012)*, pages 23–32. Association for Computational Linguistics.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872, Vancouver, Canada. Association for Computational Linguistics.

Daniel G. Bobrow and Donald Arthur Norman. 1975. Some principles of memory schemata. In *Representation and Understanding: Studies in Cognitive Science*, page 131–149. Elsevier.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL2012)*, pages 136–145. Association for Computational Linguistics.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2017. A comprehensive survey of graph embedding: Problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties.

Ondřej Cífka and Ondřej Bojar. 2018. Are BLEU and meaning representation in opposition?

Simon De Deyne, Daniel J Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single-word associations. *Behavior Research Methods*, 45(2):480–498.

Simon De Deyne, Danielle J Navarro, Amy Perfors, Marc Brysbaert, and Gert Storms. 2018. The "small world of words" english word association norms for over 12,000 cue words. *Behavior research methods*, pages 1–20.

Simon De Deyne, Amy Perfors, and Daniel J Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING2016)*, pages 1861–1870.

William K Estes. 1994. *Classification and Cognition*. Oxford University Press.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Jure Ferlez and Matjaz Gams. 2004. Shortest-path semantic distance measure in wordnet v2.0. *Informatica*, 28:381–386.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies Conference (NAACL-HLT25)*, pages 1434–1439. Association for Computational Linguistics.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1406–1414. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41:665–695.

G. Hirst and D. St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press.

Keith J Holyoak and Kyunghee Koh. 1987. Surface and structural similarity in analogical transfer. *Memory & Cognition*, 15(4):332–340.

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP-CONLL2007*, Prague, Czech Republic.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*, pages 13–24. Association for Computational Linguistics.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*.

C. Leacock and M. Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 265–285. MIT Press.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070*.

D. Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of 15th International Conference on Machine Learning*.

David E Meyer and Roger W Schvaneveldt. 1971. Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology*, 90(2):227.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. https://code.google.com/archive/p/word2vec/.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Marvin Minsky. 1975. A framework for representing knowledge. In *Psychology of Computer Vision*. McGraw-Hill.

Mark Newman. 2010. *Networks: An Introduction*. Oxford University Press.

Charles E Osgood, George J Suci, and Percy H Tannenbaum. 1957. The measurement of meaning. *Urbana: University of Illinois Press*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

M Ross Quillan. 1966. Semantic memory. Technical report, Bolt Beranek and Newman Inc., Cambridge MA.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. European Language Resources Association.

P. Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11.

Lance J Rips. 1975. Inductive judgments about natural categories. *Journal of Verbal Learning and Verbal Behavior*, 14(6):665–681.

Brian H Ross. 1984. Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, 16(3):371–416.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Chakaveh Saedi, António Branco, João António Rodrigues, and João Silva. 2018. Wordnet embeddings. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 122–131. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*, pages 455–465.

Persistence pays off: Paying Attention to What the LSTM Gating Mechanism Persists

Giancarlo D. Salton [*]
UNOESC
Campus Chapecó
Chapecó, Brazil
giancarlo.salton@unoesc.edu.br

John D. Kelleher
ADAPT Research Centre
Technological University Dublin
Dublin, Ireland
john.d.kelleher@dit.ie

Abstract

Language Models (LMs) are important components in several Natural Language Processing systems. Recurrent Neural Network LMs composed of LSTM units, especially those augmented with an external memory, have achieved state-of-the-art results. However, these models still struggle to process long sequences which are more likely to contain long-distance dependencies because of information fading and a bias towards more recent information. In this paper we demonstrate an effective mechanism for retrieving information in a memory augmented LSTM LM based on attending to information in memory in proportion to the number of timesteps the LSTM gating mechanism persisted the information.

1 Introduction

Language Models (LM) are important components in Natural Language Processing systems, such as Statistical Machine Translation and Speech Recognition (Schwenk et al., 2012). An LM is generally used to compute the likelihood of a sequence of words appearing in a given language. Recently, Recurrent Neural Networks LMs (RNN-LMs) have became the state-of-the-art approach to LMs (Józefowicz et al., 2016). However, RNN-LMs struggle to keep their level of performance as the length of the input increases.

A typical RNN-LM propagates a context vector that integrates information about previous inputs to use for the next prediction. Consequently, the information that is captured at the beginning of a sequence containing a long-distance dependency is likely to have faded from the context by the time the model spans that dependency. To address these limitations, several "memory-augmented" RNN-LMs architectures have been developed that attempt to retrieve relevant information from its past timesteps (e.g., Tran et al. (2016), Cheng et al. (2016), Daniluk et al. (2017), Merity et al. (2017), Grave et al. (2017) and Salton et al. (2017))

In this paper, we demonstrate that an efficient and effective mechanism for a memory augmented LSTM based LM (LSTM-LM) to retrieve important information from its history is to construct a representation of the LSTM unit state history that weights information in proportion to the number of timesteps the unit persisted the information. Using this strategy reinforces the decisions of the LSTM gating mechanism at each timestep regarding what is important in a sequence. Our models achieve competitive results on the Penn Treebank (Marcus et al., 1994) and on the wikitext2 (Merity et al., 2017). Structure: §2 presents the architecture of LSTMs; §3 discusses the effect of uniformly weighting the hidden states of an LSTM; §4 illustrates persistence of information in an LSTM and describes our memory augmented LSTM-LM; §5 presents experiments and results; §6 contextualizes our findings; and §7 our conclusions.

2 Long Short-Therm Memory

LSTM units (*aka.* LSTM cells) are now a normal building block for neural based NLP systems (Bradbury et al., 2017; Murdoch and Szlam, 2017). LSTMs retain and propagate information through the dynamics of the LSTM memory cell, hidden state and gating mechanism (including the *input*, *forget*, and *output* gates). The LSTM memory cell retains information that is only known by the unit itself and the hidden state shares informa-

[*] Work done while the author was at the ADAPT Center and the Dublin Institute of Technology.

tion to other LSTM units in the same or any next layer of the network. This way, the units can decide what to keep in memory and how much of that information it wants the other units/layers to know about it. If something is deemed important, the units will both keep it in memory and let other units/layers to know about it. The gating mechanism controls the flow of information between the memory cell and the hidden state. Therefore, the gating mechanism plays an important role on the LSTM hidden dynamics.

The computations of a standard LSTM unit (Gers et al., 2000) (without *peephole connections*) involve iterating over the following equations

$$\widetilde{\mathbf{c}}_t = tanh(\mathbf{W}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{(t-1)} + \mathbf{b}) \quad (1)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ii}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{(t-1)} + \mathbf{b}_i) \quad (2)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{if}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{(t-1)} + \mathbf{b}_f) \quad (3)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{io}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{(t-1)} + \mathbf{b}_o) \quad (4)$$

$$\mathbf{c}_t = \mathbf{f}_t \times \mathbf{c}_{(t-1)} + \mathbf{i}_t \times \widetilde{\mathbf{c}}_\mathbf{t} \quad (5)$$

$$\mathbf{h}_t = \mathbf{o}_t \times \tanh(\mathbf{c}_t) \quad (6)$$

where the weight matrices $\mathbf{W}_{i*}$ are associated to the input; the weight matrices $\mathbf{W}_{h*}$ are associated with the recurrence; the vectors $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the activation vectors produced by the *input*, *forget* and *output gates* respectively; $\widetilde{\mathbf{c}}_t$ is the candidate memory cell state; $\mathbf{c}_t$ is the new memory cell state; and $\mathbf{h}_t$ is the output of the unit.

The *candidate vector* (Eq. 1) contains information extracted from the input to the LSTM and, together with the *input gate vector* (Eq. 2) and *forget gate vector* (Eq. 3), is used to update the *memory cell* (Eq. 5). That update decides how much of the input is important to the memory cell, how much the *memory cell* will keep from its own content and what will be remembered in the memory cell for the next iteration.The *output vector* (Eq. 4) decides how much of the content in the memory cell $\mathbf{c}_t$ will be known on the next timestep (and by cells in the next layer if it is a multi-layered LSTM or to any layer that may come next o the network).

The success of LSTM-RNNs is attributed to their ability to retain information about the input sequence for several timesteps in their internal memory cell $\mathbf{c}_t$. That information is then made available to the next layer in the network for the amount of timesteps it is considered relevant to the current sequence. As pointed by Murdoch and Szlam (2017), each input to an LSTM makes a contribution to the hidden state of the LSTM and that is reflected when Eq. 5 is iterated. At any given timestep t, the cell state $\mathbf{c}_t$ can be decomposed into

$$\mathbf{c}_t = \sum_{i=1}^{t}(\prod_{j=i+1}^{t} \mathbf{f}_i)\mathbf{i}_i\widetilde{\mathbf{c}}_i \quad (7)$$

which, according to the authors, can be interpreted as the contribution at timestep t to the memory block $\mathbf{c}_t$ by a particular past input at timestep j. In that view, the contribution of an input to a given timestep can be understood as an importance score weighted by the LSTM's gating mechanism. Therefore, if something is important to the current context if should receive a larger importance score and be held in the memory block for a number timesteps. In addition to retaining information, Murdoch and Szlam (2017) have also demonstrated that, despite the fact that it is still difficult to interpret what specific activations in the hidden dynamics of LSTM units mean, it is possible to extract semantically meaningful rules from the memory cells to train a powerful classifier that can approximate the output of the LSTM itself. Moreover, Strobelt et al. (2016) and Karpathy et al. (2015) have demonstrated that these networks can extract meaningful attributes from the data into the memory cells. These attributes carry fine grained information and keep track of attributes such as line lengths, quotes and brackets.

Although these and other work demonstrate the power of LSTM units and their gating mechanism, RNN-LMs based on such units (LSTM-LMs) struggle to process long sequences. In our view, the main reason for this degradation in performance happens exactly because of the hidden state dynamics of the LSTM units. Once the information retained in the memory cell $\mathbf{c}_t$ is outdated, the *forget gate* $\mathbf{f}_t$ erases that block enabling the unit to store fresh data without interference from previous timesteps (Gers et al., 2000, 2003). This behaviour creates a natural bias towards more recent inputs given that the memory cell has limited capacity to store previous information and, once the memory cell is saturated, the *forget gate* will start to drop information in favour of more recent inputs. Even though the LSTM units can learn which information it must retain and for how long, the model will struggle with long sequences that are more likely to contain LDDs and that saturate the memory cell.

Once a memory cell has been saturated then, although some content has received a large importance score in past steps, it may be dropped from the memory cell (because of the inherent limitation of the LSTM's capacity of storing content) and will not be available to contribute to the next steps. For example, an LSTM-LM trained on English may persist the information related to a subject of a sentence for a number of time steps because the subject is important but this information may still have faded by the time the verb is reached. However, by augmenting the network with a memory buffer the information relating to the subject continues to be accessible so long as the memory buffer is not reset. This behaviour is an indication of why the memory augmented models such as the Neural cache model of Grave et al. (2017) and the Pointer LSTM of Merity et al. (2017) has gained success and achieved state-of-the-art results in LM research. Even though the required content has already faded from the context, the memory augmentation make it available for subsequent timesteps.

3 The Curious Effectiveness of Uniform Attention

As noted in Section 1, in recent years a number of extensions to RNN-LMs have been proposed to overcome the fading of information from context by adding a memory buffer (that is used to store the LSTM hidden states) and then at each timestep construct a representation of this history to inform the current prediction. A variety of relatively sophisticated mechanisms for retrieving information from the memory buffer have been proposed. In many cases these retrieval mechanisms include an extra neural network in the RNN-LMs that at each timestep predicts what elements in the memory buffer should be retrieved.

Salton et al. (2017) is a recent example that uses an extra neural network[1] to learn what to retrieve from memory. In this architecture at the end of a timestep the current LSTM hidden state is added to the memory buffer. At the beginning of the next timestep the additional neural network predicts an attention distribution over the elements of the buffer (*i.e.*, the previous LSTM hidden states). Using this distribution a compact representation of the RNN-LMs history is constructed by calculating a weighted sum of the elements in the memory (where the weight of each element is the attention attributed to it by the RNN). Curiously, although this architecture was successful in terms of performance the attention mechanism did not work as expected. Instead of focusing attention for each time step on particular relevant elements in memory it spread out the attention nearly uniformly across the memory. It might appear that this architecture was using a strategy of "pay equal attention to everything in the past". However, we argue this interpretation ignores the power of the LSTM gating mechanism.

Our interpretation of the uniform attention mechanism presented by Salton et al. (2017) is that their *Attentive* RNN-LM is in fact (indirectly) reinforcing the decisions of the gating mechanism of the LSTM units and is retrieving information that is persisted across multiple timesteps. This is important because it indicates that it may be more fruitful and efficient to leverage the decisions made by the LSTM gating mechanism (decisions that the network must make anyway) to drive the retrieval of information from the memory buffer rather than train a separate neural network. It is worth emphasising that to date none of the different retrieval mechanisms proposed in the literature on memory augmented LSTM-LMs have explicitly considered the behaviour of the LSTM gating mechanism.

4 The Persistence of Information

The LSTM gating mechanism will attempt to persist important information for as long as possible (or until the state is saturated). We propose that when retrieving/constructing a representation of the LSTM history from a memory buffer the information held for more than one timestep should be weighted in proportion to the number of timesteps the LSTM gating mechanism persisted it across. This way, we let the gating mechanism of the LSTM determine what is important about the input and, anything that is persisted for more than one timestep, will have a greater impact on the final prediction even if that information has already faded from the current context.

A simple and efficient way to implement this strategy is at each time point to construct a representation of the history of the RNN-LM that is simply an average of the LSTM hidden states in

[1] Similar to that proposed by Bahdanau et al. (2015) and Luong et al. (2015) for Neural Machine Translation (NMT)

the memory buffer. Pieces of information that the LSTM unit persists for several time steps will have a bigger impact on this average (simply because they are included multiple times) relative to items that are not persisted. In effect, this average weights each piece of information in proportion to the number of time steps the LSTM persisted it and so an RNN-LM that uses this average as its representation of history pays attention to what the LSTM gating mechanism persisted.

4.1 Averaging the Outputs

In this work we simplify the architecture of Salton et al. (2017) and use an average of previous outputs instead of a neural network based attention mechanism. Our intuition for this modification is that the gating mechanism of the LSTM is telling us what is important about an input and that we must find a way to make that information available for long distances in the future. In fact, Ostmeyer and Cowell (2017) have presented a model that computes a recurrent weighted average (RWA) over every past hidden state. However, the authors limit themselves to evaluate the model over simple tasks and the effectiveness of that model over language modelling is still to be demonstrated.

Compared to other memory augmented models our architecture is relatively simple. A multilayered LSTM-RNN encodes an input at each timestep and the outputs of the last recurrent layer (*i.e.*, its hidden state called h_t) is added to memory. At each timestep an average of the vectors in the memory buffer is calculated and concatenated with the h_t generated by the processing of the current input. This concatenated vector is then feed into the softmax layer which predicts the distribution for the next word in the sequence.

In our experiments with this uniform attention, we found that initialising the memory with a zero vector $\mathbf{h}_0$ and allowing the model to count this vector as part of the memory when calculating the average[2] improved the performance of the model.

5 Experiments

To test our intuitions, we evaluate the averaging process of the model using the PTB dataset using the standard split and pre-processing as in Mikolov et al. (2010) which consists of 887K, 70K and 78K tokens on the training, validation and test sets respectively. We also evaluate the model on the wikitext2 dataset using the standard train, validation and test splits which consists of around 2M, 217K tokens and 245k tokens respectively.

5.1 PTB Setup

Following Salton et al. (2017) we trained a multilayer LSTM-RNN with 2 layers of 650 units for the PTB experiment. We trained them using Stochastic Gradient Descent (SGD) with an initial learning rate of 1.0 and we halved the learning rate at each epoch after 12 epochs. We train the model to minimise the average negative log probability of the target words until we do not get any perplexity improvements over the validation set with an early stop counter of 10 epochs. We initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0 with the exception of the *forget gate* biases which is initialised at 1.0 as suggested by Jozefowicz et al. (2015). We also apply 50% dropout (Srivastava et al., 2014) to the non-recurrent connections and clip the norm of the gradients, normalized by the mini-batch size of 32, at 5.0. We also tie the weight matrix used for the transformation in the softmax layer to be the embedding matrix as in Press and Wolf (2016). Thus, the dimensionality of the embeddings is set to 650.

5.2 wikitext2 Setup

For the wikitext2 experiments we trained a multilayer LSTM-RNN with 2 layers of 1000 units. We also used SGD to minimise the average negative log probability of the target words with an initial learning rate of 1.0. We decayed the the learning rate by a factor of 1.15 at each epoch after 14 epochs and we used an early stop counter of 10 epochs. Similarly to the PTB experiment, we initialize the weight matrices of the network uniformly in $[-0.05, 0.05]$ while all biases are initialized to a constant value at 0.0 with the exception of the *forget gate* biases which is initialised at 1.0. For this model we apply 65% dropout to the non-recurrent connections and clip the norm of the gradients, normalized by the mini-batch size of 32, at 5.0. Once again, we tie the weight matrix used for the transformation in the softmax layer to be the embedding matrix. Thus, the dimensionality of the embeddings is set to 1,000.

[2] In other words, the index of the memory starts at timestep 0 instead of timestep 1. Thus, the memory at any given timestep t will be of length $t + 1$.

5.3 Data Manipulation and Batch Processing

When training each model, we use all sentences in the respective training set, but we truncate all sentences longer than 35 words and pad all sentences shorter than 35 words with a special symbol so all have the same length. We use a vocabulary size of 10k for the PTB and 33,278 for the wikitext2. Each of the mini-batches we use for training are then composed of 32 of these sentences taken from the dataset in sequence.

Contrary to the recent trend in the field, we do not allow successive mini-batches to sequentially traverse the dataset. We reinitialize the hidden state of the LSTM-RNN at the beginning of each mini-batch, by setting it to all zeros. Our motivation for not sequentially traversing the dataset is that although sequentially traversing has the advantage of allowing the batches to be processed more efficiently, some dependencies between words may not be learned if batch traversing is in use as the mini-batch boundaries can split sentences. We also found that allowing the initial state of all zeros to be included in the memory when averaging improves the performance of the Average RNN-LM.

5.4 Results

Table 1 presents the results in terms of perplexity of the models trained over the PTB dataset. As we can see, the results obtained by the Averaging RNN-LM are similar to those obtained by the *Attentive* RNN-LMs of Salton et al. (2017). Despite the simple method to retrieve information from the previous timesteps, the Averaging RNN-LM achieves the same level of performance of more complex models with less computation overhead.

Table 2 presents the results in terms of perplexity of the models trained over the wikitext2 dataset. Although the Averaging RNN-LM is still behind the *Attentive* RNN-LMs and the Neural cache model of Grave et al. (2017) on this dataset, the results are encouraging given the simplicity of the Averaging RNN-LM.

However, we should note that none of these models perform at the same level of the state-of-the-art models such as those of Merity et al. (2017) and Takase et al. (2018) as we can see in Tables 1 and 2. These models use advanced regularization techniques and matrix factorization for training the RNN-LMs whilst our Averaging RNN-LM use standard LSTM trainig regime and regularization techniques. Nevertheless, we believe that by adding the regularization scheme of the AWD-LSTM and the direct output connection of AWD-LSTM-DOC to our models we can bridge that performance gap.

6 Discussion

The Averaging LSTM-LM achieves the lowest perplexity for a single model on the PTB (see Table 1). Given the similarity of the results between the *Attentive* RNN-LMs of Salton et al. (2017) and the Averaging LSTM-LM it would appear that our hypothesis that the *Attentive* RNN-LMs was (indirectly) learning to use the dynamics of the LSTM gating mechanism is correct.

Focusing on the results for the wikitext2 dataset, the Neural cache model (Grave et al., 2017) has a higher performance than our model on this dataset. We are not able to estimate the number of parameters for the Neural cache model so we have not included the parameter size of that model in the table. In discussing the wikietext2 results it is worth noting that the *Attentive* RNN-LMs of Salton et al. (2017) and the Averaging LSTM-LM are the only models in Table 2 that reset their memory at each sentence boundary whereas the memory buffers of other models were allowed to span sentence boundaries.

The results for the wikitext2 dataset highlights an interesting trade-off and design choice for memory augmented LSTM-LMs. One approach is to use a dynamic length memory buffer which resets at sentence boundaries and uses a simple mechanism, such as averaging, to construct a representation of the memory to inform the prediction at each timestep. This is the approach we have proposed in this paper. This approach has the advantages of simplicity and that the memory length can be anchored to landmarks in the history, such as sentence boundaries. This approach is most appropriate for sentence based NLP tasks such as sentence based Machine Translation. There is a question, however, regarding whether this approach will scale to very long sequences (such as documents) as averaging over long-histories may result in all histories appearing similar. We have done some initial experiments where we have permitted the memory buffer to hold longer sequences before being reset and the performance of the Averaging LSTM-LM dipped. The alternative approach is to use a larger memory buffer and a

Model	Params	Valid. Set	Test Set
Single Models			
Neural cache model (size = 500) (Grave et al., 2017)	-	-	72.1
Attentive LM w/ *combined* score function (Salton et al., 2017)	14.5M	72.6	70.7
Attentive LM w/ *single* score function (Salton et al., 2017)	14.5M	71.7	70.1
Averaging RNN-LM	14.1M	71.6	69.9
AWD-LSTM (Merity et al., 2017)	24M	60.0	57.3
AWD-LSTM-DOC (Takase et al., 2018)	23M	54.12	**52.38**

Table 1: Perplexity results over the PTB. Please note that we could not calculate the number of parameters for some models given missing information in the original publications.

Model	Params	Valid. Set	Test Set
Averaging RNN-LM	50M	74.6	71.3
Attentive LM w/ *combined* score function (Salton et al., 2017)	51M	74.3	70.8
Attentive LM w/ *single* score function (Salton et al., 2017)	51M	73.7	69.7
Neural cache model (size = 2000) (Grave et al., 2017)	-	-	68.9
AWD-LSTM (Merity et al., 2017)	33M	68.6	65.8
AWD-LSTM-DOC (Takase et al., 2018)	37M	60.29	**58.03**

Table 2: Perplexity results over the wikitext2. Please note that we could not calculate the number of parameters for some models given missing information in the original publications.

more sophisticated retrieval mechanism, for example the Neural cache model of Grave et al. (2017). As the wikitext2 results demonstrate this second approach works well for large datasets where the sentences are in sequence, the cost of this approach being a more complex architecture.

7 Conclusions

In this paper we have highlighted the power of the LSTM gating mechanism and argued that the persistence dynamics of this mechanism can provide useful clues regarding what information is important within a sequence for language modelling. We believe that attending to the information that an LSTM gating mechanism has decided is important in an input sequence at a given timestep (and hence has persisted to a later timestep) is a natural way of deciding what information will be useful again at a subsequent timestep. Even if the information contained in the LSTM is replaced or altered later in the process, we argue that it is relevant to the entire history in proportion to the amount of timesteps it was held. Informed by this hypothesis, in our work we demonstrated that a simple average of the previous LSTM hidden states in memory is an effective mechanism for providing information to the current timestep about previous inputs.

Admittedly, rating the importance of information in terms of the number of timesteps the LSTM persisted it for is a relatively simplistic view of the dynamics of LSTM units and of the complexity of language. Furthermore, implementing this strategy using an average of past states is also a relatively blunt way of instantiating this approach. However, as our results demonstrate this simple approach is effective and we understand this is a starting point. By drawing attention to the signals implicit in the dynamics of LSTM units we hope to contribute to the development of more efficient LMs. At the same time, the fact that the internal dynamics of an LSTM unit may be used to explicitly signal what is important and what should be retrieved from a memory buffer may suggest alternative constraints and opportunities that should be considered in the design of neural units and by doing so contribute to the development of a new class of units for use in RNN-LMs.

Acknowledgments

This research was partly supported by the ADAPT Research Centre, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*. volume abs/1409.0473v6.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-Recurrent Neural Networks. *International Conference on Learning Representations (ICLR 2017)* .

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 551–561.

Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly Short Attention Spans in Neural Language Modeling. *5th International Conference on Learning Representations (ICLR'2017)* .

Yarin Gal and Zoubin Ghahramani. 2015. A theoretically grounded application of dropout in recurrent neural networks.

Felix A. Gers, Jürgen A. Schmidhuber, and Fred A. Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural Comput.* 12(10):2451–2471. https://doi.org/10.1162/089976600300015015.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.* 3:115–143. https://doi.org/10.1162/153244303768966139.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. *5th International Conference on Learning Representations (ICLR'2017)* .

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15, pages 2342–2350.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv* abs/1506.02078. http://arxiv.org/abs/1506.02078.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1412–1421.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*. pages 114–119.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *5th International Conference on Learning Representations (ICLR'2017)* .

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

W. James Murdoch and Arthur Szlam. 2017. Automatic rule extraction from long short term memory networks. *arXiv* abs/1702.02540. http://arxiv.org/abs/1702.02540.

Jared Ostmeyer and Lindsay Cowell. 2017. Machine learning on sequential data using a recurrent weighted average. *arXiv* abs/1703.01253. http://arxiv.org/abs/1703.01253.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. volume abs/1608.05859.

Giancarlo D. Salton, Robert J. Ross, and John D. Kelleher. 2017. Attentive language models. In *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*.

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*. pages 11–19.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv* abs/1606.07461. http://arxiv.org/abs/1606.07461.

Sho Takase, Jun Suzuki, and Masaaki Nagata. 2018. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 4599–4609. https://doi.org/10.18653/v1/D18-1489.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. *arXiv* abs/1601.01272.

Development and Evaluation of Three Named Entity Recognition Systems for Serbian - The Case of Personal Names

Branislava Šandrih
University of Belgrade
Faculty of Philology
Belgrade, Serbia
branislava.sandrih@fil.bg.ac.rs

Cvetana Krstev
University of Belgrade
Faculty of Philology
Belgrade, Serbia
cvetana@matf.bg.ac.rs

Ranka Stanković
University of Belgrade
Faculty of Mining and Geology
Belgrade, Serbia
ranka@rgf.bg.ac.rs

Abstract

In this paper we present a rule- and lexicon-based system for the recognition of Named Entities (NE) in Serbian newspaper texts that was used to prepare a gold standard annotated with personal names. It was further used to prepare training sets for four different levels of annotation, which were further used to train two Named Entity Recognition (NER) systems: Stanford and spaCy. All obtained models, together with a rule- and lexicon-based system were evaluated on two sample texts: a part of the gold standard and an independent newspaper text of approximately the same size. The results show that rule- and lexicon-based system outperforms trained models in all four scenarios (measured by F_1), while Stanford models have the highest recall. The produced models are incorporated into a Web platform NER&Beyond that provides various NE-related functions.

1 Introduction

Named Entity Recognition is the task of identifying named entities in text (Nadeau and Sekine, 2007), which is often used as a first step in question answering, information retrieval, anaphora resolution, topic modeling, etc. The first Named Entity set had 7 types (Grishman and Sundheim, 1996): organization, location, person, date, time, money and percent expressions. Sekine et al. (2002) proposed a NE hierarchy which contains about 150 NE types.

There are three categories of NER systems: 1) The rule-based (RB) (Krupka and Hausman, 1998; Friburger and Maurel, 2004); 2) the Machine Learning (ML) based (Finkel and Manning, 2009; Singh et al., 2010); and 3) hybrid methods (Jansche and Abney, 2002). The ML-based methods can often be "black boxes", in comparison with RB techniques which are easy to interpret. Yet, ML-based methods are state-of-the-art. Such example is a Stanford Named Entity Recognizer (Manning et al., 2014), which can be trained for many languages. Other notable NER platforms include GATE (Desktop application that enables NER across many languages and domains),[1] OpenNLP (rule-based and statistical NER),[2] spaCy (Honnibal and Montani, 2017) (module written in Python, used for advanced NLP)[3] and many others.

For Serbian, thus far a rule-based and lexicon-based NER system was developed – SrpNER (Krstev et al., 2014). Its development started with the recognition of a NE class present in all NE schemes, personal names (Krstev et al., 2005), while the recognition of other main NE classes was subsequently added. In the next Section we present briefly this system and how it was used to produce the corpus of newspaper texts annotated with personal names – the gold standard. Section 3 describes NER systems based on Machine Learning methods that were trained on the corpus derived from the gold standard, while the evaluation and discussion of results are presented in Sections 4 and 5, respectively. In Section 6 we present a web platform that enables use and evaluation of these systems. Finally, some directions for future work are given in the last Section.

[1] GATE, https://gate.ac.uk/
[2] OpenNLP, https://opennlp.apache.org/
[3] spaCy, https://spacy.io/

Proceedings of Recent Advances in Natural Language Processing, pages 1060–1068,
Varna, Bulgaria, Sep 2–4 2019.

2 SRPNER and the Gold Standard

2.1 Rule-Based NER for Serbian

The first NER system for Serbian was a rule- and lexicon-based system developed several years ago. It has been designed to recognize the main classes of NEs: 1) numerical expressions (measurement and money), 2) temporal expressions (date and time, and 3) name expressions (personal, geopolitical and organization names).

The system was designed in a form of the cascades of Finite-State Transducers (FST) in which every transducer recognizes and tags a certain class of NEs (Friburger and Maurel, 2004; Maurel et al., 2011). Each transducer rely in its work on the results of previous transducers and on e-dictionaries of Serbian (Vitas and Krstev, 2012). E-dictionaries play an important role specifically in the recognition of name expressions, since, beside general lexica, they contain many proper names, both personal and geopolitical. The system is modular which means that steps can be omitted and can change order; however, it performs best when used in predefined way since in each step the disambiguation of some names is performed.

SRPNER presented in (Krstev et al., 2014) recognizes 11 classes of NEs: dates (moments and periods), time (moments and periods), money expressions, measurement expressions, geopolitical names (countries, settlements, oronyms and hydronyms), and personal names (one or more last names with or without first names and nicknames). The presented evaluation results for the recognition of all mentioned NEs obtained on a sample of newspaper texts were $F_1 = 0.96$ ($R = 0.94$; $P = 0.98$). The system also recognized titles, roles and functions of persons when they accompany personal names.

Since this first results, system has been continually improved by adding new NE classes (organization names) and new sub-classes (e.g. for geopolitical names: regions, super-regions and city counties). In addition, the e-dictionaries of Serbian were also continually improved and enhanced, and that by itself contributes to better performance of SRPNER.

The new version of this system recognizes more variations for naming persons: distinguished persons and first names alone. Moreover, system distinguishes names used for men from those used for women (Krstev et al., 2015). Presented results show that the system was more successful in recognizing names of men than women.

The output of the system are texts with XML tags for recognized entities inserted in them. Since the system allows embedded NEs, recognized names of persons can be components of other NEs, e.g. organizations.

2.2 The Preparation of the Gold Standard

The system presented in the previous section was used for the preparation of the gold standard – a large text sample annotated with personal names dubbed GOLDPERS. The sample consists of short news published on the Web in the period 2009–2016 by 4 Serbian daily newspapers (*Politika*, *Danas*, *Blic*, *Novosti*), one news portal (*B92*) and one weekly magazine (*Bazar*). The sample consists of 321,127 tokens (simple running words).

The forms of personal names taken into account and their tagging are presented in Table 1. The gold standard was produced following these steps:[4]

- Each text was annotated using SRPNER;
- Tags that did not refer to personal names were deleted;
- The remaining tags were evaluated as correct, partially correct (overlapping), not correct (not a name);
- The missing tags were inserted, and typos that led to incorrect tagging were corrected.

For some texts this process was repeated from one to four times which yielded "four levels" of gold standard. Between these repeated runs the development of SRPNER continued, as well as the enhancement of e-dictionaries of Serbian.

3 Training Different NER Systems

3.1 Training Sets

The gold standard GOLDPERS contains $9,046$ sentences, each one enclosed in `<seg>...</seg>` tag. Named entities in sentences are annotated using tags listed in Table 1. These tags contain different levels of information: name type, role, gender. We wanted to examine the recognition of NEs on different level of details. Therefore, on the basis of the gold standard, we developed its four versions by

[4] The evaluation was performed as a homework by several generations of students of Library and Information Sciences at the Faculty of Philology, University of Belgrade, in the scope of the course Information Retrieval. Their work was checked by their professor, which means that texts were twice evaluated in each run.

type	example	original/translation
full	<m.persName.full>Mohamed El Baradei</m.persName.full>	*Mohamed ElBaradei*
	<f.persName.full>Ketrin Ešton</f.persName.full>	*Catherine Ashton*
last	<m.persName.last>Obami</m.persName.last>	*Obama*
	<f.persName.last>Timošenkove</f.persName.last>	*Tymoshenko*
first	<m.persName.first>Džim</m.persName.first>	*Jim*
	<f.persName.first>Tamara</f.persName.first>	*Tamara*
role	<role>generalnog sekretara</role>	*Secretary General*
	<m.persName.full>Bana Ki Muna </m.persName.full>	*Ban Ki-moon*
	<role>Komesarka UN za ljudska prava</role>	*UN Commissioner for*
	<f.persName.full>Nejvi Pilaj</f.persName.full>	*Human Rights Navi Pillay*
spec	<role>papa</role>	*Pope*
	<m.persName.first>Franja</m.persName.first>	*Francis*
	<role>kraljice</role>	*Queen*
	<f.persName.first>Viktorije</f.persName.first>	*Victoria*

Table 1: Examples of types of personal names and their tags; M - masculine, F - feminine personal names

defining different mappings of XML tags. These mappings are named and represented in Table 2. Different versions of GoldPers are illustrated with examples presented in Table 3.

We split each of these four versions of the gold standard (namely PERS_$\{1, 3, 4, 9\}$) into training and test sets (containing $8,151$ and 895 sentences, respectively). We named this gold test set Students-Gold.

As we wanted to have an independent text of the similar structure and content we prepared an additional set of news articles from *Danas* daily journal. This was one of the source journals for the Students-Gold, but for this new sample, we have randomly chosen recent news (from year 2018, that is 2 years after the most recent news in GoldPers). This set of articles containing 860 sentences was tagged with SrpNER and manually corrected, thus producing the second test set Danas-Gold. The distribution of NE tags in the training set, and both test sets is given in Table 4.

We used four versions of the gold standard to train two different Named Entity Recognition systems: spaCy NER (Subsection 3.2) and Stanford NER (Subsection 3.3). Trained models for Serbian are available on NER&Beyond platform, which is presented in Section 6.

3.2 spaCy NER

spaCy (Honnibal and Montani, 2017) is a free, open-source library for advanced Natural Language Processing in Python. For different natural languages, it is able to perform tokenization, POS-tagging, dependency parsing, lemmatization, sentence boundary detection, named entity recognition, similarity comparing, text classification, etc. It offers statistical models for a variety of languages, which can be installed as individual Python modules. It supports training new language models, as well.

We used it for training NER on our four versions of GoldPers. We coded a Python script that transforms each sentence into a training sample, represented as a list of triplets.[6] For example, for the sentence “srpski reditelj Aleksandar Saša Petrović” (*Serbian director Aleksandar Saša Petrović*), the corresponding triplet representation for the PERS_4 model would be:

$$(0, 14, "ROLE"), (16, 39, "PERS\_FULL")$$

where the first and the second element represent the start and the end character offset, while the third element represents the NE itself. Each of the four models were trained in 10 iterations, using 0.5 value for the drop-out parameter. These trained models can be inspected online.[7]

3.3 Stanford NER

Stanford NER (Manning et al., 2014) is a Java implementation of a Named Entity Recognizer by the Stanford Natural Language Processing group. It is also known as *CRFClassifier*, since

[6] Training NER in spaCy, https://spacy.io/usage/training#ner

[7] Visualization of spaCy NER for Serbian, http://ner.jerteh.rs/

	PERS_1	PERS_3	PERS_4	PERS_9
m.persName.full	PERS	PERS_FULL	PERS_FULL	PERS_FULL_M
f.persName.full				PERS_FULL_F
x.persName.full				PERS_FULL_X
m.persName.first		PERS_FIRST	PERS_FIRST	PERS_FIRST_M
f.persName.first				PERS_FIRST_F
x.persName.first				PERS_FIRST_X
m.persName.last		PERS_LAST	PERS_LAST	PERS_LAST_M
f.persName.last				PERS_LAST_F
x.persName.last				PERS_LAST_X
role	-	-	ROLE	-

Table 2: Mappings of NE tags GoldPers to set tags used for training[5]

PERS_1	Film "Mančester na moru" <PERS>Keneta Lonergana</PERS> je u konkurenciji za šest "Oskara", dok je <PERS>Izabel Iper</PERS>, glavna junakinja filma "Ona" Holanđanina <PERS>Pola Ferhufena</PERS> nominovana za najbolju žensku ulogu.
PERS_3	Film "Mančester na moru" <PERS_FULL>Keneta Lonergana</PERS_FULL> je u konkurenciji za šest "Oskara", dok je <PERS_FULL>Izabel Iper</PERS_FULL>, glavna junakinja filma "Ona" Holanđanina <PERS_FULL>Pola Ferhufena</PERS_FULL> nominovana za najbolju žensku ulogu.
PERS_4	Film "Mančester na moru" <PERS_FULL>Keneta Lonergana</PERS_FULL> je u konkurenciji za šest "Oskara", dok je <PERS_FULL>Izabel Iper</PERS_FULL>, <ROLE>glavna junakinja filma "Ona"</ROLE> <ROLE>Holanđanina</ROLE> <PERS_FULL>Pola Ferhufena</PERS_FULL> nominovana za najbolju žensku ulogu.
PERS_9	Film "Mančester na moru" <PERS_FULL_M>Keneta Lonergana</PERS_FULL_M> je u konkurenciji za šest "Oskara", dok je <PERS_FULL_F>Izabel Iper</PERS_FULL_F>, glavna junakinja filma "Ona" Holanđanina <PERS_FULL_M>Pola Ferhufena</PERS_FULL_M> nominovana za najbolju žensku ulogu.

Table 3: The same sentence in four versions of the gold standard – PERS_$\{1, 3, 4, 9\}$

it is based on Conditional Random Fields (Lafferty et al., 2001). For training this model,[8] we had to transform our texts into CoNLL02 IOB format (namely, "inside - outside - beginning") with *conll* extension (Sang, 2002). For this purpose, we used XML ↦ CoNLL converter available within NER&Beyond on-line tool. An example of this format is given in Table 5.

4 Evaluation

In order to evaluate three NER system for personal names in Serbian, we need to have the output results in the same format. After running spaCy NER on a text, an output is provided in BRAT format with *ann* extension. This format is similar to the one for spaCy training, described in subsection 3.2. For the example given in the same subsection, an output file has the following content:

T1 ROLE 0 14 srpski reditelj
T2 PERS_FULL 16 39 Aleksandar Saša Petrović

After running StanfordNER on a text, an output is provided in already mentioned CoNLL02 format. We used CoNLL02 ↦ BRAT converter available within NER&Beyond online tool.

Finally, for both spaCy NER and StanafordNER output files, we applied ANN + TEXT ↦ XML converter offered by Gemini, also available within NER&Beyond online tool. An output of SrpNER is already an XML file with marked named entities, as is the gold standard explained in Section 2 and illustrated in Table 3.

We evaluated three NER systems using the open source Gemini tool, described in Section 6, that offers various options for files comparison (Feng, 2018). It is possible to select matching type: *strict*,

[8] Training Stanford NER, https://nlp.stanford.edu/software/crf-faq.shtml#a

	Entity	**Train**	**S**	**D**
P_1	PERS	6312/2825	901/471	936/414
P_3	FULL	3280/1865	461/311	433/266
	FIRST	360/185	54/36	21/16
	LAST	2672/785	386/124	482/136
P_4	FULL	3280/1865	461/311	433/266
	FIRST	360/185	54/36	21/16
	LAST	2672/785	386/124	482/136
	ROLE	2069/1410	266/198	269/220
P_9	FULL$_{\text{m}}$	2732/1506	389/253	356/223
	FULL$_{\text{f}}$	547/358	72/59	75/42
	FULL$_{\text{x}}$	1/1	0	2/2
	FIRST$_{\text{m}}$	253/117	34/21	15/14
	FIRST$_{\text{f}}$	107/69	20/15	6/6
	FIRST$_{\text{x}}$	0	0	0
	LAST$_{\text{m}}$	2460/686	358/106	409/115
	LAST$_{\text{f}}$	192/93	27/17	73/21
	LAST$_{\text{x}}$	20/20	1/1	0

Table 4: Number of NE tags vs. number of different name forms in the training set and in test sets STUDENTS-GOLD (S) and DANAS-GOLD (D)

Fascinirala	O
me	O
je	O
Sonja	B-PERS
Savić	I-PERS
svojim	O
transformacijama,	O
Anica	B-PERS
Dobra	I-PERS
šarmom...	O

Table 5: CoNLL02 IOB format – the beginning of the sentence *I was fascinated by Sonja Savić and her transformation, Anica Dobra and her charm...*

where exact overlapping of NE annotations is subsumed (both annotation labels are the same) or *weighted*, where partial overlapping is taken into account, but with some weighted value to measure overlapping segment. To indicate alignment type, one can choose among the two options: the first option is *greedyMatching*, where the matching of annotations in the first and second files is done with a greedy algorithm that tries to match the closest annotations first. The second option is *maxMatching*, where the matching of annotations in the first and second file is done optimally using a maximum matching algorithm in bipartite graphs. An annotation in the first file will correspond to at most one annotation in the second file, and vice versa, in both cases.

We run 2 × 3 × 4 evaluation rounds: two test sets, three NERs and four models per each. All trials were run with *strict* matching type and *maxMatching* alignment type.

To indicate the chosen score type to evaluate the correspondence between one annotation from the first file and one annotation from the second file, calculation of precision P, recall R and F-measure in Gemini comes in three different flavors:

weak an annotation of the first file will be considered as corresponding to an annotation of the second file if they intersect on at least one character;

strict an annotation of the first file will be considered as corresponding to an annotation of the second file if they start and end exactly at the same characters;

weighted the match is scored by the ratio of the number of characters common to both annotations divided by the total number of characters covered by at least one of the two annotations.

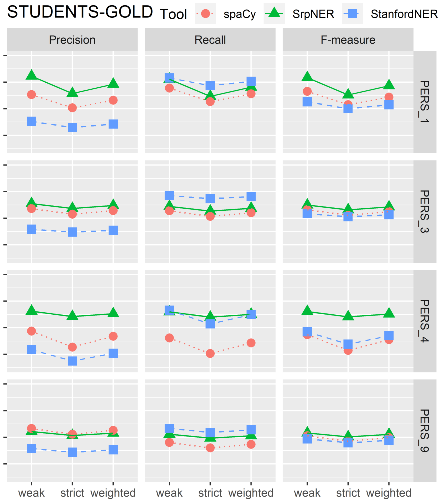


Figure 1: The evaluation of SPACY NER, SRPNER and STANFORD NER on STUDENTS-GOLD

5 Discussion

The results of three NER systems, four models and two test texts are presented in Table 6. The results show that in all cases (except one) SRPNER achieved the best precision, in all cases (except one) STANFORD NER achieved the best recall,

		spaCy NER			SrpNER			Stanford NER		
	model	***P***	***R***	***F_1***	***P***	***R***	***F_1***	***P***	***R***	***F_1***
Students	PERS_1	0.804	0.827	0.815	**0.857**	0.847	**0.852**	0.729	**0.887**	0.800
	PERS_3	0.815	0.807	0.811	**0.837**	0.827	**0.832**	0.748	**0.873**	0.806
	PERS_4	0.727	0.703	0.715	**0.842**	**0.840**	**0.841**	0.675	0.815	0.738
	PERS_9	**0.812**	0.760	0.785	0.807	0.797	**0.802**	0.744	**0.818**	0.779
Danas	PERS_1	0.819	0.870	0.844	**0.916**	0.841	**0.877**	0.790	**0.902**	0.842
	PERS_3	0.864	0.854	0.859	**0.905**	0.830	**0.866**	0.791	**0.874**	0.830
	PERS_4	0.807	0.792	0.799	**0.907**	0.824	**0.863**	0.716	**0.825**	0.767
	PERS_9	0.818	0.794	0.806	**0.872**	0.800	**0.835**	0.761	**0.808**	0.784

Table 6: The comparison of strict precision, recall and F_1 between NER systems, models and test sets.

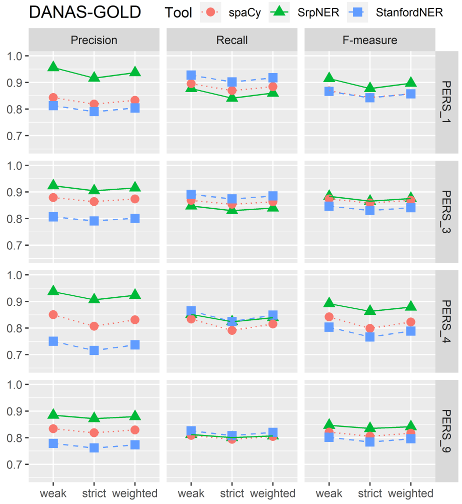


Figure 2: The evaluation of spaCy NER, SrpNER, and Stanford NER on Danas-Gold

while SrpNER achieved the highest F_1 measure in all cases. Stanford NER and SrpNER performed better on both test texts with models that use less tags (PERS_1 and PERS_2), while SrpNER performed significantly worse only for the model with 9 tags. Contrary to our expectations, all NER systems for all models achieved better results for the independent test text Danas_Gold than for the test set randomly chosen from the gold standard. Namely, a number of news in Gold-Pers that come from 6 different sources were produced at the same time period, and thus involved same persons. However, that did not influence results favorably towards Students_Gold test text.

All measures are for all NER systems and models highest for weak calculation, followed by weighted, the strict calculation giving the lowest result. However, as displayed in Figure 1 for the Students-Gold test set and in Figure 2 for the Danas-Gold test set the mutual relationship between three NER system remains the same.

We also compared performance of all three NER systems by each named entity type (Figure 3). Results for all three models distinguishing entity types show that all three systems perform poorly in recognizing first names only. For Stanford NER and spaCy NER it can be explained by the considerably smaller number of these tags in training texts compared to other tags (see Table 4). As for SrpNER one can presume that developers devoted less effort to this entity type occurring only occasionally in newspaper texts. Similarly, in all experiment settings, the recognition of full names was better than the recognition of last names only. Again, the number of last name tags was smaller than the number of full name tags (Table 4). A rule based system SrpNER makes use of a personal name context in cases of disambiguity which tends to be less specific in the case of the use of a last name only.

One can also note that when using the model PERS_4, SrpNER system performs best in recognizing the role entity. Between other two systems, Stanford NER achieves better recall and SpaCy NER slightly better precision.

The chart for model PERS_9 shows that all systems according to F_1 measure recognize better masculine names than feminine names regardless of entity types. Feminine names show grater variety of forms than masculine names, especially when only last names are used; moreover, they occur significantly less than masculine names in newspaper texts (as pointed in (Krstev et al., 2015) there is approximately one feminine name per 7 seven masculine names) and confirmed in our training sets (Table 4).

We compared the performance of SRPNER with its previously reported results. In (Krstev et al., 2005) results for the recognition of personal names were $P = 0.97$, $R = 0.86$, $F_1 = 0.91$. One notes that all measures are higher than those obtained when using the GOLDPERS (see rows PERS_1 in Table 4), which can partly be due to the inclusion of first names only into the present version of SRPNER. On the other hand, results obtained for the recognition of roles with GOLDPERS ($F_1 = 0.87$ for STUDENT and $F_1 = 0.86$ for DANAS) were higher than those presented in the same paper ($F_1 = 0.83$). The same goes for the recognition of last names only: the previous result was $F_1 = 0.78$, while the use of SRPNER on the gold standard yielded $F_1 = 0.86$ for STUDENT and $F_1 = 0.85$ for DANAS.

The capability to distinguish masculine and feminine names was compared to the results presented in (Krstev et al., 2015). The results obtained in presented experiments were lower for all NE types: masculine full $F_1 = 0.97$ vs. $F_1 = 0.90/0.94$ (STUDENT/DANAS), feminine full $F_1 = 0.94$ vs. $F_1 = 0.86/0.86$, masculine last $F_1 = 0.89$ vs. $F_1 = 0.85/0.83$, feminine last $F_1 = 0.79$ vs. $F_1 = 0.49/0.64$. One can presume that the use of gold standard produces more reliable results.

To the best of our knowledge, STANFORD NER and SPACY NER were used for the first time for the recognition of personal names in Serbian texts. Ljubešić et al. (2013) used STANFORD NER to build models for Croatian and Slovene. When they used distributional similarity to improve results, on texts coming from different sources they obtained the following results: for Croatian $P = 0.91$, $R = 0.93$ and $F_1 = 0.92$, higher than STANFORD NER for the model PERS_1, and for Slovene $P = 0.82$, $R = 0.87$ and $F_1 = 0.84$, comparable with STANFORD NER for the model PERS_1 (Table 6). One should note, however, that their test set contained a smaller number of personal names (approximately one third of number of personal names in our test sets).

Jiang et al. (2016) compared 4 NER systems, two of which were STANFORD NER and SPACY NER, for English. Their test set consisting of Wiki articles contained approximately the same number of personal names as our both sets – around 900. Their results were for STANFORD NER $P = 0.72$, $R = 0.87$ and $F_1 = 0.79$, and for SPACY NER $P = 0.73$, $R = 0.73$ and $F_1 = 0.73$. Our results are comparable in the sense that they also show that STANFORD NER achieves the best recall, while SPACY NER tends to have the more balanced precision and recall.

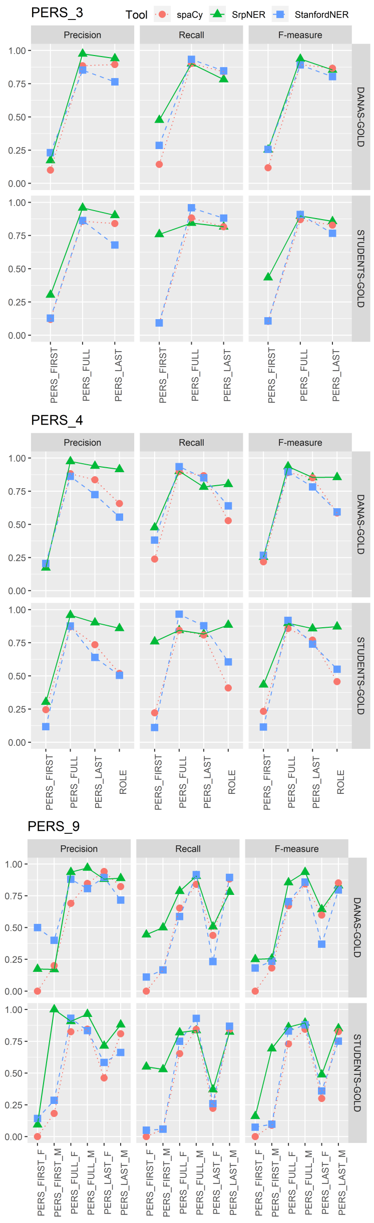


Figure 3: Evaluation of SRPNER, SPACY NER and STANFORD NER on two test sets, by each named entity type

6 Online Tool for NER

Serbian NER team (2019) offers an on-line tool for different purposes related to Named Entity Recognition. First, it supports conversion among different formats common for representations of NEs:

BRAT, CoNLL02 and XML. BRAT (Stenetorp et al., 2012) is a web-based tool[9] for text annotation, i.e., for adding notes to existing text documents. It is designed for structured annotation, allowing embedded annotations, which are especially convenient for NER. Annotations are external, so for each text file, an additional annotation file contains annotation data described in Section 4. CoNLL02 is a two-column format, also described in Section 4. An example of XML file whit tags interpreted as NEs is given in Table 3.

NER&BEYOND contains nine different modules:

XML ↦ BRAT module supports transformation of XML files which tags are interpreted as named entities, to BRAT format;

BRAT ↦ XML module supports transformation of files in BRAT format and their corresponding textual files to XML format;

BRAT ↦ CoNLL02 module supports transformation of files in BRAT format and their corresponding textual files to CoNLL02 format, using a Python script that is a part of BRAT package;

CoNLL02 ↦ BRAT module supports transformation of files in CoNLL02 format to BRAT format and their corresponding textual files, using a Python script that is a part of BRAT package;

XML ↦ CoNLL02 module supports transformation of XML files which tags are interpreted as named entities, to CoNLL02 format;

spaCy NER module provides NE annotation using spaCy (Honnibal and Montani, 2017), a free, open-source library for advanced NLP tasks in Python. This portal offers automatic annotation of texts in English, Spanish, German, Portuguese, French, Italian, Dutch and Serbian;

StanfordNER module provides Named Entity annotation using STANFORD NER models (Manning et al., 2014), which are available for Serbian, English and German with different levels of details, e.g. number of NE classes. Serbian model is developed withing presented research, while English and German are integrated from Stanford repository;

NER statistics module is developed for analysis of annotated text co llections in BRAT, that can be automatically downloaded via BRAT web interface. Various statistics related to distributions of named entities and attributes can be computed, including frequencies of annotated entities, classes, attributes per document and collection;

Gemini tool allows comparison of two text annotation files and provides different alignment scores. It is possible to compare a pair of XML files, a pair of files in BRAT for mat and one XML file against a file in BRAT format. The first file is the output of a NER system and the second file represents a gold standard.[10]

7 Future Work

For the upcoming research, we plan to apply the procedure we used for personal names to other NE classes (organization, location, event, temporal, quantitative, etc) and to experiment with other ML NER methods and tools with an ultimate goal to produce a successful hybrid system. The important next step is the enhancement of our newspaper corpus with other types of text (Wikipedia articles, domain texts, literary texts). The literary texts would be particularly important for improving the recognition of first names. Finally, another intended step is Entity Linking (EL), i.e. disambiguation of recognized named entities to a knowledge base, such as Wikidata, DBpedia WordNet and BabelNet. Such example would be automatically assigning Wikidata URL that points to a biography of a famous person to the corresponding named entity detected in text.

Acknowledgments

This research was supported by Serbian Ministry of Education and Science under the grants #III 47003 and 178006.

References

Yuheng Feng. 2018. Gemini - un module de comparaison de deux fichiers de textes annotés. Université Paris-Est Marne-la-Vallée, l'Institut Gaspard-Monge, https//github.com/fyh828/gemini.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested Named Entity Recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume*

[9] BRAT, https://brat.nlplab.org

[10] Gemini, https://github.com/fyh828/gemini/

1. Association for Computational Linguistics, pages 141–150.

Nathalie Friburger and Denis Maurel. 2004. Finite-state Transducer Cascades to Extract Named Entities in Texts. *Theoretical Computer Science* 313(1):93–104.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*. volume 1.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing. To appear, `https://pypi.org/project/spacy/2.0.16/`.

Martin Jansche and Steven Abney. 2002. Information Extraction from Voicemail Transcripts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Ridong Jiang, Rafael E Banchs, and Haizhou Li. 2016. Evaluating and Combining Name Entity Recognition Systems. In *Proceedings of the Sixth Named Entity Workshop, joint with 54th ACL*. pages 21–27.

Cvetana Krstev, Ivan Obradović, Miloš Utvić, and Duško Vitas. 2014. A System for Named Entity Recognition Based on Local Grammars. *Journal of Logic and Computation* 24(2):473–489. https://doi.org/10.1093/logcom/exs079.

Cvetana Krstev, Miloš Utvić, and Jelena Jaćimović. 2015. Ako koza laže, rog ne laže - gde su i ko su žene u srpskoj dnevnoj štampi [Facts Are Stubborn Things – Women in Serbian Daily Press]. *Knjiženstvo - časopis za studije književnosti, roda i kulture* V. https://doi.org/10.18485/knjiz.2015.1.24.

Cvetana Krstev, Duško Vitas, and Sandra Gucul. 2005. Recognition of Personal Names in Serbian Texts. In *International Conference Recent Advances in Natural Language Processing (RANLP'05)*. pages 288–292.

George R. Krupka and Kevin Hausman. 1998. Isoquest: Description of the netowl™ extractor system as used in MUC-7. In *Message Understanding Conference (MUC-7)*.

John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*. pages 282–289.

Nikola Ljubešić, Marija Stupar, Tereza Jurić, and Željko Agić. 2013. Combining Available Datasets for Building Named Entity Recognition Models of Croatian and Slovene. *Slovenščina* 2(1):35–57.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 55–60.

Denis Maurel, Nathalie Friburger, Jean-Yves Antoine, Iris Eshkol, and Damien Nouvel. 2011. Cascades de transducteurs autour de la reconnaissance des entités nommées. *Traitement automatique des langues* 52(1):69–96.

David Nadeau and Satoshi Sekine. 2007. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes* 30(1):3–26.

E.F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-independent Named Entity Recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended Named Entity Hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. European Language Resources Association (ELRA), Las Palmas, Canary Islands - Spain. http://www.lrec-conf.org/proceedings/lrec2002/pdf/120.pdf.

Serbian NER team. 2019. NER&Beyond. `http://nerbeyond.jerteh.rs/`.

Sameer Singh, Dustin Hillard, and Chris Leggetter. 2010. Minimally-supervised Extraction of Entities from Text Advertisements. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 73–81.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Junı́chi Tsujii. 2012. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations Session at EACL 2012*.

Duško Vitas and Cvetana Krstev. 2012. Processing of Corpora of Serbian using Electronic Dictionaries. *Prace Filologiczne* LXIII:279–292.

Moral Stance Recognition and Polarity Classification from Twitter and Elicited Text

Wesley Ramos dos Santos
University of São Paulo
São Paulo, Brazil
wesley.ramos.santos@usp.br

Ivandré Paraboni
University of São Paulo
São Paulo, Brazil
ivandre@usp.br

Abstract

We introduce a labelled corpus of stances about moral issues for the Brazilian Portuguese language, and present reference results for both the stance recognition and polarity classification tasks. The corpus is built from Twitter and further expanded with data elicited through crowd sourcing and labelled by their own authors. Put together, the corpus and reference results are expected to be taken as a baseline for further studies in the field of stance recognition and polarity classification from text.

1 Introduction

Computational sentiment analysis may be understood as a wide range of tasks intended to identify opinions, emotions and other types of stance expressed in natural language text (Tsytsarau and Palpanas, 2012; Liu, 2015). Among these, opinion mining is arguably the most well-studied form of sentiment analysis, consisting of identifying the target of an opinion, and/or the polarity (positive, negative, neutral etc.) of the sentiment expressed towards this target (Tsytsarau and Palpanas, 2012).

Stance recognition (Anand et al., 2011; Hasan and Ng, 2013; Lai et al., 2016; Mohammad et al., 2016b; Zarrella and Marsh, 2016; Wei et al., 2016; Mohammad et al., 2017), by contrast, consists of deciding whether the author of a piece of text shows a favourable or unfavourable attitude (or position) towards a certain target (Mohammad et al., 2017). The distinction between sentiment and stance is motivated by the observation that a sentiment, regardless of being positive or negative, may reflect a favourable or unfavourable position towards the target (Mohammad et al., 2016b). For instance, given the target topic 'veganism', a sentence as in 'beef tastes wonderful' expresses a positive feeling (which would indeed be recognised as such by traditional sentiment analysis systems), but it also reflects an unfavourable position towards this particular target.

Stance recognition from text is a well-known and yet challenging research topic. Systems of this kind enable the development of more complex sentiment analysis applications, and have been at the centre of a recent shared task (Mohammad et al., 2016b) focused on the use of supervised and unsupervised methods for stance recognition in the English language. For other less-resourced languages, however, resources remain scarce.

Based on these observations, this paper presents a labelled corpus of stances in Brazilian Portuguese, and a number of computational models addressing two related issues: *stance recognition*, is presently regarded as the binary classification problem of deciding whether a given text conveys *any* attitude towards a certain target topic or not, and *stance polarity classification*, which is regarded as the binary classification problem of deciding whether a given stance expressed as text shows a *positive or negative* attitude towards the target topic. Examples of both tasks for the target topic 'veganism' are as follows.

Stance recognition:

- *She says that avoiding animal products is just a fad (no stance towards veganism)*
- *Veganism will save the world! (a stance towards veganism)*

Stance polarity classification:

- *No one should ever eat beef (a positive stance towards veganism)*

Proceedings of Recent Advances in Natural Language Processing, pages 1069–1075,
Varna, Bulgaria, Sep 2–4 2019.

- *Vegans tend to have health issues (a negative stance towards veganism)*

As in the case of the English stance corpus in (Mohammad et al., 2016b), we will favour the recognition of stances about moral issues (e.g., abortion, drugs legislation etc.) in the Twitter domain. In addition to that, however, Twitter data will be presently expanded with a collection of moral stances elicited through crowd sourcing as well, and which were labelled by their own authors as a gold standard. Put together, the corpus and its reference results are expected to be taken as a baseline for further studies in moral stance recognition and stance polarity classification tasks.

2 Related Work

The work in (Anand et al., 2011) is among the first to address the computational recognition of stances from text, analysing a corpus of 4873 posts in on-line discussion forums. The data set considered covers 14 topics, ranging from entertainment to ideological issues. Favourable and unfavourable stances are recognised with accuracy of up to 69%, outperforming a unigram baseline model that obtained up to 60% accuracy.

Stance recognition in discussion forums is also addressed in (Hasan and Ng, 2013). In this case, however, the work focuses on the question of how the performance of a stance classifier varies in relation to the volume and quality of training data, regarding the complexity of the underlying model, the richness of the set of learning features and the use of extra-linguistic restrictions in a wide range of scenarios. The experiments leave a series of contributions on how to build models of this type, and on which kinds of knowledge to consider.

More recently, the SemEval-2016 competition (Mohammad et al., 2016a) brought together 19 participating systems engaged in the task of supervised stance recognition from tweets in the English language. The training corpus, described in detail in (Mohammad et al., 2016a), contains 2914 tweets about five target topics (atheism, climate change, feminism, Hillary Clinton and abortion legislation.) The corpus contains, on average, 583 tweets per target, but the set is unbalanced. On average, there are 25,8% positive and 47,9% negative stances. The test set, with 1249 tweets, is even more unbalanced, with 24,3% of positive stances and 57,3% negative stances. The SemEval corpus is the basis of some of studies discussed as follows.

The work in (Zarrella and Marsh, 2016) presents the best overall performance in the SemEval-2016 shared task (Mohammad et al., 2016b) on supervised stance recognition. The proposal makes use of a recurrent neural network with features learned by distant supervision from large unlabelled datasets. Word and phrase embedding models are trained using Word2Vec skip-gram (Mikolov et al., 2013), and then used for learning sentence representations with the aid of a hashtag prediction model. Finally, sentence vectors are optimised for stance recognition based on the labelled examples from the training corpus.

Also in the context of the SemEval-2016, the work in (Wei et al., 2016) presents an approach based on convolutional neural networks that, instead of simply predicting when the validation accuracy will reach its maximum, uses a voting scheme and other secondary improvements. The model is trained individually for each of the five targets of the SemEval-2016 corpus, and obtains the second best overall results for the supervised stance recognition track.

Subsequent to SemEval-2016, a number of improved systems have been proposed. The work in (Lai et al., 2016), for instance, explores the use of world knowledge - in the form of rules about friendships and political enmities - to enhance the task of recognising political stances in the SemEval-2016 corpus. The proposal consists of a stance recognition model enriched with semantic features of each target topic, which outperforms the participant systems of the original shared task.

Finally, the work in (Mohammad et al., 2017) presents a post-hoc evaluation of the SemEval-2016 stance recognition task, proposing a much simpler and more accurate model than the overall winner of the competition in (Zarrella and Marsh, 2016). The proposed model makes use of linear SVM and a set of features computed from the training data, such as word and character n-grams and word embeddings computed from an additional data set.

3 Current Work

The present investigation of moral stance recognition and polarity classification consists of a corpus data collection (described in Section 3.1), and two individual experiments: stance recognition (Section 3.2) and stance polarity classification (Section 3.3). In both cases, we shall focus on methods that rely on lexical and morphological knowledge only by making use of word and char n-grams.

3.1 Corpora

Our initial goal was to create a corpus of moral stances in the Brazilian Portuguese language that would preferably be (a) at least as large as the English training dataset for SemEval-2016 supervised stance recognition task (Mohammad et al., 2016b), (b) more well-balanced if possible, and (c) not limited to the Twitter domain. To this end, we collected a 180k-word corpus conveying over 5,000 moral stances from two sources - Twitter and stances elicited through crowd sourcing - about five topics: abortion, death penalty, drug legalisation, criminal age, and racial quotas. Elicited texts are, on average, 3.5 times longer than tweets.

Corpus descriptive statistics for our two domains are summarised in Table 1.

Twitter messages were collected by searching Brazil Twitter for specific key words (e.g., 'abortion' etc.). For each topic, an initial 7000-message set was selected for manual inspection and labelling.

Elicited stances were obtained from a crowd sourcing task involving 490 Brazilian Portuguese native speakers. Participants were requested to give their opinions about each of the target topics by providing answers in a 0 (totally disagree) to 5 (totally agree) scale and, subsequently, were requested to provide motivation for each of their opinions by writing a short text. The elicited corpus has been subject to spell-checking and it is overall much more well-formed than the Twitter data, and with a larger vocabulary.

Twitter messages were manually labelled by assigning a positive/negative class to all messages that unequivocally expressed a stance on the intended topic, and by assigning the class 'other' to any message that did not meet these criteria. Thus, the class 'other' represents the fact that, despite containing a key word of interest, the message did not convey any obvious stance about the target topic, and it was therefore regarded as noise[1].

Twitter text labelling proceeded until a minimum of 240 instances of each of the three class were identified, or until the end of the dataset was reached. This allowed us to obtain a certain balance between for/against stances for most topics, but resulted in a vast majority of samples labelled as 'other'. Thus, the 'other' class - which corresponds to non-stance text - is several times larger than the positive and negative classes in all five topics.

Elicited stances, by contrast, were assigned labels automatically based on the opinion scores provided by the crowd sourced participants. More specifically, scores 0 and 1 were taken as representing negative stances, 2 and 3 as neutral stances, and 4 and 5 as positive stances. Unlike the Twitter dataset, we notice that all elicited texts contain, by definition, some stance on the topics under discussion, and hence there is no 'other' (or non-stance) class in this domain.

Class label distributions for the Twitter and elicited datasets are summarised in Table 2.

3.2 Stance Recognition

Our first experiment - stance recognition - is presently defined as the binary classification problem of deciding whether a given text conveys any attitude towards a certain target topic or not. Since all texts from our elicited data (cf. the previous section) express, by definition, some stance about the target topic, the present task is applicable to Twitter data only.

3.2.1 Models

For the stance recognition task, a range of n-gram models - from 1 to 5 words and from 3 to 16 characters - was considered, and we found that character-based models always outperform word-based models. As a result, all models under consideration for this task are based on character n-grams only.

In what follows we consider the use of TF-IDF character counts (here by called our *Select.char* model) with k-best univariate feature selection using ANOVA F1 as a score function. By combining relatively long character sequences (which in most cases encompass words) with feature selection, we expect *Select.char* to outperform the alternatives under consideration, as discussed below.

The *Select.char* model was trained by making use of the best out of three possible learning methods - Naive Bayes, Logistic Regression and Mul-

[1] For instance, annotators came across a number of references to 'Aborto Elétrico' (electrical abortion), which is the name of a rock band with no relation to any stance about the target topic.

Table 1: Corpus descriptive statistics

Source	vocab. size	words	messages	words / msgs
Twitter	5,789	44,564	2,792	16
Elicited	9,081	137,122	2,450	56
Overall	11,845	181,686	5,242	35

Table 2: Class distribution for Twitter and elicited data.

	Twitter stances			Elicited stances		
Topic	for	against	other	for	neutral	against
Abortion	240	384	2570	310	105	75
Death penalty	801	244	1518	105	125	260
Drugs	335	181	1482	263	129	98
Criminal age	243	240	1433	198	104	188
Racial quotas	240	364	2596	205	128	157
Overall	1859	1413	9599	1081	591	778

tilayer perceptron, with optimal k-values selected in the 5000 to 90000 range at 1000 intervals by performing grid search on the training dataset.

In addition to that, the entire input feature set (i.e., with no feature selection) is taken as the basis for two simpler methods - logistic regression (*LogReg.char*) and multilayer perceptron (*MLP.char*). The latter consists of 3 layers conveying 150 neurons each, and using rectified linear units (ReLU) as an activation function.

The three models of interest - *Select.char*, *LogReg.char* and *MLP.char* are to be evaluated against a majority class baseline *Majority*.

3.2.2 Data

The experiment makes use of the Twitter dataset described in Section 3.1 with random 80:20 train-test split.

3.2.3 Evaluation

Table 3 shows F1 results of stance recognition on Twitter data for both positive (stance) and negative (others, or non-stance) classes, and overall weighted F1 scores obtained by each model under consideration. Best weighted F1 scores for each target topic are highlighted.

As expected, all models easily outperform the *Majority* baseline, and the combination of char n-grams and feature selection in *Select.char* generally outperforms the alternatives under consideration, albeit for a small difference. This may be partially explained by the heavy data imbalance (cf. Table 2), which may have obscured possible differences across models. Moreover, we notice that variation across target topics is also small, suggesting that stance recognition is relatively topic-independent.

3.3 Stance Polarity Classification

Our second experiment - polarity classification - is presently defined as the binary classification problem of deciding whether a given stance expressed as text shows a positive or negative attitude towards the target topic. For this task we consider both elicited stances, and also the portion of Twitter data that conveys a positive or negative stance, that is, disregarding only those tweets labelled as 'other' (cf. section 3.1.)

3.3.1 Models

Given the overall positive results of character-based models and feature selection in the case of stance recognition (cf. the previous section), we will consider models of this kind for stance polarity classification as well. To this end, we make use of a char n-gram model - hereby called *MLP.char* that is similar to *Select.char* in the previous section, except that in the present case we will focus on the use of MLP classifiers only.

In addition to *MLP.char*, we also consider a mode based on skip-gram word embeddings (Mikolov et al., 2013), hereby called *MLP.w2vec*. The model makes use k-best univariate feature selection with the ANOVA F1 function over TFIDF-weighted word embeddings of size 50, 100 and 300. Learning methods under considerations are

Table 3: Weighted average F1 results for Twitter stance recognition

	Majority			LogReg.char			MLP.char			Select.char		
Topic	stance	other	avg	stance	other	avg	stance	other	avg	stance	other	avg
Abortion	0.00	0.89	0.71	0.40	0.85	0.76	0.43	0.88	**0.79**	0.42	0.89	**0.79**
Death penalty	0.74	0.00	0.44	0.62	0.74	0.69	0.65	0.78	0.73	0.72	0.81	**0.78**
Drugs	0.84	0.00	0.61	0.43	0.76	0.67	0.41	0.82	0.71	0.41	0.84	**0.72**
Criminal age	0.00	0.85	0.64	0.45	0.80	0.71	0.55	0.84	**0.76**	0.53	0.84	**0.76**
Racial quotas	0.00	0.89	0.70	0.36	0.85	0.75	0.36	0.86	0.76	0.33	0.89	**0.77**
Overall	0.32	0.53	0.62	0.45	0.80	0.72	0.48	0.84	0.75	0.48	0.85	**0.76**

MLP classifiers of 1-3 layers with numbers of neurons ranging from 33 up to the size of the embedding vector, and using either ReLU or hyperbolic tangent (tanh) as an activation function. Optimal parameters and vector sizes were determined by performing grid search on the training data.

Finally, given the affective nature of the topics in the corpus, we will also consider the use of psycholinguistic knowledge as provided by the LIWC dictionary (Pennebaker et al., 2001). LIWC models word categories such as love, money, power etc. that are known to play a significant role in a range of NLP tasks such as sentiment analysis and, in particular, personality recognition from text. Psycholinguistic knowledge will hence be the basis of a simple model - hereby called *LIWC* - consisting of a 64-feature subset of LIWC category counts for Brazilian Portuguese (Filho et al., 2013).

The three models of interest - *MLP.char*, *MLP.w2vec* and *LIWC* - are to be evaluated against two baseline systems: a majority class baseline *Majority*, and a word-based TFIDF model with k-best feature selection - hereby called *LogReg.word*, in both cases making use of logistic regression.

3.3.2 Data

The experiment makes use of the elicited stance dataset, and also the stance portion of the Twitter dataset as described in Section 3.1. In both cases, a random 80:20 train-test split was performed.

3.3.3 Evaluation

Table 4 shows weighted F1 score results obtained by each model under consideration for the Twitter domain, and Table 5 shows results for the elicited data. In both cases, best weighted F1 scores for each target topic are highlighted.

Although all Twitter models outperform the *Majority* baseline, results are overall mixed and, for two topics (death penalty and criminal age), the word-based baseline model *LogReg.word* actually outperforms the alternatives. Moreover, we notice that the psycholinguistics-based *LIWC* approach produces the second lowest results of all, and that none of the top-performing models seems clearly superior to the others. We hypothesise that the close results obtained by *LogReg.word*, *MLP.w2vec* and *MLP.char* may be partially explained by the use of the same underlying feature selection method, which turned out to be more significant than the actual choice of text representation or learning method.

Contrary to the Twitter scenario, results for the elicited data were uniform, with the *MLP.char* approach outperforming all alternatives by a large margin, and once again leaving the *Majority* baseline and *LIWC* models at the bottom. We hypothesise that, as in the case of the stance recognition experiment in the previous Section 3.2, the use of long char n-gram sequences does help the present task as well, and that it may have been particularly successful in combination with the higher text quality of elicited stances in our data.

Finally, a note on the use of char n-grams. As expected, the k-best n-grams in the models that use feature selection largely correspond to single words (e.g., ‘unacceptable’) or short expressions (e.g., ‘I agree’), both of which clearly denoting stances in our domains. As a result, the model is comparable to a variable-length word n-grams, but with greater flexibility to include subwords (e.g., ‘believ*’). To illustrate this, Figure 1 shows the char n-grams distribution among the k-best terms in the polarity classification task from the elicited dataset. From these results, we notice that the selected char n-grams largely fall within the 4..10 range, peaking at n-grams with a length of 6.

Table 4: Weighted average F1 results for polarity classification from Twitter data

Topic	Majority	LogReg.word	LIWC	MLP.w2vec	MLP.char
Abortion	0.41	0.68	0.61	0.66	**0.71**
Death penalty	0.53	**0.82**	0.67	0.81	0.77
Drugs	0.55	0.60	0.60	**0.77**	0.66
Criminal age	0.29	**0.82**	0.71	0.75	0.76
Racial quotas	0.49	0.73	0.70	**0.76**	**0.76**
Overall	0.45	0.73	0.66	**0.75**	0.73

Table 5: Weighted average F1 results for polarity classification from elicited data

Topic	Majority	LogReg.word	LIWC	MLP.w2vec	MLP.char
Abortion	0.49	0.69	0.52	0.76	**0.92**
Death penalty	0.37	0.53	0.43	0.72	**0.90**
Drugs	0.37	0.48	0.50	0.57	**0.82**
Criminal age	0.23	0.64	0.47	0.67	**0.87**
Racial quotas	0.25	0.50	0.42	0.59	**0.77**
Overall	0.34	0.57	0.47	0.66	**0.84**

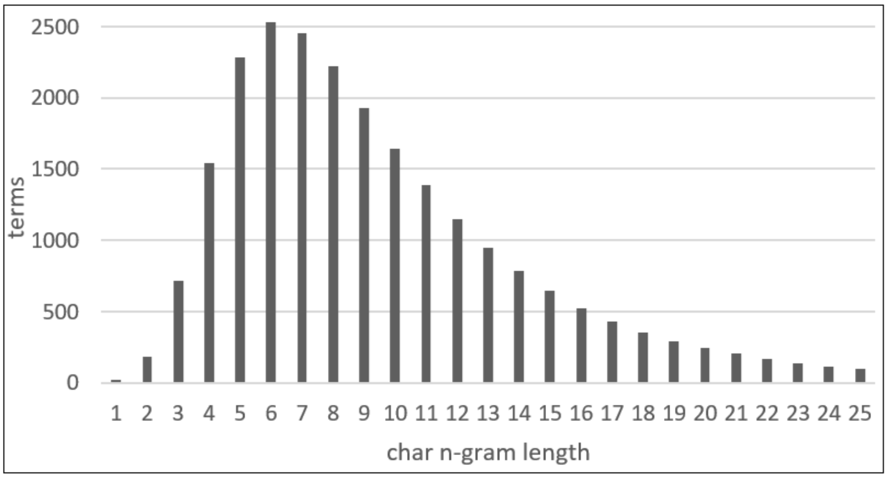


Figure 1: Char n-gram distribution across k-best terms in polarity classification from elicited data.

4 Final Remarks

This paper addressed the issue of moral stance recognition from text. We introduced a labelled corpus of stances taken from Twitter and additional crowd-sourced texts, and a number of supervised models of stance recognition and stance polarity classification.

Initial results suggest that both tasks may be performed with relatively high accuracy by making use of simple models based on char n-grams and feature selection. As expected, best results were observed when using more well-formed (in our case, crowd-sourced) texts, rather than when using Twitter data.

The corpus and the present results are expected to be taken as a reference for further studies in moral stance recognition in Brazilian Portuguese natural language processing. As future work, we intend to expand the current dataset in both domains by adding more instances and topics, and assess the use of deep learning methods for both the stance recognition and the polarity classification tasks.

Acknowledgements

This work received support by FAPESP grants # 2017/06828-1 and 2016/14223-0, and from the University of São Paulo.

References

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (ACL-HLT 2011)*. Association for Computational Linguistics, Portland, Oregon, USA, pages 1–9.

Pedro P. Balage Filho, Sandra M. Aluísio, and T.A.S. Pardo. 2013. An evaluation of the Brazilian Portuguese LIWC dictionary for sentiment analysis. In *9th Brazilian Symposium in Information and Human Language Technology - STIL*. Fortaleza, Brazil, pages 215–219.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Nagoya, Japan, pages 1348–1356.

Mirko Lai, Delia Irazu Hernandez Farias, Viviana Patti, and Paolo Rosso. 2016. Friends and Enemies of Clinton and Trump: Using Context for Detecting Stance in Political Tweets. In *Mexican International Conference on Artificial Intelligence (MICAI-2016). Lecture Notes in Computer Science, vol 10061*. Springer.

Bing Liu. 2015. *Sentiment analysis: mining opinions, sentiments, and emotions*. Cambridge University Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016a. A dataset for detecting stance in tweets. In *Proceedings of 10th edition of the the Language Resources and Evaluation Conference (LREC-2016)*. Portoroz, Slovenia.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016b. Semeval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of the International Workshop on Semantic Evaluation*. San Diego, California,USA.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *Special Section of the ACM Transactions on Internet Technology on Argumentation in Social Media* 17(3).

J. W. Pennebaker, M. E. Francis, and R. J. Booth. 2001. *Inquiry and Word Count: LIWC*. Lawrence Erlbaum, Mahwah, NJ.

Mikalai Tsytsarau and Themis Palpanas. 2012. Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery* 24(3):478–514.

Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. 2016. pkudblab at SemEval-2016 Task 6: A Specific Convolutional Neural Network System for Effective Stance Detection. In *Proceedings of the International Workshop on Semantic Evaluation*. San Diego, California,USA.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection. In *Proceedings of the International Workshop on Semantic Evaluation*. San Diego, California,USA.

The “Jump and Stay” Method to Discover Proper Verb Centered Constructions in Corpus Lattices

Bálint Sass
Research Institute for Linguistics, Hungarian Academy of Sciences
Budapest, Hungary
sass.balint@nytud.mta.hu

Abstract

The research presented here is based on the theoretical model of corpus lattices. We implemented this as an effective data structure, and developed an algorithm based on this structure to discover essential verbal expressions from corpus data. The idea behind the algorithm is the “jump and stay” principle, which tells us that our target expressions will be found at such places in the lattice where the value of a suitably defined function (whose domain is the vertex set of the corpus lattice) significantly increases (jumps) and then remains the same (stays). We evaluated our method on Hungarian data. Evaluation shows that about 75% of the obtained expressions are correct, actual errors are rare. Thus, this paper is 1. a proof of concept concerning the corpus lattice model, opening the way to investigate this structure further through our implementation; and 2. a proof of concept of the “jump and stay” idea and the algorithm itself, opening the way to apply it further, e.g. for other languages.

1 Introduction

In this paper we present a novel, original verbal construction discovery method. Our starting point will be our former paper (Sass, 2018) which describes a theoretical model considered an appropriate basis for extracting so-called *proper verb centered constructions* from analysed corpora.

First, let us look at our target. In this terminology, verb centered constructions (VCC) are verb + slot structures, where slots can be unfilled (free) or filled (by a filler word). In English, subject (SBJ), direct object (OBJ) and all prepositions can be considered as slots. For example, ‘*take* + *SBJ* + *OBJ* + *into:account*’ has two free slots (SBJ and OBJ) and a filled ‘*into*’ slot where the filler (marked by a colon) is the word ‘*account*’. In this approach, a filler is the head of the phrase realizes the slot. Length of a VCC (l) is defined as number of slots and fillers added up, the above example has a length of 4.

Then, what are *proper* verb centered constructions (pVCC)? They are complete and clean. That means they contain all necessary elements, and does not contain any unnecessary element for expressing the core meaning of the verbal expression in question. For example, ‘*take* + *SBJ* + *OBJ:part* + *in*’ is proper, while ‘*take* + *SBJ* + *OBJ:part*’ is not proper (because not complete), and ‘*read* + *SBJ* + *OBJ*’ is proper, while ‘*read* + *SBJ* + *OBJ:book*’ is not proper (because not clean). In other words, free slots in pVCCs are complements (subject included) and fillers in pVCCs are idiomatic, carrying some special meaning.

It is clear that pVCCs are constructions. They are form–meaning pairs (Goldberg, 2006; Kay and Michaelis, 2015), they are units of meaning (Teubert, 2005; Danielsson, 2007). Their meaning is assigned to the whole form, they cannot be divided into smaller units if we want to keep the original meaning.

On the other hand, pVCCs are not necessarily multiword. They are multiword in most cases as we have seen in the examples, but there are cases when the multiword property is not satisfied in the strict sense that they consists of two or more whole *words*. Consider for example ‘*read* + *SBJ* + *OBJ*’, it consists of three elements, from which only one is a word, the other two are just slots. Or consider a Hungarian example. In this language slots are specified mostly by bound morphemes, namely case markers. The Hungarian counterpart of ‘*believe* + *SBJ* + *in*’ is ‘*hisz* + *NOM* + *INE*’

Proceedings of Recent Advances in Natural Language Processing, pages 1076–1084,
Varna, Bulgaria, Sep 2–4 2019.

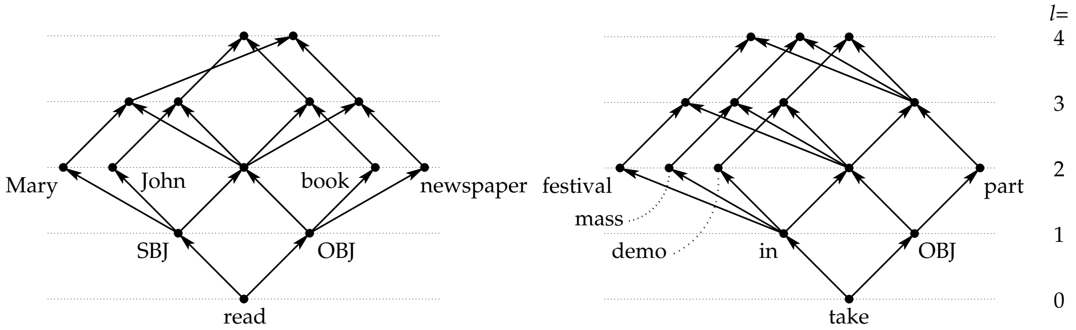


Figure 1: This pair of figures illustrate the notion of double cube and the notion of corpus lattice. On the left, the double cube of '*John reads a book*' ('*read SBJ:John OBJ:book*') and the double cube of '*Mary reads a newspaper*' ('*read SBJ:Mary OBJ:newspaper*') are combined together to create a small corpus lattice. On the right, three clauses are combined. This figure presents two dimensional structures. The subject slot is not depicted in the latter case, it would require three dimensional double cubes.

respectively, where '*NOM*' is for nominative (subject) case, and '*INE*' is for inessive case (appearing as a '*-ban/-ben*' suffix). The English version is strictly multiword, while the Hungarian is not.

We saw that a pVCC should be complete. '*take part*' is a MWE, but '*take + SBJ + OBJ:part + in*' is a pVCC of full value containing all necessary elements. According to Siepmann (2005, page 416) „collocation and verb complementation are intimately related . . . a two-word combination cannot possibly be viewed as a fully-fledged collocation." Simply put, the free slots are just as important as the words/fillers. They turn MWEs into real constructions.

This concept of completeness is essential and unique here. We barely see it neither in classical papers, nor in recent works. Formerly, many papers dealt with just e.g. verb+noun expressions (Evert and Krenn, 2001; Fazly and Stevenson, 2006; Iñurrieta et al., 2016), but recently, also, even the definition of verbal MWEs does not explicitly include the preposition or case marker constituting a complement that, we are convinced, is an inherent part of the expression (Ramisch et al., 2018; Walsh et al., 2018). It is maybe the dependency annotation itself which does not support the approach presented here as case markers are usually not taken as separate units (see Simkó et al., 2017, Fig. 4). Because of the above, we can evaulate our method in itself only.

pVCCs are a large and key group of verbal expressions: they bear the different meanings and usage patterns linked to verbs. We think that it is a good idea if a dictionary presents exactly the set of pVCCs concerning a verb. It is not crucial that they are formally multiword or not. To cover all patterns, we should handle MWEs and constructions uniformly, in one framework. The method presented here shares this attitude.

2 The Initial Model and the Conjecture

Let us summarize the initial model here. The basic processing unit is the clause, the unit which contains a verb together with its complements and adjuncts, and consequently, a pVCC. So as preprocessing, clause boundary detection and some shallow parsing is needed on the input corpus to determine the verb and the top level slots and fillers.

Corpus clauses are represented as so-called *double cubes* (Sass, 2018, Fig. 3), which are a kind of mathematical lattice structures. The verb is at the bottom, every edge adds a slot or a filler to an existing slot (chosen from the clause in question). Vertices are VCCs, they represent nested VCCs of the clause with slots and fillers present or not in all variations. The top represents the original clause: all slots are there and filled as in the original clause. One distinguished vertex is the pVCC.

In the next step, the so-called *corpus lattice (CL)* is created from double cubes containing the same verb. Using a kind of lattice combination operation, double cubes are projected onto each other in a way that where they are identical they will overlap, where they are different they will split up (Fig. 1). It is important that, at the end, the evolving large semilattice structure will represent all clauses of a given verb, and also the distribution of all free and filled slots occurring beside this verb.

The initial paper contains only vague conjectures about how to actually apply the corpus lattice

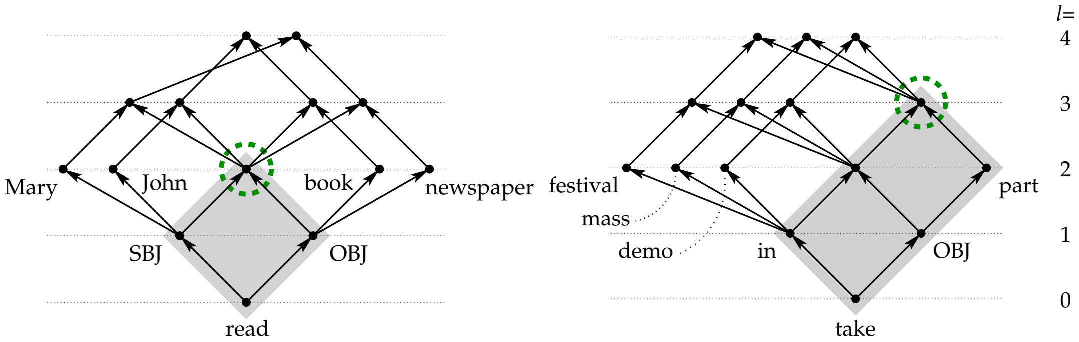


Figure 2: These are the same CLs as in Fig. 1, but here the pVCCs are marked: they are circled. The pVCC is '*read + SBJ + OBJ*' and '*take + SBJ + OBJ:part + in*' in the two figures respectively. f values are also depicted: $f > 1$ in the gray areas ($f = 2$ in the left lattice and $f = 3$ in the right one), at the other vertices $f = 1$.

structure for discovering pVCCs. We thought, the fact that the corpus lattice contains all information about the distribution of slots and fillers makes it a suitable "representation which can be a basis" for finding pVCCs. We introduced a function (we call it f now) on vertices of the corpus lattice: it is essentially the corpus frequency of the VCC represented by the given vertex. In other words, this f shows how many corpus clauses are represented by the given vertex, or how many corpus clauses this VCC fits to. We formulated the conjecture that pVCCs should be "at some kind of thickening points of the corpus lattice", and added that a future algorithm would move through the corpus lattice somehow systematically to find them.

3 The Idea of "Jump and Stay"

The model described above was purely theoretical. Our current contribution are implementation of the data structure, elaboration and implementation of an algorithm for discovering pVCCs using this data structure, and evaluation of the algorithm on real data.

For outlining our idea which leads to the algorithm, let us take a look at the already known figure from another perspective (Fig. 2). The CLs in the figure are for demonstration purposes: they are, of course, very small, but suitable for presenting the main point (cf. a real CL can be very wide (≈ how many words are there in the corpus), but not too tall (2 × how many slots are in the longest clause)). We mention that f always grows monotonically downwards in a CL.

Looking at Fig. 2, how can vertices representing pVCCs be characterized? Firstly, as we go top-down in the CL, f suddenly increases at certain points. Secondly, of these vertices, we should prefer those which are located higher in the corpus lattice. As it may be suspected, the first observation will be the basis of "jump" and second one will be the basis of "stay".

The principle of "jump and stay" can be formulated as follows: *jump* means that we advance from a vertex to an adjacent one downwards in the CL if f substantially increases, and *stay* means that we advance from a vertex to an adjacent one upwards in the CL if f remains more or less the same. (Please note, that *stay* also means advancing between vertices, the term itself refers to the fact that the value of f does not change during this step.) In other words, where one (or only few) arrows originate from a vertex, it tend to be a place of stay, similarly, when we have many arrows from the same vertex, it is usually a place of jump. Notice that if we apply the two rules of the principle starting from any of the vertices in Fig. 2, we end up at the circled pVCC. This is the point, this is the main idea of this paper itself.

If we look a bit more closely, in fact, we can end up at one of the top vertices depending on the application order of the two rules, at least in case of the CL on the left. As we will see, top vertices (fully filled clauses) will be excluded from being a pVCC.

pVCCs can be considered as some kind of thickening points indeed, where many edges converge (see the left lattice in Fig. 2), but stating the principle of "jump and stay" is much more clearer.

In addition, we have an independent argument in support of our idea. The "jump and stay" idea

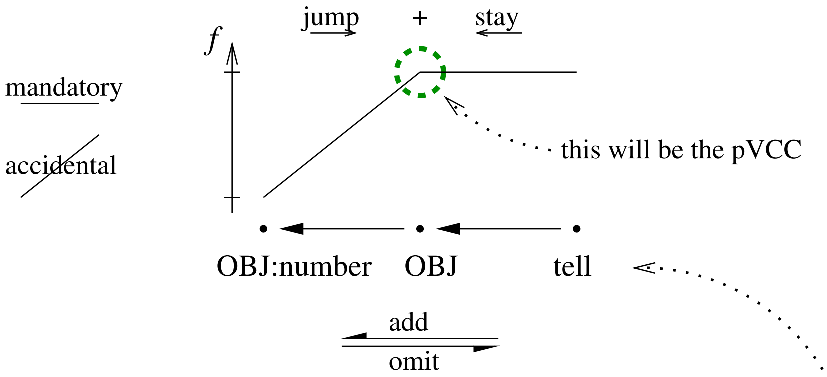


Figure 3: A summing-up of the "jump and stay" principle. A three-vertex piece of the CL of '*tell*' is shown together with the f values for each vertex. It is clear, that this verb requires a direct object, but the direct object itself can be several different words from which '*number*' is quite rare. In this case, there is a stay from '*tell*' to '*tell + OBJ*', and a jump from '*tell + OBJ:number*' to '*tell + OBJ*' so the pVCC is '*tell + OBJ*' here, and that is correct. At the bottom, the directions to add/omit an element to a VCC are shown. On the left, it is shown how a part of the graph of the f function basically look like in the case of mandatory/accidental elements.

is nicely consistent with the fact that constructions have mandatory and accidental elements, some elements are necessary while some are not, as it is also reflected in the definition of the pVCC.

When we jump, we try to omit something which is not mandatory. A typical case of jump, when there are several different fillers (e.g. the various foods as direct object of '*eat*', which are obviously relatively rarer one by one) in a given slot, and the lower vertex is the one, where this slot is free. Advancing to this vertex, we omit this diversity of fillers, we omit something that does not seem to be mandatory (cf. the vertex marked with '*in*' and the three arrows originating from it pointing to the left in Fig. 2).

When we stay, we try to add something which is mandatory. A typical case of stay, when we add an element (a slot or a filler to an existing slot) to a vertex/VCC, but we still cover roughly the same amount of original corpus clauses. This shows, that the added element is mandatory, namely it occurs in nearly all clauses represented by the original VCC. (cf. the vertex marked with '*in*' and the arrows above it in the gray area in Fig. 2).

We always advance by jump downwards (and by stay upwards) in a CL, the other way round – discarding what is needed and adding what is not needed – would not make much sense.

Yet another argument supporting our idea. As we investigate the structure of different CLs, it turns out that a typical pVCC is an endpoint of both jumps and stays, or to put it another way no jump and no stay originate from it (Fig. 2). Plain (not proper) VCCs, however, does not have this property. Almost always, they are a starting point of a jump or a stay (see '*take + OBJ:part + in:festival*' or '*take + OBJ:part*' in Fig. 2 again).

To end this section, look over the "jump and stay" principle in a summary figure (Fig. 3).

Recall the definition of pVCC: stays increase completeness, and jumps increase cleanness. We think that those vertices have the most chance to be a pVCC which can be reached by a stay from below and by a jump from above at the same time.

4 Implementation of the Data Structure

The corpus lattice is a special kind of graph structure. What crucially important to use it effectively is to be able to effectively advance from one vertex to another connected by an edge. For this purpose, we store vertices and edges in hashes (dictionaries) in our python implementation. Edges are stored firstly in one direction, and secondly in the other direction separately.

Starting from a language resource consisting of clauses represented in the form of verb + slots + fillers we build a CL (for each verb separately) as follows:

1. We go through the corpus and take clauses one by one.
2. We build the appropriate double cube of the given clause: starting from the fully filled

clause, we add adjacent edges and vertices omitting slots/fillers one by one recursively.

3. Our graph data structure for a double cube and for a CL both will be the following: we store vertices in a hash, and edges in a hash of hashes (as we have said, in each direction separately). The key of the hashes is a "canonical" JSON string form of the given VCC, its slots ordered alphabetically by slot names.
4. Finally, we combine the current double cube to the corpus lattice being built recording and updating the appropriate f values.

This way we obtain a quite effective representation of a CL.

Input data should be in a specific JSON format which can be generated from either a (shallow) dependency or a (shallow) constituency parse of the input corpus: the verb, the slots and the fillers need to be identified. It is similar to "top level syntactic sequence of the constituent tree" (Shi et al., 2016), with the difference that the order of constituents is not taken into account in our approach.

We used Hungarian data (Sass, 2015) for our experiments. This dataset contains 28 million clauses in a format which was not complicated to convert to the needed input format.

5 The Algorithm

In this section we describe how we implemented the "jump and stay" principle (section 3) using the data structure presented above (section 4).

At the beginning of work, we separated about 7 percent of the data for development purposes. That means, during developing the algorithm we used data only from this part. Developing the algorithm is a kind of learning phase, we draw conclusions based on the input data. It is very important not to use test data for this.

The algorithm consists of the following steps:

1. We go through each vertices of the CL. (The order does not matter, but we chose to begin with the bottom, and continue upwards as pVCCs tend to occur not too far from the bottom.)
2. Some kind of vertices are omitted early on: which are too long (has a length more than 8 ($l > 8$)), which are too rare (has $f < 3$), and which have no out-edge (that means which is at the top of the CL).
3. Firstly, we look for a stay, i.e. try to add a needed element. If the ratio of $f(\text{actual})/f(\text{above}) < 1.7$, then we consider this a stay, and advance to the vertex above. In case of several stays we choose the one with the smallest ratio.
4. Secondly, when no stay can be found, we look for a jump, i.e. try to discard an element which is not needed. If the ratio of $f(\text{below})/f(\text{actual}) > 4$, then we consider this a jump, and advance to the vertex below. In case of several jumps we choose the one with the largest ratio.
5. If we get to a new vertex, we repeat steps 3. and 4.
6. If neither a stay nor a jump can be found, we stop, and if the current VCC is not at the top of the CL (that means it has out-edges) then it is tagged as a pVCC.

Dealing with Hungarian data, at the beginning we do a modification based on Hungarian verb conjugation. Some Hungarian verb suffixes imply that the verb is transitive even when the direct object is not present in the clause. In such cases we add a free OBJ slot. Besides that, Hungarian being a pro-drop language we add a free SBJ slot to every clause without an explicit subject.

A small addition to step 4. In fact, we do not do a jump in every case. If the jump would omit the *last* filler from a VCC, we do not take this step. That is because specific fillers are usually important parts of a pVCC, and full-free VCCs would usually be frequent enough to swallow all pVCCs (being longer only by one filler) performing a jump. So we do the jump only if there remains at least one filler in the resulting VCC or there is no filler in the initial VCC already.

Threshold values (exactly 1.7 for stays, and 4 for jumps) are manually tested and set values. They gave the best results after some experimenting on the development corpus.

Fig. 4 shows some specific examples on how exactly our algorithm works in practice.

The source code of the algorithm and also for building and handling the CL data structure, together with some sample data, is available at `https://github.com/sassbalint/double-cube-jump-and-stay`. The algorithm is fast enough. Building a 365000 vertex CL and investigate it for pVCCs took 63 seconds in total on our server.

```
#4                                                  f= l=
["FAC", null]                                      309  1
 Processing.
 A stay found, we follow.
["FAC", null, "NOM", null]                         309  2
 A stay found, we follow.
["FAC", "jó", "NOM", null]                         307  3
 A stay found, we follow.
["ACC", null, "FAC", "jó", "NOM", null]            300  4
 No stay (ratio=5.17 > 1.7), we stop.
 No appropriate jump (keeping a filler, 1.02 < 4), we stop.
["ACC", null, "FAC", "jó", "NOM", null]            300  4  pVCC

#22699                                              f= l=
["ACC", "költségvetés", "FAC", "jó", "NOM", null]    4  5
 Processing.
 No stay (ratio=2.00 > 1.7), we stop.
 An appropriate jump (keeping a filler, 4<) found, we follow.
["ACC", null, "FAC", "jó", "NOM", null]            300  4
 No stay (ratio=5.17 > 1.7), we stop.
 No appropriate jump (keeping a filler, 1.02 < 4), we stop.
["ACC", null, "FAC", "jó", "NOM", null]            300  4  pVCC
```

Figure 4: Two examples from the output of the algorithm. This demonstrates how the algorithm works: it starts from a vertex and after some jumps and stays it finds the appropriate pVCC in the end. In the first example, we start from a one-free-slot VCC, and get to the pVCC through three stays, adding three mandatory elements (in bold), while the f value decreases from 309 only to 300. In the second example, we start from a longer VCC where also ACC is filled. Here, only one jump is needed, omitting the accidental element '*költségvetés*' ('*budget*') (in bold), to get to the pVCC. (VCCs are in black, additional info is in gray. VCCs are presented here as JSON lists in the form of: slot, filler, slot, filler. . . , where *null* stands for a free slot.) The verb is '*hagy*' ('*allow*'), input data is taken from the development corpus. As we see, the same pVCC is found in both examples, it is '*hagy + NOM + ACC + FAC:jó*' which is word by word '*allow + SBJ + OBJ + FAC:good*' meaning '*approve + SBJ + OBJ*'. (ACC is for accusative case, FAC is for factive case.) This figure gives a good example of a typical pVCC which "is an endpoint of both jumps and stays", as we said earlier (on page 4).

6 Evaluation and Discussion

The evaluation was carried out in the following manner. Two moderately frequent verbs was chosen: '*húz*' ('*draw/pull*') and '*vet*' ('*cast/throw*'). Their data was taken from the testing part of the corpus (which was 93 percent of the corpus). Our "jump and stay" algorithm was run on these two verbs, and then – according to the f value – the first 20 pVCCs was investigated whether they are correct or not. The input data for these verbs were not only taken from the test corpus, but these verbs were not even looked at in any way during the development phase.

See the results of the evaluation in Table 1. Third column of the table contains the results of the algorithm: the Hungarian pVCCs, fourth column is f value, fifth column is an English translation word by word (or element by element), sixth column is an approximate English counterpart. pVCCs are shown as usual, the verb is taken separately at the top. Slots in Hungarian are marked by the three letter abbreviation of the given case marker: NOM is for nominative (subject) case, ACC is for accusative (direct object) case, and there are some others. Their surface form is not important here, their approximate translation can be seen in the fifth column. Unfilled NOM slots are not shown. (In Hungarian there are also postpositions. Apart from that they are separate words they play similar role as the case markers. Thus,

#	eval	Hungarian pVCC	*f*	word by word	English counterpart
		húz	**9505**	**draw/pull**	
1.	✓	ACC	8304	OBJ	pull sg
2.	✓	ACC:idő	420	OBJ:time	temporize
3.	✓	ACC:haszon + ELA	412	OBJ:profit + from	profit from sg
4.	✓	ACC + SUB:maga	239	OBJ + onto:oneself	put sg on
5.	✓	ACC + után:maga	209	OBJ + after:oneself	pull sg behind oneself
6.	✓	ACC + ALL:maga	207	OBJ + to:oneself	pull/draw sy to oneself
7.	≈	ACC + SUB:fej	199	OBJ + onto:head	put sg on one's head
8.	✓	felé	169	towards	be drawn/attracted towards sg
9.	✓	ACC:rövid	166	OBJ:short	get the worst of it
10.	✓	ACC:vonal	152	OBJ:line	draw a line
11.	✓	ACC:láb	139	OBJ:foot	drag one's feet
12.	✓	ACC:ujj + INS	118	OBJ:finger + with	pick a quarrel with sy
13.	p	ACC + NOM:aki	108	OBJ + SBJ:who	who pulls sg
14.	p	ACC + TEM:az	107	OBJ + at:that	pull sg at that time
15.	✓	ACC + INS:maga	92	OBJ + with:oneself	drag sy/sg with oneself
16.	✓	ACC + felé	85	OBJ + towards	pull sg towards sg
17.	×	ACC + közé	82	OBJ + between	draw sg *(a line)* between sg
18.	✓	ACC:szék	80	OBJ:chair	draw one's chair up
19.	✓	ACC:határ	77	OBJ:border	set limits
20.	✓	ACC:idő + INS	77	OBJ:time + with	temporize on sg
		vet	**14759**	**cast/throw**	
21.	✓	ACC	13649	OBJ	cast/throw sg
22.	≈	ACC + SUB	5437	OBJ + onto	cast/throw sg on sg
23.	✓	ACC:vég + DAT	2632	OBJ:end + for	put an end to sg
24.	✓	ACC + SUB:szem	1085	OBJ + onto:eye	reproach sy for sg
25.	≈	ACC:maga	964	OBJ:oneself	throw oneself
26.	✓	ACC:pillantás + SUB	839	OBJ:glance + onto	glance at sy/sg
27.	✓	ACC + SUB:papír	673	OBJ + onto:paper	note down sg
28.	✓	ACC:fény + SUB	402	OBJ:light + onto	reflect *(well/badly)* on sy/sg
29.	✓	ACC:szám + INS	371	OBJ:number + with	take sg into account
30.	✓	ACC:gát + DAT	362	OBJ:obstacle + for	put a stop to sg
31.	≈	ACC:maga + SUB	345	OBJ:oneself + onto	throw oneself into sg
32.	✓	ACC:maga + ILL	339	OBJ:oneself + into	throw oneself into sg
33.	p	ACC:az + SUB:szem	302	OBJ:that + onto:eye	reproach sy for that
34.	✓	SUB:maga	297	onto:oneself	have only oneself to blame
35.	✓	ACC:szem + SUB	285	OBJ:eye + onto	take a fancy to sy/sg
36.	✓	ACC:kereszt	261	OBJ:cross	cross oneself
37.	✓	ACC:árnyék + SUB	258	OBJ:shadow + onto	cast/throw a shadow over sy/sg
38.	✓	ACC + ILL:lat	240	OBJ + into:*lat*	use sg *(one's power)*
39.	p	ACC + SUB:én	225	OBJ + onto:me	cast/throw sg onto me
40.	p	ACC + NOM:aki	201	OBJ + SBJ:who	who casts/throws sg

Table 1: Evaluation of the "jump and stay" method on '*húz*' ('*draw/pull*') and '*vet*' ('*cast/throw*'). Correct pVCCs are marked with ✓. Further explanation is in the main text.

they have their own slots in some pVCCs, we can find '*után*' ('*after*'), '*felé*' ('*towards*') or '*közé*' ('*between*') in the table.)

Nevertheless, the most important column is the second one which contains the evaluation of the given pVCC in column three. There are four possible values here: ✓ means correct, ≈ means roughly correct, p means contains a pronoun as a filler, and × means not correct (i.e. not complete or not clean).

On the one hand, we see that 70-80 percent of the pVCCs are completely correct, which can be considered a high value in itself. On the other hand, only one single real error is found among 40 constructions which is only 2.5 percent. This one is #17, it is not complete, the direct object slot would be filled by '*vonal*' ('*line*'). The p code indicates a rather trivial problem, which seems to be easily eliminated. Pronouns are very common so they can appear as fillers, but they very rarely bear idiomatic meaning. So the solution could be simply to delete them in a preprocessing step and leave a free slot instead. Note that '*oneself*' and '*each other*' are certainly exceptions here.

Looking through the table, we can make some interesting observations. We see several correct pVCCs, they are complete and also clean. Considering the last column we see that different pVCCs are often translated using completely different verbs. Optionality appears in the form of two (or more) versions of the same construction. #2 and #20 shows essentially the same construction, without and then with a specific complement. This shows that this expression is used both ways, and the ratio of f values (77/420 = 18%) tells us something about which one is how frequent. Constructions #28, #29, and #30 show the importance of our concept of completeness (see secton 1) which takes both collocation and complementation into account. A certain filler often brings in a certain complement, and a new complement is often a sign of a new pVCC.

7 Conclusion and Future Work

Taking the theoretical model of double cubes and corpus lattices we created a new method for discovering useful verbal expressions in corpora. Our idea is called "jump and stay" (see section 3) because, in simple terms, wandering through the corpus lattice the value of a certain function jumps up and then stays the same at certain locations, and these are the locations which points to our target expressions, the so-called proper verb centered constructions. These constructions are proper in the sense that they contain exactly the necessary elements. Consisting of a verb plus slots and fillers, they can be simple or even quite complex; they are not necessarily MWEs, but they are constructions indeed. The evaluation revealed that at least 70-80 percent of the obtained expressions are pVCCs. We worked with Hungarian data, but it would be more or less straightforward to experiment with other languages.

An encouraging feature of the algorithm that it provides complete expressions most of the time (see section 6), incomplete VCCs rarely turn up as pVCCs. However, it has limitations as well. We mentioned the problem with pronouns, another one that there is definitely place to work out some more sophisticated process for setting the threshold values for jumps and stays, but take a look at a more general observation now. In simple cases, if a stay is found (that means an additional element is needed), we add it to the VCC in question doing the step in the corpus lattice defined by this stay (cf. first listing in Fig. 4). But what if we have two (or more) potential additional elements which are not significant separately, but together (their f values added up) they would define a regular stay? In other words, what to do when two (or more) elements seem to be mutually exclusively mandatory at one point? This question can result in some incomplete pVCCs now, and solving this is a promising development direction.

Our conclusion is that the original idea works. The present implementation can be considered as a proof of concept with respect to the corpus lattice model. Clearly, properties of the corpus lattice refer to where pVCCs are located, the introduced lattice structure turned out to be suitable to find them. On the other hand, the "jump and stay" principle also proved to be promising. The basic algorithm presented here can be improved in several aspects, and also the properties, the natural structure of corpus lattices (of given verbs or verb classes) can be further investigated, explored and taken advantage of in the future.

Acknowledgement. This research was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences (case number: BO/00064/17/1; duration: 2017-2020).

References

Pernilla Danielsson. 2007. What constitutes a unit of analysis in language? *Linguistik online* 31(2).

Stefan Evert and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*. Toulouse, France, pages 188–195.

Afsaneh Fazly and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the EACL*. Trento, Italy, pages 337–344.

Adele E. Goldberg. 2006. *Constructions at Work*. Oxford University Press.

Uxoa Iñurrieta, Arantza Diaz de Ilarraza, Gorka Labaka, Kepa Sarasola, Itziar Aduriz, and John Carroll. 2016. Using linguistic data for English and Spanish verb-noun combination identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 857–867. https://www.aclweb.org/anthology/C16-1082.

Paul Kay and Laura A. Michaelis. 2015. How constructions mean. In *Proceedings of the 11th Workshop on Multiword Expressions*. Association for Computational Linguistics, Denver, Colorado, page 44. https://doi.org/10.3115/v1/W15-0907.

Carlos Ramisch, Silvio Ricardo Cordeiro, Agata Savary, Veronika Vincze, Verginica Barbu Mititelu, Archna Bhatia, Maja Buljan, Marie Candito, Polona Gantar, Voula Giouli, Tunga Güngör, Abdelati Hawwari, Uxoa Iñurrieta, Jolanta Kovalevskaitė, Simon Krek, Timm Lichte, Chaya Liebeskind, Johanna Monti, Carla Parra Escartín, Behrang QasemiZadeh, Renata Ramisch, Nathan Schneider, Ivelina Stoyanova, Ashwini Vaidya, and Abigail Walsh. 2018. Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pages 222–240. https://www.aclweb.org/anthology/W18-4925.

Bálint Sass. 2015. 28 millió szintaktikailag elemzett mondat és 500000 igei szerkezet [28 million syntactically annotated sentences and 500000 verbal expressions]. In *XI. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY2015)*. Szeged: JATEPress, pages 303–308.

Bálint Sass. 2018. A lattice based algebraic model for verb centered constructions. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue*, Springer, Berlin Heidelberg New York, pages 231–238. Lecture Notes in Computer Science, Vol. 11107.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1526–1534. https://doi.org/10.18653/v1/D16-1159.

Dirk Siepmann. 2005. Collocation, colligation and encoding dictionaries. Part I: Lexicological aspects. *International Journal of Lexicography* 18(4):409–444.

Katalin Ilona Simkó, Viktória Kovács, and Veronika Vincze. 2017. USzeged: Identifying verbal multiword expressions with POS tagging and parsing techniques. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*. Association for Computational Linguistics, Valencia, Spain, pages 48–53. https://doi.org/10.18653/v1/W17-1705.

Wolfgang Teubert. 2005. My version of corpus linguistics. *International Journal of Corpus Linguistics* 10(1):1–13.

Abigail Walsh, Claire Bonial, Kristina Geeraert, John P. McCrae, Nathan Schneider, and Clarissa Somers. 2018. Constructing an annotated corpus of verbal MWEs for English. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pages 193–200. https://www.aclweb.org/anthology/W18-4921.

Offence in Dialogues: A Corpus-Based Study

Johannes Schäfer[1] and **Ben Burtenshaw**[2]

[1] Institute for Information Science and Natural Language Processing
University of Hildesheim, Universitätsplatz 1, Hildesheim, Germany
johannes.schaefer@uni-hildesheim.de

[2] Computational Linguistics & Psycholinguistics Research Center
The University of Antwerp, Lange Winkelstraat 40-42, Antwerp, Belgium
benjamin.burtenshaw@uantwerpen.be

Abstract

In recent years an increasing number of analyses of offensive language has been published, however, dealing mainly with the automatic detection and classification of isolated instances. In this paper we aim to understand the impact of offensive messages in online conversations diachronically, and in particular the change in offensiveness of dialogue turns. In turn, we aim to measure the progression of offence level as well as its direction – For example, whether a conversation is escalating or declining in offence. We present our method of extracting linear dialogues from tree-structured conversations in social media data and make our code publicly available.[1] Furthermore, we discuss methods to analyse this dataset through changes in discourse offensiveness. Our paper includes two main contributions; first, using a neural network to measure the level of offensiveness in conversations; and second, the analysis of conversations around offensive comments using decoupling functions.

1 Introduction

Offensive language is a complex problem, especially in social media where operators are required to counter illegal hate speech in user-generated content. However, it is not clear what counts as offensive language since even humans struggle to find objective definitions (Chen et al., 2012). The general approach to this problem is to train systems for the detection of such unwanted content based on human annotations of empirically gathered data instances.

Several shared tasks engaged with the topic of offensive language detection in recent years, for example *OffensEval-2019* for "Identifying and Categorizing Offensive Language in Social Media" (Zampieri et al., 2019b) or *GermEval-2018*, the "Shared Task on the Identification of Offensive Language" (Wiegand et al., 2018). The difficulty for machine learning systems at this task becomes apparent when considering the performance scores of the submitted systems. For example, at *GermEval-2018* the best performing submitted system reached a macro-averaged F_1-score of only 76.77 %. Here, systems struggle to simultaneously detect all various types of offensive language and it remains highly questionable if we can act on these automatic predictions and delete detected offensive language in practical applications.

Following the hint that deletion based on predictions of a machine learning system might not be the most appropriate course of action, we try to approach the problem of offensive language from another direction. In almost the same manner as mentioned above, we rely on machine learning of annotated instances to detect messages which might be offensive; however, we intend to act differently. Rather than deleting supposedly unwanted instances, we suggest to use tactics to counter offence. We aim for an empirical approach to automatically gather such tactics in a first step by a data analysis of instances where humans attempt to defuse offensive situations. In this paper we present our corpus creation using social media data from *Reddit* and discuss methods to analyse offensive dialogues. With our method we intend to classify conversations by offence direction, to facilitate future study on language use in offensive conversations. This research step could

[1] https://github.com/Johannes-Schaefer/oid_ranlp19

Proceedings of Recent Advances in Natural Language Processing, pages 1085–1093,
Varna, Bulgaria, Sep 2–4 2019.

prove vital to Natural Language Generation researchers in their effort to tackle offensive language by developing similar tactics.

Hate speech detection, which is closely connected to the detection of offensive language, however, focusing on illegal offence, has also been researched extensively in recent years. An overview of seminal work in this field is given by Schmidt and Wiegand (2017).

Context aware models for hate speech detection have been analysed by Gao and Huang (2017). Their dataset also preserves the thread structure of microposts, however, their approach is to use the context to gather additional features for the detection process while they do not focus on analysing the change of offensiveness in the conversation.

In related work on offensive language detection in conversations, Khatri et al. (2018) present an extensive data collection strategy using different sources of social media data. While they also utilise *Reddit* as data source, they only analyse individual utterances regarding offence.

A corpus with a focus on analysing conversations is presented by Walker et al. (2012) as they mine data from a forum and specifically consider the structure of comments in threads. While this corpus has several annotations which might be useful for exploring issues pertaining to online debate, they do not discuss offensive language.

Our research shows similarities to Zhang et al. (2018) who try to identify conversations that are likely to turn into offenses and predict the point in which this is likely to happen. We would also like to refer to the notions of constructive language which is discussed by Kolhatkar and Taboada (2017) and to a certain extent related to cases of defusing offensive conversations.

For our analysis of contexts of offensive language in dialogues, we decided to acquire our own corpus material which we process using methods tailored for this task. In summary, the research questions which drive our data analysis are as follows: How do people in dialogues react to offensive language? – Especially in terms of: what tactics do they try to counter offence? From a methodological view, we are particularly interested in investigating how to measure the change of offensiveness in turns of a dialogue.

The further sections of this paper are structured as follows: Section 2 presents our data sources and outlines the corpus construction process. In Section 3 we list our methods for data processing and analysis which are then applied to our dataset in experiments as described in Section 4. Finally, we conclude by discussing the results and the efficiency of our methods in Section 5.

2 Corpus Data

The analysis of offensive comments in dialogues requires a dataset of user posted messages which are referencing other messages. Such data can mainly be found on online social media where multiple users discuss a certain topic while interacting with each other. In this section we discuss our corpus construction process and motivate our selection of dialogues.

2.1 Data Sources

The typical sources of online social media data, *Twitter* and *Facebook*, have been mined extensively for text data to be researched, also with regard to containing offensive language or hate speech. The microposts from *Twitter* often lead to flat conversation structures as users mostly initiate a discussion or directly reply to an initial post. Hence, we disregarded this data source for our research on the change of offensiveness in a dialogue. We also decided not to use data from *Facebook* as we could not locate a restricted domain. We were afraid that when only including messages from a few selected *Facebook*-sites, we would not be able to get enough data for a statistical analysis.

To acquire a corpus of deeply structured dialogues about constrained topics, we decided to sample comments from *Reddit*[2] which is a social news aggregation, web content rating, and discussion website. In our corpus we only include comments from the Europe-*Subreddit*[3] where users post news or discussions which are geographically or politically related to Europe. We reason this decision on the basis that we – as Europeans – feel eligible to assess the content of these posts and we assume this topic to include lively (possibly heated) discussions containing offensive language. While *Reddit* is an American organisation, authors of posts in the Europe-*Subreddit* are mostly Europeans, but this is not restricted. Non-European users are also allowed to participate in the discus-

[2] https://www.reddit.com/

[3] A *Subreddit* is a forum dedicated to a specific topic as part of the website *Reddit*. The Europe-*Subreddit* can be found at https://reddit.com/r/europe.

sions, however, the content has to be related to topics from Europe.

2.2 Source Data Structure

Comments on *Reddit* are structured in threads, each one being a directed graph form of a rooted tree – similar to forums. An initial post, called submission, is directly posted in a *Subreddit* where users share content by posting stories, links, images, and videos. It can also just consist of a headline. Users can then reply to the initial post or to previously posted replies in a recursive manner. Thus, a thread can comprise several comments structured in a tree with the submission as a main or top-level (level 0) post, which corresponds to the root of the tree. Direct replies to the top-level post, direct successors of the root, we consider as being on level 1, replies to those in turn as being on level 2, etc. Leaf nodes are posts which have no further direct replies. Thus, there is always exactly one directed path from the root to any comment in a thread. A comment can technically never be a direct reply to multiple comments, i. e. cannot have multiple direct predecessors.

2.3 Data Acquisition

The first step to construct our corpus is to download *Reddit* posts using the *Python 3 psaw*[4] module, which is a minimalist wrapper for searching public *Reddit* comments and submissions via the *pushshift.io Reddit API*[5]. Our script selects all comments in the time frame from 2009-12-31 23:00:00 (first posts in the Europe-*Subreddit*) until 2019-04-04 22:00:00 (date of our corpus initialisation). The downloaded submissions and comments are stored as individual dictionary objects, however, contain metadata (ID for itself and a link to the ID of the predecessor) which allows to reconstruct the abovementioned tree structure. We store them in a *pickle* object and save it into a file.

2.4 Corpus Format

To be able to efficiently process these threads automatically as well as manually by annotation, we convert the downloaded data into a specifically tailored *Extensible Markup Language (XML)* corpus format with the following general structure: A single root element `subredditcorpus` is defined as containing all the other elements. It consists of multiple `submission` elements which correspond to the threads in our Europe-*Subreddit* data. A `submission` element contains an optional `main_post` element (which is not present in a few cases when the submission consists only of a headline) and an arbitrary number of `comment` elements. A `main_post` element can either consist of a `link` element, in cases where no text comment is submitted and only a link to an external site or another *Reddit* post is given, or of a `comment` element itself. A `comment` consists of a text string and can recursively nest further comments – besides the `comment` element of the `main_post` element, which never has a successor; however, we do not ensure this in our *XML Document Type Definition (DTD)*.

Several types of metadata are maintained in our *XML* corpus, such as post IDs which allow us to find the original source on the website. Additionally, we store for each comment the `date` of the post, the `author` ID (*Reddit* user name), the `author_flair` (which is a customisable string appearing next to the user name and specific to each *Subreddit*; in the Europe-*Subreddit* it is possible to choose a country name as a flair and users usually select their country of origin) and the `score` of the post which was assigned by other users (via down- or upvoting the post).

3 Methods

In this section we describe our methods for corpus annotation and processing. First, we present our offensive language detection system in Section 3.1 which is based on a neural network and predicts offensiveness probabilities for each comment. Then we give our methods for automatically extracting uniformly structured linear dialogues containing offensive language from the corpus in Section 3.2. Finally, in Section 3.3, we show how we intend to further analyse these linear dialogues by applying decoupling functions to model the change of offensive probability.

3.1 Offensive Language Detection

To detect the level of offensiveness of comments we use a supervised machine learning method, which is a typical approach for this task. We train a model on manually labelled messages which have been classified whether they contain offensive language or not. Our system operates solely on the linguistic text of an individual comment

[4] https://github.com/dmarx/psaw

[5] https://github.com/pushshift/api

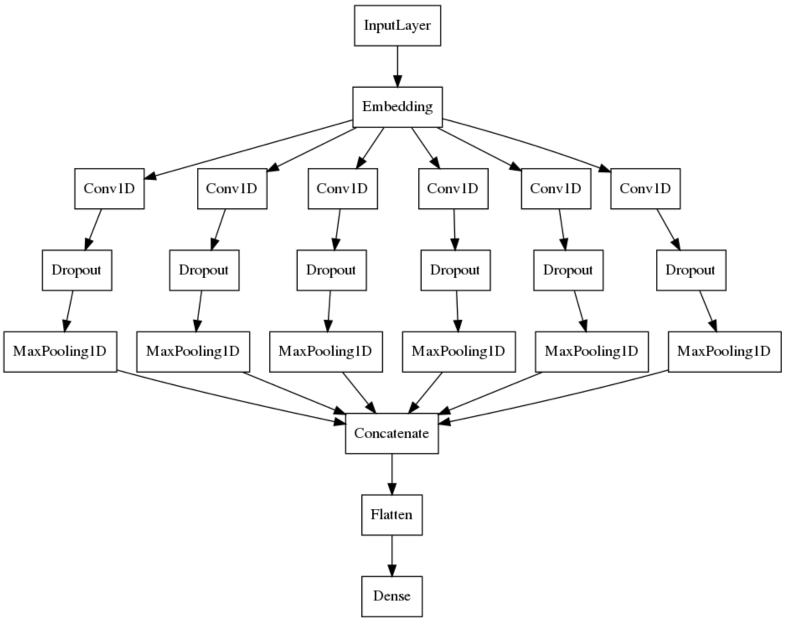


Figure 1: Neural network model architecture of our offensive language detection system.

as input since there is no overlap in the types of metadata between the training data and our corpus data. For each comment, given its text and the trained model, the system computes an offensiveness probability which is annotated in our corpus. In the remainder of this section we elaborate on a few details of our system – for the full configuration refer to our provided code.

We use a neural network to train a model of offensiveness of short text posts. A neural network is able to learn highly complex functions given enough labelled training data instances. Convolutional Neural Networks (CNNs) are structures commonly applied to natural language which can automatically identify sequences of words in a text that are significant features for a specific classification task. By design, CNNs contain regularisation which is capable to abstract from a limited set of training instances to unseen – ideally similar – test data instances.

Neural Network Architecture: Our overall neural network model is designed as follows (see also Figure 1). In a first step the tokenised[6] input text is encoded as a sequence in an embedding layer. We use (*Tweet-*) word embeddings by Deriu et al. (2017) to represent the meaning of our input text based on the principle of distributional semantics. To model offensive language based on the word embeddings of the input text, we apply six parallel CNNs with different window sizes (from one to six). These CNNs process the given sequence by moving a filter over it from left to right, in each step selecting a number (depending on its fixed window size) of consecutive words and computing a weighted combination of the dimensions of their word embeddings. Thereby, we aim to select word n-grams which are likely to be relevant to assess if a given text instance contains offensive language. Each CNN consists of a single convolutional layer followed by a dropout layer and a max pooling layer. With the inherent regularisation of a CNN by convoluting and max-pooling plus the additional dropout layer (here we use a dropout probability of 0.25), this step includes a considerabe amount of regularisation. We justify this design for our encoder by considering that our test data (*Reddit* comments) is vastly different from the training data (*Twitter* microposts) in terms of text type and website. By using a high amount of regularisation we intend to be able to generalise well when predicting on out-of-domain text. In the final steps of our neural network, the encoded output of the parallel CNNs is concatenated, reformatted using a flatten layer, and finally we compute a score which can be interpreted as offensiveness probability using a densely-connected layer with a sigmoid activation function. The exact parametrisation of the model is given in our provided code.

[6] For tokenisation we use the *NLTK TweetTokenizer* (`www.nltk.org/api/nltk.tokenize.html`) which includes custom-built methods to deal with social media text.

Training Procedure: We train our model using the *Offensive Language Identification Dataset (OLID)* (Zampieri et al., 2019a) which has been used for the *OffensEval 2019*[7] shared task of *"Identifying and Categorizing Offensive Language in Social Media"*. The training dataset consists of 13,240 *Tweets* as individual messages labelled for containing offensive language or not. The distribution of non-offensive to offensive messages in this dataset is approximately 2:1, i. e. there are 4,400 messages labelled as being offensive. Our training algorithm optimises the weights of the model based on 11,916 samples (90%) of these annotated instances and validates on the remaining 1,324 samples (10%) to avoid overfitting. Early stopping is executed when the performance on the validation set did not improve in the last few training epochs and we load the weights from after the epoch which lead to the maximum validation set performance.

Prediction: During testing we apply the abovementioned trained model to each comment of our corpus and annotate the predicted offence as follows. We add new metadata to our *XML* corpus by including the attributes `p_off` and `off` for the `comment` elements. While the predicted probability score is directly stored as value of `p_off`, the value of `off` expresses if the predicted probability of a comment to contain offensive language is higher than 0.5, which we annotate as one of the possible binary values `"True"` or `"False"`.

3.2 Extraction of Linear Dialogues

The abovementioned corpus creation process (cf. Section 2) provides us with a dataset of comments structured in a forum-like manner as a tree – a top post with direct replies which can in turn have direct replies themselves and so on. After we processed this corpus using our offensive language detection system, the comments in this corpus include annotations expressing their offensive probabilities. However, as we aim to analyse conversations as turn-based dialogues around offensive comments, we now have to filter the corpus and extract such linear dialogues. In this section we describe our method for this extraction process.

Our corpus can be formally described as a set of comments C, a set of submissions $S \subset C$ and a set of relations R, where each comment $c_i \in C$ is either a submission or top post $c_i \in S$ or a direct reply to exactly one other comment $c_j \in C$, i. e. $\forall_{c_i \in C} \exists!_{c_j \in C} : (c_j, c_i) \in R$.

#comments = n	#submissions
$n \leq 10$	265,068
$10 < n \leq 100$	83,295
$100 < n \leq 1000$	8,539
$n > 1000$	80
average	14
average$_{(n>1)}$	54

Table 1: Number of comments per submissions.

For our target of linear conversations we now need to extract a simplified dialogue out of this dataset, i. e. a linear structure of turns. In general, we consider a linear conversation an ordered list of comments from our corpus as $\{c_1, c_2, ..., c_n\}$ where $\forall_{i \in \{1,\ldots,n-1\}} : (c_i, c_{i+1}) \in R$.

We assert the following requirements to refine this structure. As we intend to compare dialogues around offensive comments with each other and find patterns amongst them, we have to analyse a uniform structure of contexts. Thus, we require all linear conversations to have a fixed length (number of comments) and the context before and after an offensive comment to be equal in size. Additionally, we included the the top-level comment for each linear conversation to give information about the general topic in each case.

Thus, we define a linear conversation from our corpus as $l = \{c_0, c_1, c_2, ..., c_n\}$, where $\forall_{i \in \{0,\ldots,n\}} c_i \in C$, $\forall_{i \in \{1,\ldots,n-1\}} : (c_i, c_{i+1}) \in R$, $c_0 \in S$, $\texttt{p\_off}(c_{\frac{n}{2}}) > 0.5$ with $n = 2 * k + 2$, where $k \in \mathbb{N}$ corresponds to the number of comments of the context to the left and right of the offensive comment (or window size) and there has to be a path from c_0 to c_1.

3.3 Decoupling Functions

In order to determine the progression of offensive probability we have preliminarily tested two gradient based approaches on linear dialogues of two users: firstly, using the gradient of all dialogue turns $(c_1, \ldots, c_n)$; secondly, comparing the gradient before and after the offensive comment (comparison of $(c_1, \ldots, c_{\frac{n}{2}-1})$ to $(c_{\frac{n}{2}+1}, \ldots, c_n)$). We visualise these approaches in our experiments in Section 4.4. In future work we intend to compare the gradient of the two different users. This will theoretically show the most meaningful forms of

[7] https://competitions.codalab.org/competitions/20011

c_i	author	off_p	comment
0	user_1	0	Do you find your government to be trustworthy?
1	user_2	0.11	So why exactly they don't feel safe? Because Russia planned Babchenko assasination? Or because SBU managed to prevent that and safe him and 30+ others?
2	user_1	0.20	The way I understood it, they were not feeling safe in the first place, and now they feel that the authorities undermined what little trust they had.
3	user_2	0.97	If your understanding is correct than I must state that those journalists are dumb as fuck and infantile people.
4	user_1	0.85	Why are they dumb? Because they are sceptical of this whole circus?
5	user_2	0.30	Because they can't set priorities between national security, life and death of people and their personal feelings.
6	user_1	0.52	The issue is that they don't trust the government. They don't believe this thing was needed, or even worse, they don't believe that there was an actual threat in the first place. Do you find your government to be trustworthy? Why do you believe that this was real?
7	user_2	0.64	"The issue is that they don't trust the government." No one trust gov-t in Ukraine. Any gov-t. It's a national feature. Sometimes it's good, sometimes it's bad. And up to 100% of journalists only criticizing the gov-t all the time - it's just how things work here. People don't trust the officials and journalists wat to be in trend. I doubt that any of those journalists actually lost some trust to gov-t. They just never had it and now using this situation to state it.

Table 2: Example of a linear dialogue containing offence from our corpus.

dialogue progression. We plan to publish a detailed analysis of these examples, but for illustrative purposes and to articulate the motivation in data collection, below are brief overviews of instances where offence probability declines.

4 Experiments

In this section we describe our observations when we applied the abovementioned methods to our corpus data.

4.1 Corpus Analysis

The download of *Reddit* submissions and comments using the mentioned API led to the total number of 11,217,768 posts (356,982 being submissions). In Table 1 we show the distribution of comments per submissions for certain ranges, to gain an understanding of how comments are structured in threads in this dataset. While the vast majority of submissions (approximately 74%) have very few replies ($\leq$ 10 comments), only 80 submission have more than 1000 comments. However, there is a substantial amount of threads with more than 10 comments and the average number of comments per thread – if we exclude threads with none or only one comment – is 54. Thus, we consider this dataset to be rich enough to study conversations and to be representative for different phenomena.

4.2 Offensive Language Detection

Our training algorithm for offensive language detection calls early stopping on training epoch 10, where a maximum binary accuracy is reached on the validation dataset. The evaluation after this epoch shows a binary accuracy of 0.82 on the training data and a binary accuracy of 0.73 on the validation data.

4.3 Extraction of Linear Conversations

To extract linear conversations as we defined them above, we tested different window sizes for the context. We decided to at least have 50k instances of conversations around offensive comments as we expect this to be a representative set of different types of phenomena. We set the window size $k = 3$ for the contexts before and after offensive comments, which leads to 67,456 instances of linear dialogues of length 7 (plus one for the top post), i. e. here $l = \{c_0, c_1, \dots, c_7\}$ while $\texttt{p\_off}(c_4) > 0.5$. In Table 2 we provide an individual example of a linear conversation from our

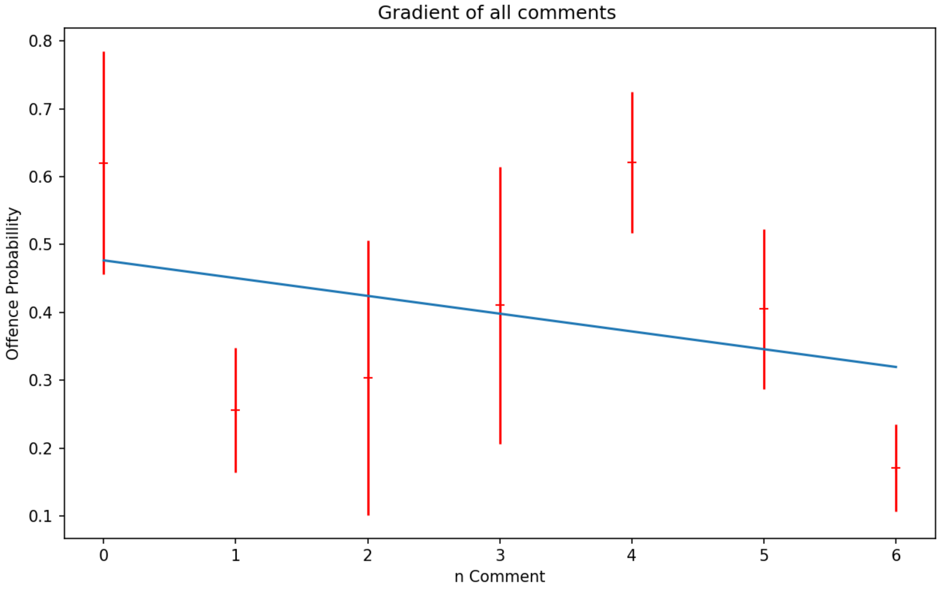


Figure 2: Gradient of all comments in extracted linear conversations.

dataset including offensive probabilities.

We have to consider that linear conversations from our dataset can have a certain partial overlap. From the data structure it only follows that the comments in the context before the offensive comment are given, here $\{c_0, c_1, c_2, c_3\}$. For the comments after the offensive comment (here $\{c_5, c_6, c_7\}$), there is the possibility to produce multiple linear conversations when there are multiple direct replies to one comment. We call this case branching, i. e. when we have multiple linear conversations which differ only in some of the comments $\{c_5, c_6, c_7\}$ – when following a different branch in the tree.

To analyse the frequency of this phenomenon, we counted how many unique first parts exist in our set of 67,456 linear conversations. If we only consider $\{c_0, \ldots, c_4\}$, there are 54,286 unique instances. If we add c_5, there are 57,077 unique instances and if we add c_6 there are 60,647 unique instances. Considering these values, we assume that branching is rather infrequent and we have mostly entirely different linear dialogues of this fixed length.

4.4 Progression of Offensive Probability

We now want to investigate the change of the level of offensiveness in turns of our linear dialogues statistically.

Complete Linear Dialogue Progression: We calculated the linear regression gradient of all comments in the collected linear dialogue, and searched for instances where gradient decline was the steepest. In simple terms, these reflect conversation where replies have been less offensive than the stimulus. The graph in Figure 2 shows the mean placement of each dialogue turn for the 50 steepest gradient dialogues, as well as standard deviation in red; a line of best fit is shown in blue. A clear downward vector in the overall gradient is shown, as well as a steep decline in the latter half of the dialogue, standard deviation accounted for.

Pre/Post Offence Comparison: We used the initiating offensive instance (comment 4) to split the dialogue into two halves (comment 1-4 and 4-7), and then we compared their gradients. We searched for instances with opposing directions. Unlike the above approach, comparing two halves allows us to see the impact on dialogue of the offensive comment, and measure how it was reacted to. The graph in Figure 3 shows a mean downward trajectory for the fifty most clear examples of this gradient.

In the example instance given in Table 2 the initiating comment probability is 0.85 in a reply to a clearly offensive comment towards a group of journalists (probability of 0.97). The latter half of the dialogue passively expands on that point, without returning to profanity or offence. We can reasonably say that user_1 is consistently responding to user_2 without offending, and therefore of interest to a study on inoffensive approaches to offensive dialogue.

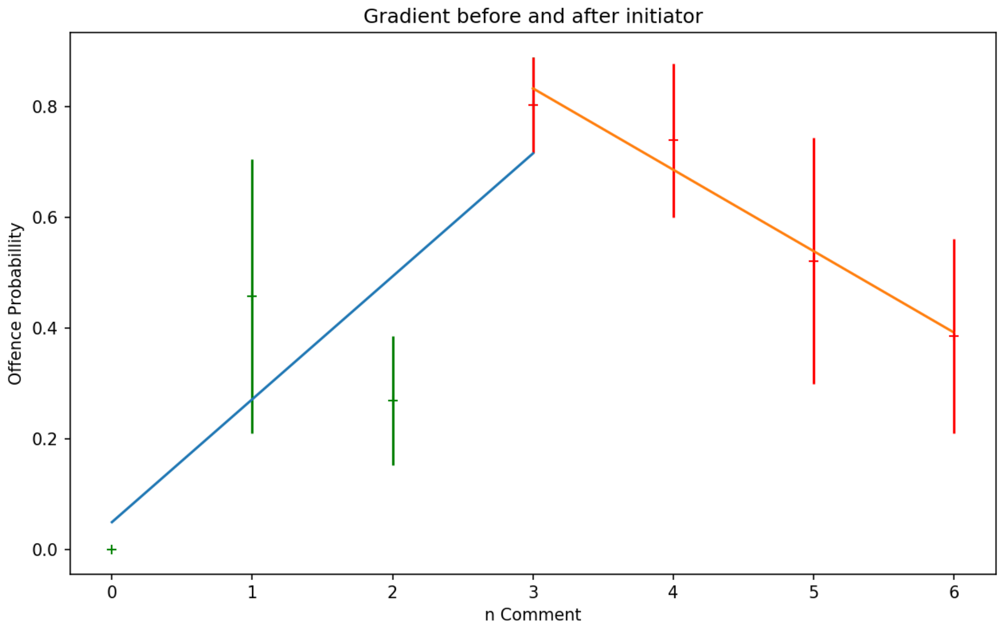


Figure 3: Gradient in comments before and after offensive comment trigger.

5 Conclusion

In this paper we presented our data collection strategy for a corpus containing conversations and discussed methods to analyse the corpus for linear dialogues around offensive comments. With a substantial dataset of over 11 million posts we are able to analyse over 50k unique linear dialogues, each consisting of seven turns predicted to contain offensive language. We now discuss our two main contributions.

Using offensive-tagged individual messages as training data for assessing the offensiveness of dialogue turns: Our method to detect offence is based on a machine learning model which predicts whether individual messages contain offensive language or not. In our model architecture, the computed probability expresses the confidence of the system that words and n-grams of the post are typical expressions of offensive language. We understand the offensiveness of a dialogue turn to be approximately in line with this assessment. A post can be seen as highly offensive when it contains several word sequences which are usually used to express offence. A low level of offensiveness can be expressed by using word sequences which are only incidentally used in offensive comments, i. e. they might for example not be offensive to everyone.

We also want to note that our offensive language detection model is not fully optimised: we intend in future work to experiment with different training data. The rather low scores given for the performance on the training/validation set can also be justified by the high amount of regularisation in our model which was implemented to make it generalise more when applied to our dataset (which is different from the training data). The training algorithm also does not optimise on accuracy as we use class weights, giving the more infrequent class of offensive comments a higher weight. If we further go into the direction to base methods for automatic analyses on the computed scores, it might be worth to investigate further how to optimise the detection system.

Analysing conversations around offensive comments using decoupling functions to find tactics to counter offence: We have shown that the analysis of gradients should be considered when we want to measure the change in offensiveness of a conversation. With the use of decoupling functions it especially seems suitable to split dialogues around offensive comments into two halves to find tactics to counter offence, i. e. especially instances where the gradient declines after the offensive comment trigger.

In future work we aim to focus on researching manual and statistical methods to find tactics to counter offence. However, the analyses of the given dataset and the provided code show promising results by pointing into the right direction.

References

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. IEEE, pages 71–80.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, pages 1045–1052.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*. INCOMA Ltd., Varna, Bulgaria, pages 260–266.

Chandra Khatri, Behnam Hedayatnia, Rahul Goel, Anushree Venkatesh, Raefer Gabriel, and Arindam Mandal. 2018. Detecting offensive content in open-domain conversations using two stage semi-supervision. *arXiv preprint arXiv:1811.12900* .

Varada Kolhatkar and Maite Taboada. 2017. Constructive language in news comments. In *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics, Vancouver, BC, Canada, pages 11–17.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. pages 1–10.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*. Istanbul, pages 812–817.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018), Vienna, Austria September 21, 2018. - Vienna, Austria: Austrian Academy of Sciences. Pp. 1-10* .

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983* .

Justine Zhang, Jonathan Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 1350–1361.

EmoTag – Towards an Emotion-Based Analysis of Emojis

Abu Awal Md Shoeb   **Shahab Raji**   **Gerard de Melo**
Department of Computer Science
Rutgers University, New Brunswick, NJ, USA
{abu.shoeb, shahab.raji, gerard.demelo}@rutgers.edu

Abstract

Despite being a fairly recent phenomenon, emojis have quickly become ubiquitous. Besides their extensive use in social media, they are now also invoked in customer surveys and feedback forms. Hence, there is a need for techniques to understand their sentiment and emotion. In this work, we provide a method to quantify the emotional association of basic emotions such as anger, fear, joy, and sadness for a set of emojis. We collect and process a unique corpus of 20 million emoji-centric tweets, such that we can capture rich emoji semantics using a comparably small dataset. We evaluate the induced emotion profiles of emojis with regard to their ability to predict word affect intensities as well as sentiment scores.

1 Introduction

In recent years, information technology has profoundly altered the way humans communicate. A substantial proportion of the global population has adopted the use of social media platforms (such as Twitter, Facebook, and Instagram) and messaging technology (such as Facebook Messenger, WeChat, and WhatsApp) to interact and voice their opinion. The unique properties and expressive capabilities afforded by computer and mobile device-mediated communication has led to quite distinct forms of expression in comparison with classic email etiquette, let alone traditional written correspondence.

Meanwhile, for any meaningful analysis of social interactions or expression of opinions, it is critical to extract and understand the sentiment and the affect of the source. There are numerous studies investigating the connection between words or sentences and the affects they convey. However, emojis are a particularly prominent feature of modern online interaction. Thus, this paper introduces a new basis for studying this new modality with regard to conveyed affective associations. Emojis have become widespread in social media, and are variously used to carry emotional and contextual information pertaining to the content of social media posts. There have been studies exploring the relationship between hashtags and tweets (Ferragina et al., 2015), and between emojis and tweets (Campero et al., 2017). Additional research has aimed at conducting sentiment analysis based on emojis and hashtags (Novak et al., 2015). A number of other works study the connection between words and emotions, resulting in datasets such as *EmoLex* (Mohammad and Turney, 2013). Most of these studies relied upon a crowdsourcing approach to compile the data and lexicons and to capture relationships among linguistic and paralinguistic elements (Kulahcioglu and de Melo, 2019).

However, previous work has neglected to focus on the emotional aspects of emojis. For instance, we may ultimately be interested in devising a system that jointly assesses the affect conveyed by a tweet based not only on the words, but also in part on the emojis occurring within it. In some cases, an emoji may reinforce the emotion conveyed by the text. In other cases, it may reveal an additional dimension of affect. In some cases, it may also point in the opposite direction, e.g., by helping to discern sarcasm, which otherwise might be hard to ascertain in certain contexts. Currently, there are no readily available resources to understand the direct relationship between emojis and emotions.

We address this gap by harvesting an emoji-centric collection of tweets. From this, we create the *EmoTag* resource. The name alludes to its usefulness in exploiting emoji for emotional tag-

Proceedings of Recent Advances in Natural Language Processing, pages 1094–1103,
Varna, Bulgaria, Sep 2–4 2019.

ging. The resource is based on a series of co-occurrence statistics that allow us to quantify the emotional associations of individual emojis. We subsequently assess these connections in a series of experiments and case studies.

2 Background

Emotion and Communication. Facial expression has been an important aspect of communication that predates the emergence of mankind. Chevalier-Skolnikoff, in ascending order of phylogenetic complexity, draws connections between the degree of evolution of the brain and the spectrum of facial expression observed for a species (Chevalier-Skolnikoff, 1973). Charles Darwin's well-known volume on the expression of emotions (Darwin, 1872) analysed the connection between emotions and their expression. He remarked for instance, that for both animals and humans, anger coincides with eye muscle contractions and teeth exposure, and commented on the fact that humans lift their eyebrows in moments of surprise. His work then goes on to study the role of such forms of facial expression in conveying to others how an animal feels, studying primates as well as human infants and adults.

In light of this, humans continue to rely extensively on such nonverbal cues even in oral forms of linguistic communication. Although a person's emotion and mood can to some extent be conveyed by means of suitable content words (e.g., "I am happy to hear that!") or interjections ("Wow!"), face-to-face communication has important properties that written communication tends to lack (Bordia, 1997). These include facial expressions of the aforementioned sort, but also gesture and intonation. In certain circumstances, e.g. certain problem-solving settings, face-to-face communication may hence prove more efficient and effective (Bordia, 1997).

Accordingly, since the beginning of writing, humans have resorted to surrogate mechanisms to convey emotive signals, attempting to push the boundaries and overcome some of the inherent restrictions of plain written language as a medium. Examples include illustrative embellishments and ornaments, calligraphy, a judicious use of color, and various typographic instruments. For instance, it has been shown that the choice of font may radically alter the affective perception of a text (Juni and Gross, 2008; Kulahcioglu and de Melo, 2018).

Emoticons and Emoji. While emoticons such as ":-)" and Japanese 顔文字 (kaomoji) such as "(^_^)", both based on regular characters, have been in use for several decades, emojis originated in Japan in the 1990s and have only recently spread globally. Despite the lexicographic similarity between the two words *emoji* and *emotion*, etymologically, the former stems from the Japanese words 絵 (e, *picture*) and 文字 (moji, *character*). Emoji characters, similar to earlier dingbat characters, are pictorial and colorful.

Their principal use has indeed been to convey emotion, particularly via facial expression emojis. In 2015, Oxford Dictionaries declared the *Face with Tears of Joy* emoji its Word of the Year 2015. Kaye et al. (2017) explained how emojis may aid the interlocutor in disambiguating utterances that would otherwise remain ambiguous. Emojis may also be useful as a more instantaneously and widely recognized form of communicating degrees of satisfaction. Kay et al. go as far as suggesting them for consideration as possible alternatives to regular Likert scales (Kaye et al., 2017).

Historically, the spread of emojis has been driven in large part by their adoption in popular messaging and social media platforms, which led, among things, to their inclusion in Shift JIS, and, subsequently, the Unicode standard. Nowadays, they are ubiquitous in social media and chat applications, but increasingly also in emails and other digital correspondence.

3 Related Work

Emoticons. Early studies focused on the use of emoticons in social media. Go et al. (2009) proposed a form of distant supervision by using emoticons as noisy labels for Twitter sentiment classification. Davidov et al. (2010) adopted a fairly similar approach by handpicking smileys and hashtags as tweet labels and relying on a supervised method for sentiment analysis of tweets.

Emoji Semantics. A prominent work on emojis is the DeepMoji project (Campero et al., 2017) from MIT. It provided a model that recommends emojis given a natural language sentence as input. The deep learning model was trained on a collection of 1.2B tweets to learn the sentiment, emotions, and the use of sarcasm in short text.

Barbieri et al. (2016) proposed a method to learn vector space embeddings of emojis using the standard word2vec skip-gram approach, applied to a large collection of tweets. In contrast, Eisner et al. (2016) attempted to learn vector embeddings of emojis based on their short descriptions in the Unicode standard.

Emoji Associations. The first paper that thoroughly investigated the sentiment of emojis (Novak et al., 2015) proposed a sentiment ranking of 715 emojis on a corpus of 70,000 tweets. This work provides a basis for future research on the logographic usage of emojis in social media.

Zhou and Wang (2017) trained a natural language conversation model that accounts for the underlying emotion of utterances by exploiting the existence of emojis as a signal.

Rakhmetullina et al. (2018) proposed a method to classify emojis with regard to their sentiment and emotion. Their corpus consists of 500 labeled tweets, and they categorize emojis by assigning them labels for 8 emotions. For this, they applied a distant supervision technique for a reliable mapping based on manually annotated data.

4 EmoTag

Given the prominence of emojis in human communication, our work seeks to study relevant associations of emojis. We begin by assembling a dataset for this purpose (Section 4.1), and subsequently induce a series of lexicons that reveal potential connections (Section 4.2), including between words and emojis, as well as between emojis and emotions.

4.1 Data Collection

In assembling a collection of social media postings containing both emojis and hashtags with tweets, one strategy would be to rely on available datasets and filter them so as to retain only those entries that contain both emojis and hashtags. However, this approach results in a comparably small number of postings. Despite the overall surge in popularity of emojis, only a fraction of all postings includes emojis.

Instead, we proceeded to compile a new dataset of about 20.8 million tweets by specifically searching for postings that contain emojis. For the set of target emojis, our goal was to focus on emojis associated with emotions, as opposed to generic symbols from domains such as transportation or household appliances. To this end, we relied on a set of most frequently used 620 emojis from Novak et al. (2015) and from Emoji Tracker[1], a website that monitors the use of emojis on Twitter in realtime.

Using our set of frequent emojis as search terms, we retrieved tweets that specifically contain one or more of these target emojis. The number of tweets is evenly distributed across different emojis. While tweets can be in any language, we only collected tweets labeled as being in English. In total, we obtained a set of 20.8 million tweets over a span of one year. In addition to the volume that such a large time span provides, collecting the data for every day of the year aids in mitigating the effect of potential biases in the data. All collected tweets contain at least one emoji.

Note that only a fraction of all tweets have hashtags. Specifically, within our collected data, we found that only 10-15% of our tweets with emojis also include hashtags. To clean up the data, we removed usernames (marked with @-symbol), tweets consisting only of hashtags and emojis but no text, tweets that only contain a short time stamp such as “6AM” or simply a URL (with or without the “http://” prefix), as well as all duplicate tweets.

4.2 Lexicon Induction

Based on the corpus, EmoTag is constructed as a series of lexicons.

4.2.1 Co-occurring Emojis

We first collect a series of co-occurrence based lexicons. Each entry in such a lexicon is the representation of pairwise count of desired unigram tokens. These resources can be useful for the community, but also allow us to conduct analyses of the data.

In our tweet collection, there are roughly 36K tweets per emoji, and these have a uniform distribution across the collection time period.

Inspecting the results, we observe that the overall top-ranked pair of co-occurring emojis in our dataset is U+1F61D 😝 and U+1F61C 😜. These showed up together 42K times, which is fairly frequent in comparison with other pairs. Note that U+1F61D 😝 is the “face with stuck-out tongue and tightly-closed eyes” emoji, while U+1F61C 😜 is the “face with stuck-out tongue and winking eye” emoji.

Another emoji, U+1F602 😂, the “face with tears of joy” one, appears to be the most common emojis to co-occur saliently with others. It appears

[1] http://emojitracker.com/

Unicode	Emoji	Sentiment Score	Description
U+1F649	🙉	0.333	Hear-no-evil monkey
U+1F648	🙈	0.432	See-no-evil monkey
U+1F649	🙉	0.333	Hear-no-evil monkey
U+1F64A	🙊	0.459	Speak-no-evil monkey
U+1F62D	😭	-0.093	Loudly crying face
U+1F602	😂	0.221	Face with tears of joy
U+1F620	😠	-0.299	Angry face
U+1F608	😈	0.265	Smiling face with horns
U+1F620	😠	-0.299	Angry face
U+1F629	😩	-0.368	Weary face

Table 1: Similarities and Contrasts of Co-occurring Emojis

Word	Emoji	Description
miss	U+1F62D 😭	Loudly crying face
	U+2764 ❤	Heavy black heart
	U+1F622 😢	Crying face
happy	U+1F389 🎉	Party popper
	U+2764 ❤	Heavy black heart
	U+1F618 😘	Face throwing a kiss
love	U+1F60D 😍	Smiling face with heart-shaped eyes
	U+1F618 😘	Face throwing a kiss
	U+2764 ❤	Heavy black heart

Table 2: Co-occurring Emojis and Words

with a broad range of other emojis with a relatively high frequency. Three other popular emojis that co-occurred with U+1F602 😂 include U+1F62D 😭 (“loudly crying face”), U+1F648 🙈 (“see-no-evil monkey”), and U+1F629 😩 (“weary face”).

Somewhat different from the previous cases, the fourth pair in Table 1 involves the emoji U+1F62D 😭, i.e., a crying face, and U+1F602 😂, i.e., a face with tears of joy. This is unusual in the sense that these two emojis possess opposite sentiment polarities. According to Novak et al. (2015), the sentiment value of U+1F62D 😭 is -0.093, whereas the sentiment value of U+1F602 😂 is 0.221, i.e., a positive sentiment. This suggests that people tend to conflate the two due to their similar appearance, as both involve tears. Another possibility is that people may be using one of the two sarcastically. As shown in the table, similar observations can also be made for certain other pairs of emojis.

Our results also show a correlation between U+1F60D 😍 and U+1F629 😩. The two are paired up around 2,500 times, illustrating another connection between a positive and a negative sentiment emoji. U+1F629 😩 is the “weary face” emoji, whereas U+1F60D 😍 is the “smiling face with heart-shaped eyes” one. This appears to stem from tweets that express positive sentiment about a target entity, but also negative sentiment about the current situation.

4.2.2 Emoji–Words Lexicon

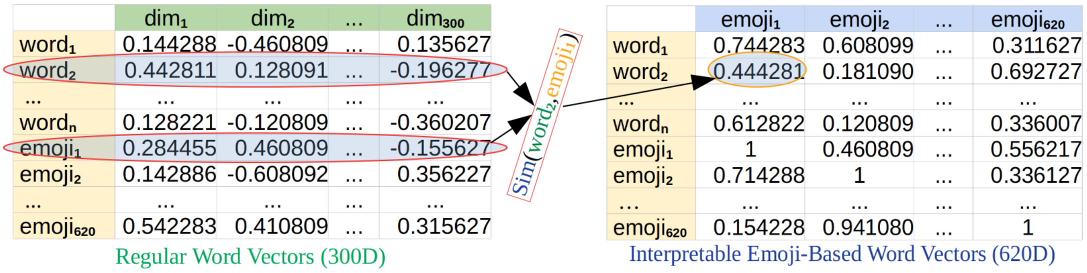
Another lexicon that we produce aims to provide co-occurring words for a given emoji, or, vice versa, emojis for a given word. Table 2 shows an excerpt of the emoji–word lexicon, grouped by words. For example, the word “miss” co-occurs with a wide range of emojis, but the top co-occurring emojis are U+1F62D 😭, U+2764 ❤, and U+1F622 😢. These emojis are likely to be used when someone misses someone or something. Similarly, the words “happy” and “love” appear with numerous emojis that carry happy and positive sentiment.

4.2.3 Emoji–Hashtags Lexicon

This lexicon provides a collection of hashtags along with the emojis that they co-occur with. The resource also includes the corresponding co-occurrence frequencies between emojis and hashtags. According to our findings, the emoji U+1F637 😷 (“face with medical mask”) co-occurs with the hashtags #sick, #flu, #yuck, #cold, #insomnia, and #dying, which all are clearly semantically relevant for this emoji.

4.3 Interpretable Emoji-Based Word Vectors

Interpretability and explainability are widely regarded as highly desirable attributes of AI-driven decision making (Xian et al., 2019). Dense word vectors such as those produced by word2vec (Mikolov et al., 2013) are ubiquitous in NLP (de Melo, 2017). However, it is often remarked that they lack interpretability, in the sense that individual values in such vectors do not carry any easily interpretable inherent significance. Previous work has proposed interpretable word vectors consisting of one or more sentiment polarity scores for a word (Dong and de Melo, 2018; Dong and de Melo, 2018). Given that emojis represent a wide spectrum of aspects considered relevant in human communication, we study to what extent emojis can serve as a means of inducing word vectors endowed with interpretability.



	dim_1	dim_2	...	dim_{300}
$word_1$	0.144288	-0.460809	...	0.135627
$word_2$	0.442811	0.128091	...	-0.196277
...	...	...	...	...
$word_n$	0.128221	-0.120809	...	-0.360207
$emoji_1$	0.284455	0.460809	...	-0.155627
$emoji_2$	0.142886	-0.608092	...	0.356227
...	...	...	...	...
$emoji_{620}$	0.542283	0.410809	...	0.315627

Regular Word Vectors (300D)

	$emoji_1$	$emoji_2$	...	$emoji_{620}$
$word_1$	0.744283	0.608099	...	0.311627
$word_2$	0.444281	0.181090	...	0.692727
...	...	...	...	...
$word_n$	0.612822	0.120809	...	0.336007
$emoji_1$	1	0.460809	...	0.556217
$emoji_2$	0.714288	1	...	0.336127
...	...	...	...	...
$emoji_{620}$	0.154228	0.941080	...	1

Interpretable Emoji-Based Word Vectors (620D)

Figure 1: Inducing Interpretable Word Vectors via Emojis

This can be achieved by assigning every word a 620-dimensional word vector, in which each dimension reflects the association of that word with one out of 620 emojis. Since we use a list of the 620 most frequent emojis, the dimensionality of a vector becomes 620. An obvious method would be to adopt just simple frequency counts as the values in these vectors, i.e., for a given word, the entries in its word vector would simply reflect the number of times that word co-occurred with a given emoji.

However, we can improve over this by relying on the word2vec Skip-Gram with Negative Sampling algorithm (Mikolov et al., 2013) as an intermediate representation, as illustrated in Fig. 1. We first train such a word2vec model on the EmoTag corpus. Then a cosine similarity score is calculated between all words and emojis. This yields a semantic relatedness score in $[0, 1]$ for any word–emoji pair. Thus, we can view the score as reflecting to what extent a word correlates with an emoji. We use these correlation coefficients to form a word vector $\mathbf{v}_w \in [0, 1]^d$ for every word w, such that each of the $d = 620$ dimensions reflects the correlation with a particular emoji. This is the final EmoTag word vector representation that we use in all experiments.

5 Evaluation

In the following, we evaluate EmoTag for machine learning-driven emotion analysis of tweets and show how it can be used to reveal the sentiment and emotion of individual emojis.

The first study aims at evaluating the usefulness of our interpretable EmoTag word vectors in a downstream task, exploiting them in a machine learning-driven system that seeks to identify the emotion intensity of tweets.

Subsequently, we use our data to compute sentiment polarity scores for emojis, comparing these against existing human annotations of emoji sentiment.

Finally, we develop the first resource providing emotion scores for emojis. We evaluate these by showing how they can be used to automatically induce emotion scores for words.

5.1 Emotion Intensity Prediction with Interpretable Emoji-Based Word Vectors

We begin by evaluating the interpretable emoji-based word vectors, assessing to what extent they are able to keep up with regular word vectors in a downstream task relating to emotions.

Benchmark. In particular, we consider the EmoInt Shared Task from WASSA (Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis) 2017 (Mohammad and Bravo-Marquez, 2017a), which involves determining the intensity or degree of emotion felt by a speaker when a tweet and a target emotion are given. Tweets were provided for four different emotion categories (anger, fear, joy, and sadness), and the ground truth intensity values range between 0 and 1.

The Affective Tweets (AT) package was provided to all participants as a baseline for the competition (Mohammad and Bravo-Marquez, 2017b), providing a rich set of features constructed based on several emotion and sentiment lexicons such as NRC-EmoLex, NRC10E, etc. (Mohammad and Bravo-Marquez, 2017a).

Model. We rely on a deep neural network to predict the emotion intensity for each tweet, adopting a similar CNN-LSTM architecture as that of IMS (Köper et al., 2017), the 2nd-ranked system among all participants in the competition, with the CNN architecture based on that proposed by Kim (2014). In training, each tweet is represented by a matrix of size m x d, where d is the dimensionality of the pre-trained word vectors and $m = 50$ is the maximal token sequence length considered for

Method	A	F	J	S	Avg	d
Interpretable						
AT	0.65	0.66	0.60	0.69	0.65	n/a
EmoTag	0.70	0.73	0.69	0.75	0.72	620
Non-Interpretable						
Random Init.	0.68	0.72	0.66	0.73	0.70	300
Google News	0.70	0.72	0.67	0.75	0.71	300
GloVe	0.70	0.73	0.68	0.76	0.72	300
GloVe–Twitter	0.72	0.74	0.68	0.76	0.73	200
IMS	0.71	0.74	0.71	0.71	0.72	300

Table 3: Comparing with other methods, with regard to anger (A), fear (F), joy (J), sadness (S), average (Avg), dimensionality (d).

a tweet. We can thus feed in either regular word vectors or our interpretable emoji-based EmoTag vectors for the series of words in the tweet. We applied a dropout rate of 0.25. The obtained matrix then serves as input to a convolutional layer with a window size of 3, followed by a max-pooling layer (size 2) and an LSTM (Hochreiter and Schmidhuber, 1997) to predict a numerical output for each tweet. This numerical value was then added as a feature along with other auxiliary features, and passed to a Random Forest regressor to obtain the final intensity score for a particular emotion. The IMS team used a total of 142 features, including the 45 baselines features from Affective Tweets. Since we are comparing our results with both the baseline features and the features used by the IMS team, our classifier is also fed with the 142 features. All features were passed to a random forest regressor with 800 trees for identifying the intensity of a given emotion. A separate model is trained for each of the four target emotions.

Results. Table 3 summarizes the results for the EmoInt task, providing Pearson correlations for each emotion as well the average Pearson correlation for all four emotions, along with the dimensionality of the respective word vectors used in the experiments. The results show that the interpretable word vectors in EmoTag are able to yield results that are comparable with those of other dense word representations that are not interpretable. It should be kept in mind that EmoTag was built based on a very small corpus, i.e., only 20M tweets, comparing to the massive size of the corpora used for pretrained word vectors such as the two GloVe models. For further comparison, we also report results on just the AffectiveTweets (AT) features, as well as the original IMS system. In some cases, for example for the *sadness* emotion, EmoTag actually outperforms the IMS team's baseline.

Unicode	Emoji	Novak Score	EmoTag Score	Description
U+1F61C	😜	0.455	0.482	Face with stuck-out tongue and winking eye
U+1F617	😗	0.611	0.591	Kissing face
U+1F49A	💚	0.656	0.654	Green heart
U+1F48B	💋	0.691	0.744	Kiss mark

Table 4: Comparison of emoji sentiment scores from EmoTag and Novak et al. (2015).

5.2 Evaluating the Sentiment of Emojis

Next, we evaluate to what extent our interpretable word–emoji vectors can aid in revealing the sentiment of emojis.

Method. For obtaining sentiment scores, we rely on the NRC Emotion Lexicon EmoLex (Mohammad and Turney, 2013), a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The associations are merely given as Boolean labels (0 or 1). To obtain a sentiment score for an individual emoji, we first consider all words with a sentiment score of 1 in EmoLex. Then, we rank all words associated with the given emoji based on their similarity score according to our interpretable word vectors, where a higher similarity score results in a higher rank. According to the ranking, the top $K = 3$ words are picked and their similarity scores are aggregated using a simple addition, which becomes the ultimate sentiment score for the given target emoji.

Results. To evaluate the sentiment score of emojis, we measure the Pearson correlations for several groups of emojis treating the scores by Novak et al. (2015) as gold scores. Table 5 summarizes the Pearson correlations for several groups of emojis. The first row of the table represents Novak's top 100 positive sentiment emojis. We also consider additional groups based on the Unicode standard emoji descriptions, particularly those with a face and those with monkey faces.

Note that we observed a high positive sentiment score for all emojis with kiss symbol or kissing face in our data, compared to Novak's scores. For some emojis, our model obtains a high sentiment score such as 0.991 for 😽 "Kissing Cat Face

Emoji Group	Correlations
Top 100 positive emojis	0.71
Emojis with face	0.45
Monkey face emojis	0.53
Emojis with kissing	0.14

Table 5: Pearson Correlations for Sentiment Score

with Closed Eyes" U+1F63D, whereas the score by Novak et al. (2015) is 0.571. This can happen for several reasons. In some cases, the sentiment scores they propose may be misleading for certain emojis, especially if they are less frequent in their dataset. An example is 😽 U+1F63D, which has an occurrence frequency of 88 only, compared to emojis such as 😂 "Face with Tears of Joy" U+1F602, which occurred 14,622 times. Thus, in some cases, their results may not be reliable.

Still, the results often show a strong agreement, although our method produces sentiment scores for emojis only indirectly via their associations with words. Table 4 provides examples of such sentiment scores generated by EmoTag and Novak et al. (2015).

5.3 Evaluating Emotion Profiles of Emojis

Finally, we use our data to evaluate to what extent emojis are associated with certain emotions. For this, we again rely on our emoji-based word vectors in conjunction with EmoLex, the NRC Emotion Lexicon (Mohammad and Turney, 2013). EmoLex provides a set of words along with a set of binary labels, where 1 signifies that the word carries a particular association, while 0 represents the negative case.

Method. First, for each emoji, we identify the top K in EmoLex according to their cosine similarity with the emoji, as obtained in our interpretable word vectors, where a higher similarity score entails a higher rank. For the top K words, we compute a weighted average of emotion labels. The emotion labels are taken from EmoLex, while the similarity scores are used as weights. This weighted average then serves as the final emotion score of the emoji. The same process is followed for all emojis. **Results.** We evaluate our induced emoji emotion scores indirectly by using them to reproduce emotion intensity scores for words, for which we have ground truth intensity scores in the Affect Intensity lexicon by (Mohammad, 2018). This lexicon comes with 6K tokens, where tokes are grouped by the four emotions *anger*, *fear*, *joy*, and *sadness*. It provides crowdsourced emotion intensity scores, which range between 0 and 1, with 1 meaning that the word exhibits the highest degree of association with a particular emotion and 0 referring to the lowest degree. Note that this ground truth resource is distinct from the NRC Emotion Lexicon used in inducing our scores. The latter merely provides Boolean labels for word–emotion pairs, and thus it is non-trivial to derive affect intensity scores from it, particularly via emojis.

To reproduce word emotion intensities based on our emoji emotion scores, we proceed as follows. For a given word w, we rank the top K emojis based on their similarity score in the EmoTag word vectors, where higher scores entail a higher rank. Once the top K emojis have been identified, we then compute the arithmetic mean of the emotion scores of those related emojis, which yields the final emotion score for the target word w. We chose $K = 10$, which led to better results than alternative values.

Table 8 depicts the Pearson correlations for different subsets of the Affect Intensity lexicon. These correlations reveal how close we are in predicting the emotion score for a given word based on our emoji emotion scores. The first row shows the scores for words that are common to all four emotion groups, whereas the last row includes all words. Table 7 provides examples of emotion scores for a few select emojis.

Analysis. For further analysis, we compare our scores with the classification obtained by Rakhmetullina et al. (2018). Table 6 compares the emotional label that their classification provides against our emotion scores for *anger*, *joy*, *sadness*. Note that this is the complete set of emoji results provided in their paper, apart from one additional emoji for the emotion *surprise*, which our method currently does not support, due to its omission in EmoLex. Their labeling did not include the emotion *fear*, so we omit it in our comparison. The bold scores in the last three columns indicate what emotion labeling we would obtain if we had to select a single label for an emoji based on our obtained emotion intensity scores. For example, in our case, emoji 🙏 "Folded Hands" U+1F64F obtains the highest score 0.485 for the emotion *joy*, which is labeled as being in the *joy* category in their study as well. There are three cases (high-

Unicode	Name	Emoji	E2E Label	Anger	Joy	Sadness
U+1F612	Unamused face	😒	anger	**0.418**	0.119	0.333
U+1F602	Face with tears of joy	😂	joy	0.381	0.099	0.326
U+1F60D	Smiling face with heart-eyes	😍	joy	0.307	**0.308**	0.137
U+1F60A	Smiling face with smiling eyes	😊	joy	0.067	**0.248**	0.247
U+1F495	Two hearts	💕	joy	0.172	**0.383**	0.142
U+1F601	Beaming face with smiling eyes	😁	joy	0.091	**0.123**	0.079
U+263A	Smiling face	☺️	joy	0.095	**0.245**	0.176
U+1F604	Grinning face with smiling eyes	😄	joy	0.184	**0.188**	0.149
U+1F618	Face blowing a kiss	😘	joy	0.233	0.215	0.144
U+1F64F	Folded hands	🙏	joy	0.187	**0.485**	0.351
U+1F62D	Loudly crying face	😭	sadness	0.246	0.198	**0.272**
U+1F629	Weary face	😩	sadness	0.236	0.186	0.234
U+1F622	Crying face	😢	sadness	0.284	0.210	**0.333**

Table 6: A comparison between Emoji2Emotion (E2E) and EmoTag

Emoji	Name	A	F	J	S
U+1F620 😠	Angry face	0.49	0.36	0.07	0.44
U+1F46E 👮	Police officer	0.34	0.49	0.16	0.27
U+1F492 💒	Wedding	0.09	0.14	0.63	0.25
U+1F4A9 💩	Pile of poo	0.35	0.34	0.15	0.47

Table 7: Emotion scores of emojis for anger (A), fear (F), joy (J), sadness (S).

Tokens	A	F	J	S	Avg
Common Words	0.51	0.41	0.19	0.50	0.40
All Words	0.45	0.40	0.20	0.45	0.37

Table 8: Pearson Correlations of gold scores and our predicted scores for Affect Intensity lexicon

lighted in red) at which our scoring would fail. According to their labeling system and results, emoji 😩 "Weary Face" U+1F629 should have obtained its highest score for *sadness* (0.234) instead of *anger* (0.236), though both scores are very close in our case.

The emoji 😂 "Face With Tears of Joy" U+1F602 scored 0.381 on *anger*, which is the highest among all scores for it, although the authors of Emoji2Emotion marked it as belonging to the *joy* category. This may stem from the phenomenon of people frequently confusing this emoji with the 😭 "Loudly Crying Face" U+1F62D emoji. In Table 1, we observe that both appear together very often, which results in a strong association with a negative emotion (anger) for an emoji that intrinsically ought to be more associated with joy.

6 Conclusion

The characteristics of a medium profoundly affect the way that people express themselves using said medium. While written communication lacks the non-verbal cues that make face-to-face communication particularly effective for problem-solving (Bordia, 1997), modern social media, and messaging platforms have unique properties that are interesting in their own right. Among these, the use of emojis stands out as meriting very special consideration, not least due to their ability to compensate for some of the shortcomings of written language as a medium in conveying emotion and affect.

While research in social science and social media analytics has extensively studied the use of emojis in everyday communication, previous work has not fully explored the connection between emojis and emotion. This paper presents a detailed analysis of how emojis and words co-occur in social media, including their connection to emotions. It also shows how an interpretable word embedding can be formed with the help of emojis, which shows promise as an additional ingredient in emotion detection-related tasks.

Another key contribution of this work is the creation of a large resource, consisting of several different sub-lexicons that describe connections among emoji, words, and other items, as well as emotion scores for emojis, which are released to the public[2]. We hence believe that this work will substantially benefit other researchers in several different fields.

[2] http://emoji.nlproc.org

References

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016. What does this emoji mean? a vector space skip-gram model for twitter emojis. In *Language Resources and Evaluation conference, LREC*. Portoroz, Slovenia.

Prashant Bordia. 1997. Face-to-face versus computer-mediated communication: A synthesis of the experimental literature. *The Journal of Business Communication (1973)* 34(1):99–118. https://doi.org/10.1177/002194369703400106.

Andres Campero, Bjarke Felbo, Joshua B. Tenenbaum, and Rebecca Saxe. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv 1708.00524* https://arxiv.org/abs/1708.00524.

Suzanne Chevalier-Skolnikoff. 1973. Facial expression of emotion in nonhuman primates. In *Darwin and Facial Expression: A Century of Research in Review*, Academic Press New York, NY, pages 11–89.

Charles Darwin. 1872. *The Expression of the Emotions in Man and Animals*. Appleton. The original was published 1898 by Appleton, New York. Reprinted 1965 by the University of Chicago Press, Chicago and London,.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, Stroudsburg, PA, USA, COLING '10, pages 241–249. http://dl.acm.org/citation.cfm?id=1944566.1944594.

Gerard de Melo. 2017. Multilingual vector representations of words, sentences, and documents. In *Proceedings of IJCNLP 2017*. http://www.aclweb.org/anthology/I17-5002.

Xin Dong and Gerard de Melo. 2018. Cross-lingual propagation for deep sentiment analysis. In *Proceedings of AAAI 2018*. AAAI Press.

Xin Dong and Gerard de Melo. 2018. A helping hand: Transfer learning for deep sentiment analysis. In *Proceedings of ACL 2018*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bosnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*. Association for Computational Linguistics, Austin, TX, USA, pages 48–54. http://aclweb.org/anthology/W16-6208.

Paolo Ferragina, Francesco Piccinno, and Roberto Santoro. 2015. On analyzing hashtags in twitter. In *International AAAI Conference on Web and Social Media*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing* pages 1–6.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Samuel Juni and Julie S Gross. 2008. Emotional and persuasive perception of fonts. *Perceptual and motor skills* 106(1):35–42.

Linda K. Kaye, Stephanie A. Malone, and Helen J. Wall. 2017. Emojis: Insights, affordances, and possibilities for psychological science. *Trends in Cognitive Sciences* 21(2):66 – 68. https://doi.org/https://doi.org/10.1016/j.tics.2016.10.007.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. https://doi.org/10.3115/v1/D14-1181.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. IMS at EmoInt-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark.

Tugba Kulahcioglu and Gerard de Melo. 2018. Fontlex: A typographical lexicon based on affective associations. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the 11th Language Resources and Evaluation Conference (LREC 2018)*. European Language Resources Association (ELRA), Paris, France.

Tugba Kulahcioglu and Gerard de Melo. 2019. Paralinguistic recommendations for affective word clouds. In *Proceedings of ACM IUI 2019*. ACM, New York, NY, USA, pages 132–143. https://doi.org/10.1145/3301275.3302327.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Saif Mohammad. 2018. Word affect intensities. In *Proceedings of the 11th Language Resources and Evaluation Conference*. European Language Resource Association, Miyazaki, Japan. https://www.aclweb.org/anthology/L18-1027.

Saif Mohammad and Felipe Bravo-Marquez. 2017a. WASSA-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, Copenhagen, Denmark, pages 34–49. https://doi.org/10.18653/v1/W17-5205.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017b. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM)*. Vancouver, Canada.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence* 29(3):436–465.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *Plos One* 10(12). https://doi.org/10.1371/journal.pone.0144296.

Aisulu Rakhmetullina, Dietrich Trautmann, and Georg Groh. 2018. Distant supervision for emotion classification task using emoji 2 emotion. In *Proceedings of the 1st International Workshop on Emoji Understanding and Applications in Social Media (Emoji2018)*. Stanford, CA, USA. http://ceur-ws.org/Vol-2130/.

Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of SIGIR 2019*. ACM, New York, NY, USA, pages 285–294. https://doi.org/10.1145/3331184.3331203.

Xianda Zhou and William Yang Wang. 2017. Mojitalk: Generating emotional responses at scale. *arXiv 1711.04090* https://arxiv.org/abs/1711.04090v1.

A Morpho-Syntactically Informed LSTM-CRF Model for Named Entity Recognition

Lilia Simeonova
FMI
Sofia University
Bulgaria
lilia.valentinova.simeonova@gmail.com

Kiril Simov, Petya Osenova
LMaKP
IICT-BAS
Bulgaria
{kivs,petya}@bultreebank.org

Preslav Nakov
Qatar Computing
Research Institute, HBKU
Qatar
pnakov@qf.org.qa

Abstract

We propose a morphologically informed model for named entity recognition, which is based on LSTM-CRF architecture and combines word embeddings, Bi-LSTM character embeddings, part-of-speech (POS) tags, and morphological information. While previous work has focused on learning from raw word input, using word and character embeddings only, we show that for morphologically rich languages, such as Bulgarian, access to POS information contributes more to the performance gains than the detailed morphological information. Thus, we show that named entity recognition needs only coarse-grained POS tags, but at the same time it can benefit from simultaneously using some POS information of different granularity. Our evaluation results over a standard dataset show sizeable improvements over the state-of-the-art for Bulgarian NER.

1 Introduction

Although in recent years the *Named Entity Linking* (also known as Named Entity Disambiguation) task has been central in NLP research, the *Named-entity recognition* (NER) task has remained far from solve, having in mind the productivity of names and the amount of information available in the era of big and noisy data.

NER plays a critical role in the processing of texts with application to many real-world Natural Language Processing (NLP) tasks such as Question Answering, Information Extraction, Machine Translation, Dialog Systems, and chatbots, where it is sometimes called *Concept Segmentation and Labeling* (Saleh et al., 2014).

Traditionally, NER has focused on recognizing entities such as person (PER), organization (ORG), location (LOC), and miscellaneous (MISC). This tradition goes back to the Message Understanding Conference (MUC) for English (Grishman and Sundheim, 1996), and the subsequent CoNLL 2002/2003 shared tasks, which also targeted other European Languages such as Spanish, Dutch, and German (Tjong Kim Sang and De Meulder, 2003).[1] This same setup was followed in more recent work for a number of other languages, and we also follow it in the present work.

Early systems relied on hand-crafted rules with pattern-matching (Appelt et al., 1995). Unfortunately, this required an large pre-annotated datasets, collecting which was time-consuming and error-prone. The next step was to add gazetteers and lexicons that were generated automatically or semi-automatically (Popescu and Etzioni, 2005). Adding such resources required special approaches to resolve the ambiguity between names and common words. Such problems were solved using models such as Hidden Markov Models (Zhou and Su, 2002) and Conditional Random Fields (Sutton and McCallum, 2012).

In our work here, we use deep neural networks for Bulgarian NER. Lample et al. (2016) have shown remarkable results for English, using a combination of Bi-LSTMs (Bi-directional Long Short-Term Memory) and CRF. However, the approach is problematic for morphologically rich languages. The main problem is the missing information within word embeddings for the numerous word forms involved in multiword names that require additional grammatical knowledge in order to be processed properly. Here we incorporate such information as additional input to our neural model.

[1] Other schemata such as ACE (Doddington et al., 2004) used a richer inventory of entity types.

Proceedings of Recent Advances in Natural Language Processing, pages 1104–1113,
Varna, Bulgaria, Sep 2–4 2019.

Our contributions are as follows:

- We show that for morphologically rich languages such as Bulgarian the access to POS and morphological annotation is crucial and can yield very sizeable performance gains.
- We achieve sizable improvements over the state-of-the-art for Bulgarian NER.
- Finally, we make our data and code freely available, which should enable direct comparison in future work.[2]

2 Related Work

Our work is based on Bulgarian, but we claim that it is appropriate also for other languages with rich morphological systems like Slavic and Romance languages, for example. For that reason, we present first the best results for NER in other Slavic languages having in mind that they are synthetic, while Bulgarian is a predominantly analytic language whose morphological richness lies exclusively in the verbal system and not so much in the nominal one. Analytism implies more types of multiword named entities in Bulgarian but less inflection variety, and different distribution of the common types for these languages. The direct comparison of the numbers presented below should be taken with a grain of salt as they are on different datasets and for different languages. Yet, they are indicative for the different methods used for these languages.

For Russian, a Hybrid Bi-LSTM approach was applied by Le et al. (2018), who achieved precision of 89.57, recall of 84.89, and F1 score of 87.17. These results are comparable to the ones by our model using the same approach.

For Czech, Straková et al. (2013) reported precision of 88.27, recall of 78.00, and F1 score of 82.82 using a Maximum Entropy Markov Model. The feature modeling also proved to be working in Czech, as their best results used features based on morphological analysis, two-stage prediction, word clustering, and gazetteers.

For Polish, Piskorski et al. (2004) achieved precision of 91.0, recall of 77.5, and F1 score of 82.4. They used the SProUT system, which is an NLP platform, consisting of pattern/action rules.

In the last years, the interest in NER for Slavic languages grew. Two shared tasks were organized —- the first and the second Multilingual Named Entity Challenge in Slavic Languages. They have been descibed in (Piskorski et al., 2017) and (Piskorski et al., 2019). The challenges included several tasks: recognition of mentions of named entities in Web documents in seven Slavic languages (Bulgarian, Croatian, Czech, Polish, Russian, Slovak, Slovene, Ukrainian), their normalization/lemmatization as well as cross-lingual linking.

Our evaluation on NER in this paper is more similar to the relaxed evaluation parameter where the string is detected and classified, not the invariant. Considering the complexity of the task, the drop of the results per language and per entity types have been expected. Such a task, however, is also good motivation for improving the NER systems for Slavic languages, including Bulgarian.

There is some previous work on NER for Bulgarian. Georgiev et al. (2009) presented a model using Conditional Random Fields with several hand-crafted features. They combined well-established features used for other languages with language-specific lexical, syntactic, and morphological information. Their result is the previous state-of-the-art for Bulgarian.

So far, the highest reported results for NER are for English. For example, Chiu and Nichols (2016) reported an F1 score of 91.20 using Bi-LSTM + CNN + gazetteers + linking, while Passos et al. (2014) achieved an F1 score of 90.90 using a new form of learning word embeddings that can leverage information from relevant lexicons. For German, Gillick et al. (2016) achieved an F1 score of 82.84, which shows that the rich morphology causes a drop in the performance.

Currently, the prevalent paradigm in NLP is to use neural networks, typically based on LSTMs or CNNs. As we have mentioned above, Lample et al. (2016) proposed an LSTM-CRF model for NER.[3] The model uses a bi-directional LSTM to encode the left and the right context of the current input word. Then it passes the concatenation of the two hidden vectors (one produced by the left LSTM and one by the right LSTM) to a CRF model. Its task is to ensure the global consistency of the NER tags.

[2] http://github.com/lilia-simeonova/NER-bg/

[3] They also proposed a transition-based model inspired by shift-reduce parsers, but the results were worse.

In this model, each input word is represented as a concatenation of its word embedding and the character-level embedding for the word produced by a character Bi-LSTM. The character embedding provides features for the suffix and the prefix of the word. Thus, the left-to-right character-based LSTM embedding models the word suffix, while the right-to-left one models the word prefix. The word embeddings are trained in an unsupervised manner on external data, while the character-based LSTM embeddings are trained on the training data as part of the end-to-end training of the full LSTM-CRF model. This model does not need any explicit feature engineering nor does it need manual gazetteers; yet, it achieved state-of-the-art performance for four languages: English, German, Dutch, and Spanish. Here we take this model as a basis, and we augment it to model part-of-speech (POS) and grammatical information, which turns out to be very important for a morphologically complex language such as Bulgarian.

Strubell et al. (2017) extended the above model by substituting the LSTM with Iterated Dilated Convolutional Neural Networks, a variant of CNN, which permit fixed-depth convolutions to run in parallel across entire documents, thus making use of GPUs, which yields up to 20-fold speed up, while retaining performance comparable to that of the LSTM-CRF model. They further aggregated context from the entire input document, which they found to be helpful. In our preliminary monolingual experiments, this model performed very similarly, but slightly worse, than the LSTM-CRF model, and thus we chose LSTM-CRF for our experiments below.

3 Data

In this paper, we work with a Bulgarian corpus, annotated with BIO tags and positional tags, the same as in the CoNLL-2002 shared task (Tjong Kim Sang, 2002). The data is in BIO format, which encodes for each token in the text whether it is at the beginning of the expression of interest (Named Entity in our case), inside or outside of it. The annotation in the available Bulgarian data comes from the manually annotated Bulgarian treebank, known as BulTreeBank (Simov et al., 2004a). In the treebank, each NE is represented as a constituent consisting of one or more tokens. Each NE phrase is annotated by the categories Person, Organization, Location, and Other.

Христо Hristo	Стоичков Stoichkov	пристигна arrived	в in	София Sofia
B-PER	I-PER	O	O	B-LOC

Table 1: An example in BIO encoding.

The BIO tags for the tokens forming a given named entity (NE) in the treebank are created on the basis of the syntactic annotation. The first word in the phrase is marked as the beginning of the NE, while the rest of the tokens ar emarked as inside of the NE. The tokens that are not part of any NE are encoded as outside elements. In order to represent the category of the NE, each tag for begin and inside tokens includes a modifier for the category. Thus, we use nine labels: B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, B-MISC, I-MISC, O. The example in Table 1 shows a simple sentence annotated with two named entities: a person name (*Hristo Stoichkov*) and a location name (*Sofia*).

Besides the BIO tagging, the texts in the dataset inherited the morphosyntactic annotation from the treebank. This annotation uses the BulTreeBank Morphosyntactic Tagset (Simov et al., 2004b). The tagset is positional. It encodes parts-of-speech and grammatical features for Bulgarian. For example, *Npfsi* stands for noun, proper, feminine, singular, indefinite. This annotation offers an opportunity to explore how the morphological features can affect NER.

The resulting dataset is divided into three disjoint sets: training set (Train), development set (Dev), and test set (Test). Table 2 shows statistics about the annotated data. We can see that a large number of examples are labeled as Person names, and that the distribution of Locations, Persons and Organizations is not balanced. While we can still build a stable system based on this data, the class imbalance makes our model more vulnerable to overfitting. Thus, we use early stopping in order to prevent the model from continuing to learn weights and parameters if it does not see an improvement in the final score.

	Sent.	**Tokens**	**PER**	**ORG**	**LOC**	**MISC**
Train	28,636	528,567	16, 804	3,028	6,786	911
Dev	4,063	64,014	2,514	515	1,021	227
Test	3,907	60,645	1,875	305	781	112

Table 2: Statistics about the data.

4 Model

As mentioned above, we construct our model as a modification of the Bi-LSTM-CRF architecture from (Lample et al., 2016). After some experiments with the original system, we decided to modify its input: we added a vector representing some of the information encoded in the morphosyntactic tags. Thus, we created the input vectors for the tokens in the sentences as a concatenation of three vectors: a word embedding vector, a character embedding vector, and a vector containing some grammatical features, called a *grammatical vector*. We experimented with different grammatical vectors, as explained below.

The rest of the Bi-LSTM-CRF architecture of (Lample et al., 2016) was kept as in the original model: First, we run a Bi-LSTM over the sequence of word vectors so that we could get their contextual word representation. We use a fully connected neural network to get a score for each of the tags. At the end, we run a CRF decoder to decide what the best combination of scores is.

The key takeout of our model is that we use some feature modeling to show that for a morphologically rich language such as Bulgarian using POS and grammatical information can improve the results. Thus, we mix automatically learned features — the word and the character embeddings —, with hand-crafted features encoded as a grammatical vector.

In the rest of this section, we describe the different components of our system.

LSTM-CRF Implementation For the implementation of the general LSTM-CRF architecture, we use Tensorflow (Sak et al., 2014).

Word Embedding Nowadays there are many different approaches to train word vectors such as Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), FastText (Bojanowski et al., 2017), and many more. In our experiments, we use the pre-trained Bulgarian word embeddings from FastText (Bojanowski et al., 2017).[4] This choice was motivated by the fact that FastText uses the structure of the words by taking into consideration character n-grams, thus modeling morphology and many out-of-vocabulary words.

Character Bi-LSTM Embedding In order to produce character embeddings, we use a bidirectional LSTM over the character representation of the text. For each character in the text, each of the two LSTMs produces an hidden vector. For each word, the hidden vector for the last character produced by the left-to-right LSTM models information about the suffix of the word. Similarly, the hidden vector for the first character produced by the right-to-left LSTM models information about the prefix of the word. Following the approach, used in (Ling et al., 2015), we constructed the final character embedding of the word as a concatenation of the prefix and the suffix vectors.

Grammatical Vectors We use several types of grammatical vectors or their combinations. They are divided into POS vectors that encode different combinations of parts-of-speech and morphological vectors encoding other grammatical features.

POS Vectors The part-of-speech information for each word is represented as an a one-hot vector with eleven positions. This vector is concatenated to the vector for the word embedding. In the tagset, we have the following parts-of-speech: **N** — noun, **A** — adjective, **V** — verb, **H** — hybrid, **D** — adverb, **R** — preposition, **P** — pronoun, **C** — conjunction, **T** — particle, **M** — numeral, and **I** — interjection. In our experiments, we divided these parts-of-speech into different groups depending on the role they play in the representation of the named entities. For example, the tags A, N, H, R were viewed as a possible part of a named entity in contrast to the others that cannot form named entities. In this case, the one-hot vector contains only two positions. The groups are given in the experimental section below. The Hybrid tag (H) is special in the tagset. It refers to both family names and name adjectives. Bulgarian family names (as other Slavic ones) are proper names, but morphologically they behave like adjectives due to their adjectival origin.

Morphological Vectors The nominal system of Bulgarian shares some features with other Slavic languages, such as agreement in grammatical gender and number, rich pronoun system, etc. However, it has also specific features, such as the post-positioned definite article and lost nominal declension system. Our aim is to show the contribution of all these types of features to the named entity recognition task.

[4] In this work, we do not use any contextualization of the word embeddings such as ElMo (Peters et al., 2018) and BERT (Devlin et al., 2019), as our Bi-LSTM architecture already performs contextualization.

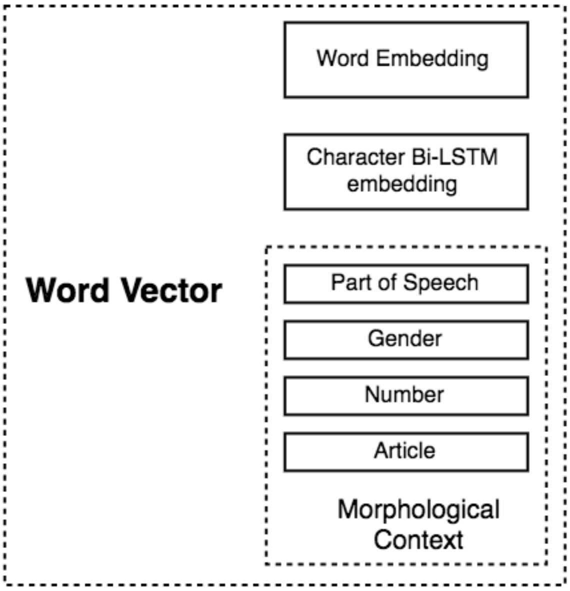


Figure 1: Full input vector representation with concatenation of word embeddings, character embeddings, and morphological features.

Another thing worth mentioning is that the nouns, adjectives and hybrid tags share some common features. This information appears to be very useful for recognizing the more specific types of named entities. Note that the existence of a preposition in a sequence can distinguish some further patterns as well.

The morphological features vary between the different entities, but there are few that can be defined for nouns, adjectives, hybrid tags and pronouns. Some of them are arguably useful, such as gender, number, and definiteness, and we will describe them briefly here:

Gender can have three values: *masculine, feminine, and neutral*;

Number can have four values: *singular, plural, only plural* and *count form* (which is only for masculine nouns for non-persons)'

Definiteness can have four values: *indefinite, definite, short definite* and *full definite*. The latter two are for singular masculine nouns only.

For each word with a POS tag of noun, pronoun, adjective or hybrid, we concatenate a one-hot vector representation for each of the features above (we use a zero vector for the rest). We form the final version of our word vector as a concatenation of all contextual vectors, as shown on Figure 1.

Dropout In order to prevent overfitting, we use a dropout layer on top of our word vectors as described in (Hinton et al., 2012). For each presentation of each training example, we randomly exclude a hidden unit from the network with a certain probability. In this way, the system learns to detect and use more useful features.

5 Experiments and Evaluation

In this section, we present the experimental setup and the evaluation results for the different models we experimented with.

5.1 Training and Hyper-parameters

We experimented with different values of the hyper-parameters and we found that changing some of them can result in sizeable improvements. The most considerable difference was for the learning method and the learning rate. Our best resulst were achieved using the Adam optimizer (Kingma and Ba, 2014), which is computationally efficient and has minimal memory requirements.

At the beginning, we set the learning rate to the initial value of 0.001, and then at each epoch, we multiplied it by a specific learning decay value. Decreasing our learning rate over time can help us find the minimum of our function without actually missing it. While Adam already decays the learning rate at each iteration, previous work has found that tuning the initial learning rate could yield sizeable improvements over the default settings. Thus, we use this additional decay. (Wilson et al., 2017)

The word embeddings we use from FastText have a dimensionality of 300, while the character embedding vectors have a dimensionality of 100. In order to produce them, we uses the TensorFlow default Xavier initializer and then we ran a Bi-LSTM on top of them in order to obtain contextual vectors with no additional layers.

We set the batch size to 20 and the dropout to 2. The Adam's parameters we used are as follows: $learningrate = 0.001$; $\beta 1 = 0.9$; $\beta 2 = 0.999$; $\epsilon = 1e - 08$; $use\_locking = False$. We also added a gradient clipping with a value of 1.

At decoding time, we used a linear-chain CRF (Lafferty, 2001). This model has been shown to outperform a simple SoftMax classifier as the tagging decision needs to be global.

Model	No Morphology	With Morpholofy		
	F1	F1	P	R
Model + POS-2	90.04	90.44	92.02	88.90
Model + POS-3	91.20	91.11	91.66	90.57
Model + POS-4	90.16	90.83	92.30	89.41
Model + POS-5	91.32	90.58	92.42	88.82
Model + POS-11	90.96	91.03	91.60	90.48
Model + POS-3+11	91.18	92.20	93.31	91.12
Model + POS-4+11	90.89	91.04	92.17	89.94

Table 3: Evaluation results for Bulgarian POS tagging. Shown are results where the standard input to the Bi-LSTM-CRF model is augmented with different POS tags and morphological features.

English	Ivan Valtchev visited the Bulgarian Academy of Sciences.						
Bulgarian	Иван Вълчев посети Българската академия на науките.						
Tokens	Иван	Вълчев	посети	Българската	академия	на	науките
POS11	N	H	V	A	N	R	N
POS2	ANHR	ANHR	O	ANHR	ANHR	ANHR	ANHR
POS3	ANH	ANH	O	ANH	ANH	R	ANH
POS4	NH	NH	O	A	NH	R	NH
POS5	N	H	O	A	N	R	N

Table 4: Example of the POS tags for annotation schemes of different granularities when applied to the same Bulgarian sentence.

5.2 Experiments

The experimental results suggest that adding grammatical features can have a sizeable impact on the performance of the general LSTM-CRF model for Bulgarian NER. In Table 3, we can see an example of different combinations of POS tags, where *Model* stands for Bi-LSTM-CRF and POS represents a one-hot encoding for the following:

POS11 = all part of speech tags separately

POS2 ”ANHR” vs. REST

POS3 ”ANH” vs. ”R” vs. REST

POS4 ”A” vs. ”NH” vs. ”R” vs. REST

POS5 ”A” vs. ”N” vs. ”H” vs. ”R” vs. REST

POS3 + POS11 POS11 vs. ”ANHR” vs. REST

POS4 + POS11 POS11 vs. ”ANH” vs. REST

Morph Gender, number, and definiteness

We further perform several experiments in order to determine whether we need the full set of part-of-speech tags or it is enough just to know whether the entity is part of the group of the nouns, adjectives, hybrid tags, and prepositions.

Model	F1
(Georgiev et al., 2009)	89.40
Our model	92.20

Table 5: Comparing the previous state-of-the-art results to our best morphologically informed Bi-LSTM-CRF model.

In Table 4, we see how the entities map to the different POS groups. It appears that knowing the concrete tag of each entity can help us improve the performance by almost one percent.

Table 5 shows our best result compared to the previous state-of-the-art result as reported in (Georgiev et al., 2009). They achieved an F1 score of 89.4% by sing regular expressions, gazetteers and non-local morpho-syntactic characteristics. Our model improves this to 92.20% without using any external resources.

More detailed evaluation results for our best model are presented in Table 6, where we show the precision, recall and F1 score for each type of named entity. We can observe relatively worse F1 score of 84.70 for Organization compared to 95.86 for Location and 94.95 for Person. We explain this drop in F1 score by the fact that many organizations are named after persons.

Entity	Precision	Recall	F1
Location	97.75	94.05	95.86
Person	95.67	94.23	94.95
Organization	75.57	96.34	84.70
Miscellaneous	96.15	22.73	36.76
Overall	93.31	91.12	92.20

Table 6: Detailed results for the different kinds of named entities.

Model	F1
(1) LSTM-CRF (words only)	82.03
(2) fwd-LSTM-char + (1)	85.15
(3) bwd-LSTM-char + (1)	85.40
(4) Bi-LSTM-char + (1)	86.44
(5) POS11 + (4)	90.96
(6) Morph + (5)	91.03
(7) POS3 + (6)	92.20

Table 7: The impact of different components and different component combinations on the performance of our best model.

Table 7 shows the cumulative effect of adding different components to our model. The basic model we started with is shown on line (4). Then, on lines (1)-(3) we remove different components from this basic model, and on lines (5)-(7) we add POS and morphological information to it. We can see sizeable improvement for the standard Bi-LSTM-CRF model with only word vector representationa and the model with character-level LSTM. Interestingly, there is almost no difference between the suffix and the prefix vectors. We can further see that adding POS11 (5) vector improves the performance by almost four percent absolute. The morphological vectors and POS3 also improved the F1 score to 92.20 points absolute. These improvements show that using known linguistic knowledge such as grammatical features could improve the representation vectors learned over huge text corpora. From the point of view of feature learning, we speculate that vectors trained over texts in morphologically rich languages do not learn enough grammar such as POS and morphology. The character embeddings also seem not to help much. One explanation for this could be that the suffixes and prefixes in Bulgarian are also highly ambiguous.

Model	Word (Bulgarian / English)
Model	Еминем / Eminem
Model + POS11	Фердинанд / Ferdinand
Model + POS2	Ваксберг / Vaksberg
Model + POS3	Гьоте / Goethe
Model + POS4	Обзървър / Observer
Model + POS5	Шехеразада / Scheherazade
Model + POS11 + POS3	Ингмар / Ingmar

Table 8: Examples of words which could not be handled correctly by the specific configuration

6 Error Analysis

Our manual analysis of the errors shows that one of the main reasons for our model to work better when POS tags are provided is due to the presence of many loanwords in the Bulgarian text. The LSTM-CRF model manages to learn the grammar of the language itself, but it needs additional help with words borrowed from other languages.

A common problem for the LSTM-CRF model is the mislabeling of foreign person or organization names. In such cases, the POS tags help by suggesting the possible part-of-speech for each word. In our test set, around 10% of the wrongly labeled words are loan words, borrowed primarily from English, Russian, German and Turkish. Table 8, shows some examples of words that could not be handled properly.

The loanwords cannot be successfully recognized by our algorithm in all cases. Even though some people try to write them in Cyrillic, their structure is different from the typical structure of the Bulgarian words. That is why, we further tried to add gazetteers and lexicons with existing loan words in Bulgarian. Similarly to the way we added POS vectors, we created a one-hot vector for each word that says whether that word is part of our lexicon with loan words or not. We then concatenated the vector to the rest of the word embeddings. However, this approach did not result in significant improvements because new words are added to the language almost every day, and it is impossible to capture them all.

Sometimes, the loanwords come in the Latin alphabet, as they are spelled in their original language. For such cases, we added a feature to the model that captures the information whether the words are in Latin or in Cyrillic. This feature, by itself, did not make much of a difference. Yet, we plan to explore it further in our future work.

	B-ORG	I-ORG	B-PER	I-PER	B-LOC	I-LOC	B-MISC	I-MISC	O
B-ORG	**258**	0	1	0	2	0	6	0	6
I-ORG	0	**31**	0	0	0	0	0	0	1
B-PER	9	1	**1,169**	6	8	0	7	0	58
I-PER	0	1	15	**595**	0	0	0	0	6
B-LOC	15	0	8	0	**676**	0	4	0	36
I-LOC	1	0	0	1	3	**36**	0	0	1
B-MISC	27	0	2	0	1	0	**39**	0	41
I-MISC	0	1	0	0	0	0	0	**1**	0
O	12	2	10	22	2	0	3	0	**57, 522**

Table 9: Confusion matrix for our best model on the test dataset: the columns represent the true labels and the rows show the predictions.

Table 9 shows a confusion matrix for the nine BIO tags that we used for the four kinds of named entities that we are recognizing. In the table, the columns represent the actual expected gold tags, while the rows show the predictions of our model. There are several interesting observations that we can make about this confusion matrix. First, it looks like the biggest problem for the model is with the tag B-PER, which is often confused with the tag O, i.e., Outside. This is probably due to the fact that in Bulgarian the first names sometimes have more than one meaning, which can confuse the model. The same argument holds for the tag B-LOC, which is also often confused with the tag O. Another place for improvements would be to distinguish better between B-LOC and B-ORG, as many places and organization have identical names, or at least the identical first words. The miscellaneous entities such as the names of books or movies can also have names that are identical to those of some organizations. Even more often, Miscellaneous entities could be confused with the Other category as they contain many common Bulgarian words.

7 Conclusion and Future Work

We explored the potential of using morphological information to a recurrent Bi-LSTM-CRF neural network architecture with the aim to improve named entity recognition for morphologically rich languages such as Bulgarian, which pose different challenges for named entity recognition compared to English. Our experiments have shown that adding morphological and part-of-speech information to the model's input yields sizable performance gains over a model that only relies on word-level and character-level embeddings as an input to the neural network.

In future work, we plan to extend the modeling of the morphological structure of the entities. Here, we only used a limited number of features, namely gender, number and definiteness, but it might be interesting to add the full linguistic knowledge encoded in the BulTreebank. We further plan to explore features and models that can help identify loan words in Bulgarian.

Another promising research direction is to compare the differences in the graphical representation of named entities in Bulgarian and English. For example, in English all components of a named entity are capitalized (except for the functional words). In order to have comparable data, we envision to pre-transform the Bulgarian dataset to which to apply the English capitalization rule for the phrasal named entities.

Finally, we plan to experiment with different monolingual representations from ElMo (Peters et al., 2018), BERT (Devlin et al., 2019), ROBERTa (Liu et al., 2019c), XLNet (Yang et al., 2019), and Ernie 2.0 (Sun et al., 2019), pooled representations from Flair (Akbik et al., 2019), distilled representations from MT-DNN (Liu et al., 2019a,b) or cross-language representations from XLM (Lample and Conneau, 2019).

8 Acknowledgements

This research was partially supported by *the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH – CLaDA-BG*, Grant number DO01-164/28.08.2018

We would like to thank the anonymous reviewers for their constructive comments, which have helped us improve the paper.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, MN, USA, NAACL-HLT '19, pages 724–728.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI international FASTUS system: MUC-6 test results and analysis. In *Proceedings of the 6th Conference on Message Understanding*. Columbia, MD, USA, MUC6 '95, pages 237–248.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, NAACL-HLT '2019, pages 4171–4186.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*. Lisbon, Portugal, LREC '04.

Georgi Georgiev, Preslav Nakov, Kuzman Ganchev, Petya Osenova, and Kiril Simov. 2009. Feature-rich named entity recognition for Bulgarian using conditional random fields. In *Proceedings of the International Conference on Recent Adcances in Natural Language Processing*. Borovets, Bulgaria, RANLP '09, pages 113–117.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, CA, USA, NAACL-HLT '16, pages 1296–1306.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics*. Copenhagen, Denmark, COLING '96, pages 466–471.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

John Lafferty. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, ICML '01, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, CA, USA, NAACL-HLT '16, pages 260–270.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *CoRR* abs/1901.07291.

The Anh Le, Mikhail Y. Arkhipov, and Mikhail S. Burtsev. 2018. Application of a hybrid Bi-LSTM-CRF model to the task of Russian named entity recognition. In Andrey Filchenkov, Lidia Pivovarova, and Jan Žižka, editors, *Artificial Intelligence and Natural Language*. Springer International Publishing, pages 91–103.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, EMNLP '15, pages 1520–1530.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *CoRR* abs/1904.09482.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *CoRR* abs/1901.11504.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019c. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR* abs/1907.11692.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. Ann Arbor, MI, USA, CoNLL '14, pages 78–86.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, EMNLP '14, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. New Orleans, LA, USA, NAACL-HLT '18, pages 2227–2237.

Jakub Piskorski, Peter Homola, Małgorzata Marciniak, Agnieszka Mykowiecka, Adam Przepiórkowski, and Marcin Woliński. 2004. Information extraction for Polish using the SProUT platform. In Mieczysław A. Kłopotek, Sławomir T. Wierzchoń, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining*. Springer Berlin Heidelberg, pages 227–236.

Jakub Piskorski, Laska Laskova, Michał Marcińczuk, Lidia Pivovarova, Pavel Přibáň, Josef Steinberger, and Roman Yangarber. 2019. The second cross-lingual challenge on recognition, normalization, classification, and linking of named entities across Slavic languages. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. Florence, Italy, BSNLP '19, pages 63–74.

Jakub Piskorski, Lidia Pivovarova, Jan Šnajder, Josef Steinberger, and Roman Yangarber. 2017. The first cross-lingual challenge on recognition, normalization, and matching of named entities in Slavic languages. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*. Valencia, Spain, BSNLP '17, pages 76–85.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Vancouver, Canada, EMNLP '05, pages 339–346.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR* abs/1402.1128.

Iman Saleh, Scott Cyphers, Jim Glass, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, and Preslav Nakov. 2014. A study of using syntactic and semantic structures for concept segmentation and labeling. In *Proceedings of the 25th International Conference on Computational Linguistics*. Dublin, Ireland, COLING '14, pages 193–202.

Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004a. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation, Special Issue*. Kluwer Academic Publishers, pages 495–522.

Kiril Simov, Petya Osenova, and Milena Slavcheva. 2004b. BTB-TR03: BulTreeBank Morphosyntactic Tagset. BulTreeBank Project, IICT-BAS.

Jana Straková, Milan Straka, and Jan Hajič. 2013. A new state-of-the-art Czech named entity recognizer. In Ivan Habernal and Václav Matoušek, editors, *Text, Speech, and Dialogue*. Springer Berlin Heidelberg, pages 68–75.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, EMNLP '17, pages 2670–2680.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. ERNIE 2.0: A continual pre-training framework for language understanding. *CoRR* abs/1907.12412.

Charles Sutton and Andrew McCallum. 2012. An introduction to conditional random fields. *Found. Trends Mach. Learn.* 4(4):267–373.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning*. Taipei, Taiwan, COLING '02, pages 1–4.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning*. Edmonton, Canada, CoNLL '03, pages 142–147.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. 2017. The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the Conference on Neural Information Processing Systems*. Long Beach, CA, USA, NIPS '17, pages 4151–4161.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR* abs/1906.08237.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, PA, USA, ACL '02, pages 473–480.

Named Entity Recognition in Information Security Domain for Russian

Sirotina Anastasiia
Lomonosov Moscow State University
(Russia)
overnastuhed@yandex.ru

Loukachevich Natalia
Lomonosov Moscow State University
(Russia)
louk_nat@mail.ru

Abstract

In this paper we discuss the named entity recognition task for Russian texts related to cybersecurity. First of all, we describe the problems that arise in course of labeling unstructured texts from information security domain. We introduce guidelines for human annotators, according to which a corpus has been marked up. Then, a CRF-based system and different neural architectures have been implemented and applied to the corpus. The named entity recognition systems have been evaluated and compared to determine the most efficient one.

1 Introduction

For a cybersecurity expert, it is vital to stay aware of all of the newly discovered vulnerabilities and exploits. Although various security databases, such as National Vulnerability Database (NVD) or MS Bulletins, contain detailed information on various cybersecurity problems, they do not usually provide any data on the latest discoveries. The most up-to-date descriptions of vulnerabilities are usually posted on specific websites and forums in form of **unstructured informal** texts. Therefore, a system extracting relevant cybersecurity information from unstructured publications could be of great use for a cybersecurity expert.

As there are a lot of NER systems for news documents, the first idea one comes up with is to apply such a system to the cybersecurity domain. Unfortunately, all such attempts prove to be unsuccessful for two reasons. Firstly, there is a great difference between general news documents and informal publications on cybersecurity: the latter contain a lot of non-vocabulary words (such as jargonisms, borrowings and domain-specific terms), include various grammar and spelling mistakes and use spoken syntax. What is more, changing the domain, where the extraction is conducted, usually means that some new named entity types should be accounted for (e.g. for cybersecurity domain these are names of programs and viruses). As none of the existing NER systems can be applied to the cybersecurity domain, our only option is to create a new system, which is trained on a newly labeled corpus, where all the domain-specific names and terms are taken into account.

This paper observes the named entity recognition (NER) task for unstructured Russian texts related to cybersecurity. Our first step is a thorough analysis of unstructured texts related to cybersecurity, elaboration of guidelines for human annotators and, finally, corpus labeling. At the second step, we implement several NER systems (one based on CRF-method and others based on artificial neural networks), apply them to the corpus, evaluate and compare of the results shown by the systems.

2 Related Work

The information extraction task in cybersecurity domain has been discussed in several works. However, the vast majority of the works consider information extraction only from structured or semi-structured English texts. For instance, (Bridges et al., 2013) and (Weerawardhana et al., 2014) use training corpora consisting of MS Bulletins and NVD vulnerability descriptions mainly. The training corpus presented in (Joshi et al., 2013) does contain unstructured blog posts, but those comprise less than 10% of the corpus.

NER systems elaborated in the works mentioned above are based on several different methods such as principle of Maximum Entropy in (Bridges et al., 2013), Conditional Random Fields (CRF) in (Weerawardhana et al., 2014: Joshi et al., 2013).

One of the latest works on the topic is (Gasmi et al., 2018). The authors use the corpus created by Bridges et al. (2013) as a dataset for two different NER systems. The corpus of over 850 000 tokens includes a large amount of NVD

Proceedings of Recent Advances in Natural Language Processing, pages 1114–1120,
Varna, Bulgaria, Sep 2–4 2019.

descriptions and also some MS Bulletins and Metasploit Framework's descriptions. The corpus was auto-labeled by a special algorithm. The labeling scheme is BIO. The two NER systems trained by Gasmi et al. (2018) are a CRF-model (CRFSuite implementation by Okazaki (2007)) and a neural network (NN) based model LSTM-CRF (as suggested by Lample et al. (2016)). The NN-based model combines bidirectional LSTM, word2vec models as a source of pre-trained word embeddings and CRFs as an output layer.

The only work that discusses the NER task for unstructured Russian text from cybersecurity domain is (Mazharov and Dobrov, 2018). The authors train their NER system on an early version of Sec_col collection, which was annotated with a CRF-classifier trained on news documents (Mozharova and Loukachevitch, 2016a).

3 Labeled Corpus Construction

The source of the text for our corpus is Sec_col collection. It consists of 2000 texts (posts and forum publications) from SecurityLab.ru website. These texts can be described as follows: they do not include any structured information; the writing style is informal; they contain quite a lot of lexical, spelling and grammar mistakes, many borrowings, words in foreign languages, words containing non-alphabetic characters and many instances of jargon. The average length of the texts in Sec_col collection is about 400 words.

The texts were manually marked up by four independent annotators, not all of which are cybersecurity experts. For the annotation purpose, the BRAT web based annotation tool was used.

There is no conventional set of labels for texts related to cybersecurity, but different authors usually consider quite similar classes of named entities to be relevant for the cybersecurity domain. For instance, both Bridges et al. (2013) and Joshi et al. (2013) annotate such types of named entities as software names, software versions, file names, vulnerability names. Taking into account the experience of the works mentioned above, we propose the following set of labels to annotate named entities: **Person** – people's names; **Loc** – locations; **Org** – names of organizations; **Hacker** – individual hackers' nicknames; **Hacker_Group** – names of hacker groups; **Program** – software products and parts of programs; **Device** – electronic gadgets; **Tech** – technologies; **Virus** – various malicious software; **Events** – for example, conferences.

At first, annotators had quite modest instructions on how various named entities must be annotated. Our assumption was that this would help to detect the trickiest contexts and regular mistakes that should be mentioned in the full-fledged guidelines for annotators.

In total 1124 publications have been marked up. Then those texts that do not contain any named entities that are relevant for cybersecurity were excluded. The final corpus contains 861 texts, which is more than 400 000 tokens.

In order to ascertain the degree of correctness and consistency of the annotation, we conducted a thorough analysis of the labeled texts. As the result, we found out that annotators tend to mark up similar contexts quite differently. For example, token *ICQ* was labelled as **Program** 18 times and received label **Tech** in 9 other cases.

Also some other instances of inconsistent markup were discovered:

- The number of named entities (one or two) in the contexts where an abbreviation is followed by the full name of the same entity or vice versa: *Software Defined Network (SDN)*; *MITM (Man-in-the-Middle)*;

- The number of named entities (one or two) in the contexts where a name or term in Russian is followed by the same name in English;

- Presence or absence of a named entity on a version of software or device when it is separated from the name by a punctuator or a conjunction: *Android 7.1, 7.1.1;*

- Inclusion or exclusion of paired punctuators (such as quotation marks or brackets) that surround a named entity: *сканер уязвимостей (nessus)* - (scaner ujazvimostej nessus) "vulnerability scanner nessus";

- Inclusion or exclusion of the second part of a compound noun with a hyphen, the first part of which is a named entity: *Android-устройств* – (android ustrojstv) "devices run on Android".

The vast variety of mistakes and instances of inconsistency reveal that there is a strong need for

Labels	Org	Loc	Person	Tech	Program	Device	Virus	Event	Hacker_group	Hacker
Amount of entities	3797	1553	1130	3280	3995	586	561	310	45	16

Table 1: Statistics of the annotated named entities.

guidelines for human annotators, whether they are cybersecurity experts or not.

3.1 Guidelines for Annotators

Within our study, comprehensive annotators' guidelines have been developed. It includes description of all the problematic and/or ambiguous contexts and indicates how named entities should be annotated in these contexts. Some passages from the guidelines are provided hereunder.

- If an abbreviation is followed by the full name of the same entity or vice versa, then the whole sequence is annotated as a single NE;
- If a name or a term in Russian is followed by the same name in English, then the whole sequence is annotated as a single NE;
- If a version of a software or a device is separated from its name by a punctuator or a conjunction, then it is annotated as a named entity only if it contains an alphabetic character, for example: *PowerPC G3, G4;*
- Paired punctuators (such as quotation marks or brackets) that surround a named entity are included in its annotation;
- The second part of a compound named entity with a hyphen is included in its annotation.

The guidelines also include detailed labels' description.

For each label we indicate the types of named entities that should receive this label. For example, label **Program** should be assigned to: operating systems (*iOS 9*); browsers (*Google Chrome*); programs that can be downloaded and installed (*Adblock*); websites (e.g. *SlideShare*, but not a link: *https://www.slideshare.net/);* files and processes, whose names are in format «name.extension» (*Autorun.exe*, *ipfilter.dat*) and some others. Likewise, label **Tech** should be assigned to: formats and filename extensions (when mentioned separately, e.g. *XML*); programming languages (*Javascript*); protocols (*pptp*), standards (*PCI DSS*) and some others.

For each label we also give the types of named entities that should not receive this label. For example, label **Program** should not be assigned to: links and directories, long instances of programmers' code, some abbreviations (*OS*). Likewise, label **Tech** should not be assigned to: abbreviations such as *BYOD, CYOD*, which are not technological, but business approaches.

The annotation of the corpus was corrected in accordance with the guidelines. To access how useful the introduction of the guidelines is, we could compare agreement between annotators before and after the guidelines were introduced. Unfortunately, at the current point annotator's agreement cannot be measured as there are no texts for which we would have several annotations from different annotators.

In Table 1 the statistics of the annotated named entities in the renewed corpus is presented.

The renewed corpus was used to train and test several NER systems based on the modern machine learning methods.

4 Labeling Scheme

In recent works on NER, several specific labelling schemes are usually used: IOB, BIO or IOBES. The general idea is to enrich standard labels with prefixes that indicate position of a token inside of a named entity: for example, whether it is the first word of a named entity.

It was shown that BIO-scheme is the one most efficient tagging schemes for NER system training: see (Mozharova and Loukachevitch, 2016a) for CRF-model and (Reimers and Gurevych, 2017) for neural networks. BIO-scheme suggests marking the first token of any named entity with a B- tag (beginning), every other token within the named entity should receive an I- tag (inside). Every token that is not a part of a named entity re-

List name	Examples of objects	Amount of objects
Device_name	*Macbook, Em Marine*	54
Device_type	*лэптоп* (laptop, “laptop”), *плеер* (player, “player”)	57
Hacker_descr	*хакер* (hacker, “hacker”), *онлайн-вор* (online-vor, “online-thief”)	31
Hacker_name	*UGNazi, AnonCoders*	39
Org_name	*ABBYY, ZYXEL*	306
Org_type	*холдинг* (holding, “corporate group”), *лаборатория* (laboratoriya, “laboratory”)	46
Program_name	*Amazon, Blackberry*	335
Program_type	*firewall, antispy*	167
Tech_name	*CD, SSH*	195
Tech_type	алгоритм (algoritm, „algorithm“), протокол (protokol, “protocol”)	19
Virus_name	*MITM, NonPetya*	42
Virus_type	Trojan, Malware	179

Table 2: Lists information.

ceives tag O (outside).

5 CRF-Model

CRF-model is a discriminative classifier, which can be used to predict sequences (Lafferty et al., 2001). A CRF-classifier uses information about the whole sequence of observable states (words) and information from previous unobservable states (labels). CRFs proved to be one of the most successful methods for NER.

To test a CRF-model, we use an open source implementation CRF++[1].

The following set of tokens’ features was used for the model training:

- **String features**: length of the token; initial letter’s case; presence of non-alphabetic characters; presence of a vowel, etc.
- **Token’s lemma**;
- **Part of speech (POS) of a token**;
- **Lexicon features**: whether a token is mentioned in a special vocabulary list (see below);
- **Cluster feature**: a number of the token’s cluster (see below);
- **Context features**: all the features mentioned above for the two preceding tokens and for the two following tokens;
- **Bigram feature**: previous token’s label.

To determine lemma and POS of each token, an open source morphological analyzer MyStem[2] was used. For tokens that do not have any lemma (for example, punctuation marks), lemma feature coincides with the token. For tokens that are absent in MyStem’s vocabulary, POS feature gets value ‘N\A’ or ‘PUNCT’, if a token is a punctuation mark.

We explain lexicon and cluster features in more details.

5.1 Lexicon Features

To improve NER system’s results we can use vocabularies that contain lists of objects of a certain type. By object we mean a word or a phrase. Words can also be treated as single-word phrases.

At our disposal there are 12 lists (see Table 2 for more detailed information). To create these lexicons, a large collection of texts from different sources on cybersecurity was used. Mutli-word terms and names were extracted from the collection. Then the extracted objects were manually classified into several categories.

Each token has 12 lexicon features. For a token T, a lexicon feature gets value ‘0’ if there is no word T or phrases that include T in the correspondent list. If token T is included in some phase in the list, then the correspondent lexicon feature gets value which equals the matched phrase’s length.

Let us consider an example. For token *Google* (as a part of a named entity *Google Chrome*), the feature that corresponds to Org_name list will get

[1] https://taku910.github.io/crfpp/

[2] https://yandex.ru/dev/mystem/

value ‘1’, while the feature that corresponds to Program_name list will get value ‘2’.

5.2 Cluster Feature

To form word clusters, we use an open source model ruwikiruscorpora_upos_skipgram_300_2_2019 (RusVectōrēs[3], (Kutuzov et al., 2016)). In total 300 clusters were formed. For each token the cluster feature get value that equals the number of the cluster, that contains the token. If there is no such a cluster, then the feature gets value ‘-1’.

6 Neural Networks

Nowadays the most successful NER systems are usually those that are based on neural networks.

Within our study, six different neural network architectures were implemented:

(A) BiDirectional LSTM: **BiLSTM**;

(B) BiDirectional LSTM with a CRF-classifier as an output layer: **BiLSTM-CRF**;

(C) BiDirectional LSTM with BiDirectional LSTM embeddings: **BiLSTM$_{\mathbf{CHAR}}$-BiLSTM**;

(D) BiDirectional LSTM with BiDirectional LSTM embeddings and a CRF-classifier as an output layer: **BiLSTM$_{\mathbf{CHAR}}$-BiLSTM-CRF**;

(E) BiDirectional LSTM with CNN embeddings: **CNN$_{\mathbf{CHAR}}$-BiLSTM**;

(F) BiDirectional LSTM with CNN embeddings and a CRF-classifier as an output layer: **CNN$_{\mathbf{CHAR}}$-BiLSTM-CRF**;

The core layer in all the architectures is Bidirectional Long-Short Term Memory (BiLSTM) NN (Graves et al., 2013), (Huang et al., 2015). BiLSTM is capable of learning long-term dependencies and considers both left and right context of every token.

All the architectures use pre-trained word embeddings created by model araneum_none_fasttextskipgram_300_5_2018 (RusVectōrēs, (Kutuzov et al., 2016)). Fasttext models are able to build embeddings for non-vocabulary words (e.g. jargonisms or borrowings), which is vital for a big corpus like ours.

Models (B), (D) and (F) use a CRF-classifier as an output layer (Huang et al., 2015), (Lample et al., 2016), (Ma et al., 2016). Therefore, these models are capable of learning standard constraints of the markup such as that a token with I-label must always follow a token with B-label of the same class.

Models (C)-(F) also have special layers that build character embeddings (Lample et al., 2016), (Ma et al., 2016), which is said to improve the results shown by NER systems (Reimers and Gurevych, 2017), (Zhai et al., 2018). While models (C)-(D) use BiLSTM-layer to build character embeddings, models (E)-(F) use CNN-layer for the same purpose. Reimers and Gurevych (2017) and Zhai et al. (2018) have shown that both layers provide the same improvement of a NER system, but CNN-layer is characterized by higher computational efficiency.

7 Evaluation

To evaluate all the NER systems, standard metrics such as Precision, Recall and F-score were used for each named entity type (label).We also compute macro and micro metrics for each system. The evaluation method is as follows: a named entity considered to be correctly identified by a NER system (true positive decision) only when both the type and the boundaries are correctly defined. This method is called exact (full) matching method.

To calculate the metrics the 3:1 cross validation technique was used.

Table 3 presents Precision for all the systems, while Table 4 and Table 5 present Recall and F-measure, respectively. The letters in the head of the tables stand for NN-based models (as it was introduced in Section 6). We also use following notations: **P** for **Person** ; **L** for **Loc**; **O** for **Org**; **H** for **Hacker**; **Hg** for **Hacker_Group**; **Pr** for **Program**; **D** for **Device**; **T** for **Tech**; **V** for **Virus**; **E** for **Events**; **Ma** for macro measures; **Mi** for micro measures.

As we can see, the CRF-model outperforms all the NN-based models. A possible explanation could be the fact that only CRF-model uses lexicon features.

As far as NN-based models are concerned, BiLSTM$_{\mathrm{CHAR}}$-BiLSTM-CRF proves to be the most successful one, judging from micro and macro metrics. CNN$_{\mathrm{CHAR}}$-BiLSTM-CRF shows quite similar results and, as it was expected, it

[3] http://rusvectores.org/ru/about/

	CRF	(A)	(B)	(C)	(D)	(F)	(F)
O	**85.9**	68.7	73.0	75.3	78.1	78.3	76.4
L	**96.7**	90.2	88.1	92.7	92.9	95.5	94.6
P	85.4	28.9	61.2	79.1	**85.7**	72.8	79.2
H	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hg	**87.5**	0.0	0.0	0.0	0.0	0.0	0.0
Pr	82.1	56.6	65.1	77.6	**85.8**	71.4	78.5
D	**65.4**	0.0	0.0	0.0	11.1	18.8	11.9
T	71.3	63.0	67.2	71.8	**77.4**	70.2	76.6
V	**68.5**	0.0	0.0	0.0	37.5	3.0	23.8
E	67.8	0.0	0.0	0.0	71.4	0.0	37.6
Ma	**71.0**	30.7	35.5	39.7	54.0	41.0	47.9
Mi	**82.2**	63.1	70.0	76.9	79.7	74.5	78.4

Table 3: Precision.

	CRF	(A)	(B)	(C)	(D)	(F)	(F)
O	65.5	30.3	38.3	62.1	**69.1**	48.6	67.5
L	81.9	39.4	53.5	70.0	**82.3**	52.5	73.5
P	**57.8**	8.9	30.0	46.9	54.7	35.0	49.1
H	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hg	**14.0**	0.0	0.0	0.0	0.0	0.0	0.0
Pr	**61.2**	29.0	40.4	51.3	60.0	57.1	58.2
D	**21.9**	0.0	0.0	0.0	0.8	2.5	0.8
T	53.6	4.1	16.8	**55.5**	41.9	48.0	53.7
V	**28.3**	0.0	0.0	0.0	5.1	0.4	3.8
E	**27.2**	0.0	0.0	0.0	5.9	0.0	7.2
Ma	**41.1**	11.2	17.9	28.6	32.0	24.4	31.4
Mi	**59.0**	21.6	31.3	51.7	55.3	45.4	55.2

Table 4: Recall.

	CRF	(A)	(B)	(C)	(D)	(F)	(F)
O	**74.3**	42.0	50.2	68.1	73.3	59.9	71.6
L	**88.6**	54.8	66.6	79.8	87.3	67.6	82.7
P	**68.9**	13.5	40.3	58.9	66.8	47.2	60.6
H	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Hg	**24.0**	0.0	0.0	0.0	0.0	0.0	0.0
Pr	70.0	38.4	49.9	61.8	**70.6**	63.4	66.6
D	**32.5**	0.0	0.0	0.0	1.5	4.3	1.3
T	61.1	13.3	26.9	62.6	54.4	57.0	**63.1**
V	**39.6**	0.0	0.0	0.0	9.0	0.7	6.6
E	**38.5**	0.0	0.0	0.0	10.9	0.0	12.0
Ma	**49.7**	16.2	23.4	33.1	37.4	30.0	36.4
Mi	**68.7**	32.2	43.3	61.8	65.3	56.4	64.7

Table 5: F-score.

outperforms BiLSTM$_{\text{CHAR}}$-BiLSTM-CRF model in training time.

As for the results shown for different classes of named entities (labels), we could also suggest several explanations. The poor quality for **Hacker** and **Hacker_group** could be explained by the small amount of named entities of these classes in the corpus (16 and 45 respectively). The reason for the low scores for **Virus** is possibly semantic heterogeneity of the class, which, according to our latest guidelines, comprises both malicious software and various technologies that hackers use (e.g. *DDos*). Therefore, **Virus** class is also semantically similar to **Tech** and **Program** classes, which could also influence the scores. As for **Event** class, it is also semantically heterogenic, as the label is assigned to both human arranged events (e.g. seminars and conferences) and historical and cultural events (e.g. holidays and wars).

Unfortunately, the results performed by our models cannot be compared to the results in (Mazharov and Dobrov, 2018) for two reasons.

Firstly, although the same text collection Seq_col was used in both studies, the markup of the collection differs significantly. Only about 300 texts from dataset in (Mazharov and Dobrov, 2018) were annotated manually and contained labeled named entities that are relevant for cybersecurity. Other 1700 texts in the dataset had poor quality automatic annotation, provided by a CRF-classifier trained on news documents. In our study the dataset contains 861 texts that were manually marked up in accordance with the annotators' guidelines.

Secondly, as it was mentioned above, we used full matching method of evaluation, whereas the method used in (Mazharov and Dobrov, 2018) is incomplete matching.

8 Conclusion and Future Work

In this paper we discuss the NER task for unstructured Russian texts concerning cybersecurity problems. Our first step is creating a labeled corpus of such texts. In order to ensure correctness and consistency of the markup, we elaborate detailed annotators' guidelines, which include description of every label and numerous examples of various tricky contexts. Within our study, the first consistently labeled corpus of unstructured Russian texts on cybersecurity was created. The corpus can now be used either as a dataset for NER systems or to conduct linguistic analysis of the text in question.

Our guidelines can be used to create new labeled corpora of texts on cybersecurity or to annotate the rest of the texts in Sec_col to increase our corpus.

We applied several NER systems based on CRF method and on NN to our corpus. The most successful is the CRF-model. Our hypothesis is that the CRF-model outperforms all the other models because it is the only one that uses lexicon features. Among NN-based models, BiLSTM$_{\mathrm{CHAR}}$-BiLSTM-CRF shows the best results.

As for the future work, the usefulness of our annotators' guidelines should be captured by comparing agreement between annotators before and after the guidelines were introduced. What is more, the set of features for the CRF-model could be widen by adding such features as text statistics or text collection statistics (Mozharova and Loukachevitch, 2016b). Furthermore, some additional features such as lexicon features and casing features could also be used to improve the performance of NN-based NER systems.

Acknowledgments

This work has been supported by the Russian Foundation for Basic Research (project 16-29-09606).

References

Robert A. Bridges, Corinne L. Jones, Michael D. Iannacone, Kelly M. Testa, & John R. Goodall, (2013). Automatic labeling for entity extraction in cyber security. arXiv preprint arXiv:1308.4941.

Houssem Gasmi, Abdelaziz Bouras, & Jannik Laval, (2018). LSTM Recurrent Neural Networks for Cybersecurity Named Entity Recognition. ICSEA 2018, 11.

Alan Graves, Abdel-rahman Mohamed, & Geoffrey Hinton, (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE. https://doi.org/10.1109/ICASSP.2013.6638947

Zhiheng Huang, Wei Xu, & Kai Yu, (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.

Arnav Joshi, Ravendar Lal, Tim Finin, & Anupam Joshi, (2013, September). Extracting cybersecurity related linked data from text. In 2013 IEEE Seventh International Conference on Semantic Computing (pp. 252-259). IEEE. https://doi.org/10.1109/ICSC.2013.50

Andrej Kutuzov, & Elizaveta Kuzmenko, (2016, April). WebVectors: a toolkit for building web interfaces for vector semantic models. In International Conference on Analysis of Images, Social Networks and Texts (pp. 155-161). Springer, Cham. https://doi.org/10.1007/978-3-319-52920-2_15

John Lafferty, Andrew McCallum, & Fernando Pereira, (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, & Chris Dyer, (2016). Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

Xuezhe Ma, & Eduard Hovy, (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354.

Ivan Mazharov & Boris Dobrov (2018). Named Entity Recognition for Information Security Domain.

Valeriia Mozharova, & Natalia Loukachevitch, (2016, April). Combining knowledge and CRF-based approach to named entity recognition in Russian. In International Conference on Analysis of Images, Social Networks and Texts (pp. 185-195). Springer, Cham. https://doi.org/10.1007/978-3-319-52920-2_18

Valeriia Mozharova, & Natalia Loukachevitch. (2016, August). Two-stage approach in Russian named entity recognition. In 2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT) (pp. 1-6). IEEE. https://doi.org/10.1109/FRUCT.2016.7584769

Naoaki Okazaki, (2007). Crfsuite: a fast implementation of conditional random fields (crfs).

Nils Reimers, & Irina Gurevych, (2017). Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. arXiv preprint arXiv:1707.06799.

Sachini Weerawardhana, Subhojeet Mukherjee, Indrajit Ray, & Adele Howe, (2014, November). Automated extraction of vulnerability information for home computer security. In International Symposium on Foundations and Practice of Security (pp. 356-366). Springer, Cham. https://doi.org/10.1007/978-3-319-17040-4_24

Zenan Zhai, Dat Nguyen, & Karin Verspoor, (2018). Comparing CNN and LSTM character-level embeddings in BiLSTM-CRF models for chemical and disease named entity recognition. arXiv preprint arXiv:1808.08450.

Cross-Family Similarity Learning for Cognate Identification in Low-Resource Languages

Eliel Soisalon-Soininen
Department of Computer Science
University of Helsinki
eliel.soisalon-soininen@helsinki.fi

Mark Granroth-Wilding
Department of Computer Science
University of Helsinki
mark.granroth-wilding@helsinki.fi

Abstract

We address the problem of cognate identification across vocabularies of any pair of languages. In particular, we focus on the case where the examined languages are low-resource, to the extent that no training data whatsoever in these languages, or even closely related ones, is available for the task. We investigate the extent to which training data from another, unrelated language family can be used instead. Our approach consists of learning a similarity metric from example cognates in Indo-European languages and applying it to low-resource Sami languages of the Uralic family. We apply two models, following previous work: a Siamese convolutional neural network (S-CNN) and a support vector machine (SVM), and compare them with a Levenshtein distance baseline. We test performance on three Sami languages and find that the S-CNN outperforms the other approaches, suggesting that it is better able to learn such general characteristics of cognateness that carry over across language families. We also experiment with fine-tuning the S-CNN model with data from within the language family in order to quantify how well this model can make use of a small amount of target-domain data to adapt.

1 Introduction

Cognate identification is a core task in the *comparative method*, a collection of techniques used in historical linguistics for the inference of language family trees, reconstruction of protolanguages, and other areas of study related to language history (List, 2013). Cognate information can also be used to improve natural language processing (NLP) applications, such as machine translation (Grönroos et al., 2018). In addition, knowledge of cognates can be useful for second-language learning (Beinborn et al., 2014).

For a subset of the world's languages, such as Indo-European, language-family trees, protolanguages, and etymological databases are well-established. However, the majority of languages have only few speakers, and such resources are scarce. Since the tasks of the comparative method are laborious to do manually, computational approaches have been taken to automatize these tasks. In addition to cognate identification, previous work addresses phonetic alignment (Kondrak, 2000; Prokić et al., 2009; List, 2013), inference of family trees (Chang et al., 2015; Jäger, 2014; Bouckaert et al., 2012), and reconstruction of proto-words (Bouchard-Côté et al., 2013).

Ideally, computational approaches to historical linguistics should be applicable to any language, even in the absence of hand-crafted resources and analyses. Recent work addressing cognate identification for *low-resource* languages assumes either the existence of high-resource relatives, to be used as training data (McCoy and Frank, 2018), or the availability of detailed dictionary definitions (St Arnaud et al., 2017).

In this paper, we address cognate identification in a scenario where we are only given a set of unannotated vocabularies from truly low-resource languages, namely South, North, and Skolt Sami of the Uralic family, without the aforementioned resources. We only assume a training dataset of example cognates in Indo-European languages, highly unrelated to our languages of interest. It might be expected that knowledge of general tendencies in patterns of correspondence between related languages, such as common phoneme substitutions, might be of some use, even

Proceedings of Recent Advances in Natural Language Processing, pages 1121–1130,
Varna, Bulgaria, Sep 2–4 2019.

Word *x*	**Word *y***	**Meaning of *x***	**Meaning of *y***
it: *notte*	es: *noche*	'night'	'night'
en: *attend*	fr: *attendre*	'attend'	'wait'
fi: *huvittava*	et: *huvitav*	'amusing'	'interesting'
en: *oath*	sv: *ed*	'oath'	'oath'
fi: *pöytä*	sv: *bord*	'table'	'table'
en: *bite*	fr: *fendre*	'bite'	'split'

Table 1: Examples of cognates, i.e. etymologically related words. The degree of similarity in form and meaning may vary quite substantially.

when searching for potential cognates in a different language family. Naturally, some knowledge of more closely related languages, or of the language pair in question, is more informative, and we attempt to quantify how well one of these models is able to make use of that.

Our aim is to investigate the extent to which a similarity learning approach, that is learning a similarity metric in a data-driven manner, is able to generalize across language families. We experiment with two similarity learning approaches from previous work, namely a support vector machine (SVM, Hauer and Kondrak, 2011) and a Siamese convolutional neural network (S-CNN, Rama, 2016), compared with a Levenshtein distance baseline (LD, Levenshtein, 1966). We train the models on examples of cognates in Indo-European language pairs, then test how well they are able to identify cognates in the Sami language pairs, not seen at training time. In addition, we fine-tune the S-CNN model on labelled target-language pairs, in order to quantify how much the lack of target-family training data affects performance.

Next, we explain the cognate identification problem and its difficulties, and review previous approaches to the problem. Then we present the approaches we use in our experiments, as well as the experimental setup in more detail. Finally, we analyse the results of the experiments.

2 The Cognate Identification Problem

The term *cognate* has several distinct uses in the literature. In historical linguistics, two words are considered cognates only if they have descended from the same ancestor word in a common proto-language, implying that they also belong to two related languages (e.g. Jäger et al., 2017; List, 2013; Kondrak, 2009). Meanwhile, a number of broader definitions have been used in NLP, motivated by practical concerns. For example, some authors refer to any etymologically related pair of words (i.e. sharing a common origin) as cognates, including, for example, loanwords (e.g. Kondrak, 2001; Beinborn et al., 2013; Bloodgood and Strauss, 2017). Others assume that cognates share both a similar form and common meaning (e.g. Nakov and Tiedemann, 2012; Bergsma and Kondrak, 2007). This assumption is problematic for historical linguistics, since it excludes cognate words that have come to have different meanings since the languages diverged, but it may be more useful for some language learning applications. In this paper, we regard any pair of etymologically related words as cognates, including genetically related true cognates as well as direct loanwords or loans from a common origin.

We formulate the cognate identification problem as follows. We are given two string sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. The task is to extract those pairs (x, y) in relation R:

$$R = \{(x, y) \in X \times Y | \; x \text{ is cognate with } y \}.$$

Each element $x \in X$ and $y \in Y$ is a string over alphabets Σ_x and Σ_y respectively. The alphabets do not necessarily overlap, since the orthographies of different languages may vary. This issue is often circumvented by using phonetic transcriptions of words. Lacking phonetic transcriptions for our test data, we deal with orthographic forms. Nonetheless, orthographic similarity often reflects phonetic similarity, in particular in the Sami languages we take as examples here, where the orthography is largely phonemic.

Several factors have been found to predict cognateness: phonetic similarity (reflected by orthographic similarity), semantic similarity, and the presence of *regular sound correspondences*, word segments regularly occurring in similar phonetic positions and contexts (Kondrak, 2009).

The example cognates in Table 1 illustrate the difficulty of cognate identification. A straightforward example is the Italian–Spanish pair *(notte, noche)*, with a similar form and common meaning. However, many cognates have similar surface forms, but differ in meaning, such as the English–French *(actual, actuel)* and Finnish–Estonian *(huvittava, huvitav)*. Such words are referred to as *false friends* in the context of language learning.

Furthermore, cognates might look very different on the surface. English–Swedish cognates *(oath, ed)* and Finnish–Swedish *(pöytä, bord)* look quite different, but share a meaning (and common origin). On the other hand, English–French *(bite, fendre)* are similar neither in form nor meaning. The only way to recognise such cognates from their surface forms alone is to identify regular correspondences, such as *th – d* for English–Swedish.

Consequently, and in contrast to much previous work, we make no strict assumptions about the degree of similarity in form or meaning that any two cognates should exhibit. Instead, following Jäger (2014), we treat regular correspondences as the main driving factor in the cognate relation and attempt to capture these in a completely data-driven manner.

3 Related Work

Earlier computational approaches to cognate identification attempt to design a string similarity (or distance) metric that assigns a higher score to cognate words and a lower score to unrelated ones. A common approach is to extend the traditional Levenshtein distance (Levenshtein, 1966) by associating specific weights to pairs of symbols using linguistic knowledge (e.g. Kondrak, 2000; List, 2013), or sets of example cognates (e.g. Bergsma and Kondrak, 2007; Rama, 2015).

Kondrak (2000) proposes the ALINE algorithm using specific weights based on several predetermined phonetic features. In addition, Kondrak (2005) generalizes the Levenshtein distance with the n-gram similarity measure. Turchin et al. (2010) use a heuristic based on mapping consonants to ten classes, and consider words matching in their first two consonant classes to be cognates. The SCA algorithm of List (2013) uses a larger set of sound classes and also considers prosodic aspects of words.

Other authors rely on learning regular correspondences (sometimes called *mismatches* or *substitution patterns*) from example cognates using an alignment algorithm. For example, Ciobanu and Dinu (2014) and Bergsma and Kondrak (2007) use a global alignment algorithm to align orthographic word pairs and extract substring pairs, which they use as features for an SVM. Gomes and Pereira Lopes (2011) use the same approach to develop a weighted string similarity metric for words in orthographic form. Rama (2015) use gap-weighted subsequences as features. McCoy and Frank (2018) use character embeddings and cosine similarity to extend Levenshtein distance.

Hauer and Kondrak (2011) convert word pairs into features for an SVM using a set of string similarity metrics. This approach has been extended with features for semantic similarity, for example using the lexical database WordNet (Jäger et al., 2017; St Arnaud et al., 2017; Kondrak, 2009). Bloodgood and Strauss (2017) improve further such an SVM model using global constraints and reranking. In addition, St Arnaud et al. (2017) utilise English and Spanish word embeddings of dictionary definitions. This SVM classification approach is one of the methods that we apply to cross-language family learning.

Jäger (2014) and Rama (2016) take data-driven approaches not relying on hand-designed features. Jäger proposes a similarity metric based on weights for symbol pairs given by *pointwise mutual information*, the values for which were learned from a training set of cognate pairs. Rama applies deep learning, encoding words into a grid-like representation and applying a Siamese convolutional neural network to cognate identification for multilingual wordlists. He uses two methods to encode a phonetic symbol into vector, a one-hot encoding and one based on phonetic features, achieving better performance with one-hot encodings for two out of three language families. This approach is another method in our comparison of models for cross-language family learning.

In recent work, Hämäläinen and Rueter (2019) take an alternative approach of applying neural machine translation methods to the problem of predicting a cognate given a word in a related language. The same model could in principle be applied to the task we present here and we intend to make a direct comparison in future work.

4 Methods

In this section, we present the three approaches to solving the cognate identification problem that we have used in our experiments: a string similarity metric based on the Levenshtein distance (Levenshtein, 1966) used as a baseline, an SVM with several string similarity metrics as features (Hauer and Kondrak, 2011), and a Siamese convolutional neural network (Rama, 2016).

4.1 Levenshtein Distance–Based Similarity

The *Levenshtein distance* $d_L(s_1, s_2)$, or *string edit distance*, between strings s_1 and s_2 over an alphabet Σ is the minimum number of insertion, dele-

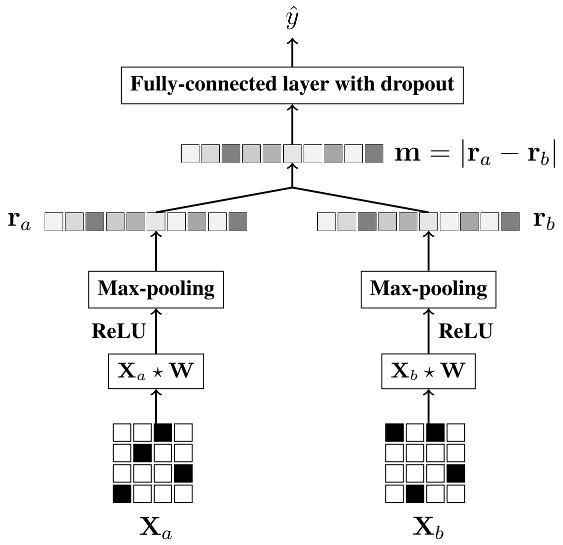

$\hat{y}$
Fully-connected layer with dropout
$\mathbf{m} = |\mathbf{r}_a - \mathbf{r}_b|$
$\mathbf{r}_a$
$\mathbf{r}_b$
Max-pooling
Max-pooling
ReLU
ReLU
$\mathbf{X}_a \star \mathbf{W}$
$\mathbf{X}_b \star \mathbf{W}$
$\mathbf{X}_a$
$\mathbf{X}_b$

Figure 1: Architecture of the S-CNN. Column vectors in input matrices represent one-hot-encoded characters. The same filter $\mathbf{W}$ is convolved with both inputs.

tion, or substitution operations needed to transform one string to the other. To obtain the *normalised* Levenshtein distance, this number is divided by the length of the longer word, equal to the maximum possible distance between s_1 and s_2. The similarity metric is then:

$$sim_L = 1 - \frac{d_L(s_1, s_2)}{max(|s_1|, |s_2|)}.$$

For example, for the cognate pairs *(coupe, Kopf)* and *(pöytä, bord)*, the respective similarities are $1 - \frac{3}{5} = 0.4$ and $1 - \frac{5}{5} = 0$. It is assumed that both strings are drawn from overlapping alphabets, since the similarity is always zero for disjoint alphabet sets.

Previous work has introduced a variety of Levenshtein-based measures by defining different ways of learning or computing the cost associated with a character substitution. Here we apply the basic version, in which a matching pair of characters have a zero cost and any other a unit cost.

4.2 Support Vector Machine

The support vector machine (SVM) is a supervised learning model trained by finding the optimal separating hyperplane between multi-dimensional data points of different classes. The basic SVM is a non-probabilistic, linear binary classifier. For data that is linearly separable, the optimal hyperplane creates the maximum margin between training points in the two classes. When the data classes are not linearly separable, the margin can still be maximised while allowing some data points to be on the wrong side of the optimal hyperplane. Another approach is to use a non-linear kernel function, which enlarges the feature space using basis expansions, such as a polynomial or a radial-basis function.

For the model comparison in our experiments, we have implemented the SVM model used by Hauer and Kondrak (2011). In this model, a pair of strings (s_1, s_2) is represented by a feature vector $\mathbf{x} \in \mathbb{R}^6$ such that

- x_1 is the Levenshtein distance $d_L(s_1, s_2)$,
- x_2 is the number of common bigrams,
- x_3 is the prefix length,
- x_4 is the length of s_1,
- x_5 is the length of s_2, and
- x_6 is the absolute difference between the lengths, i.e. $x_6 = |x_4 - x_5|$.

We have chosen this SVM model as it is based on string similarity measures that are applicable to the low-resource language setting. More recent SVM-based approaches to cognate identification exist, but they either require detailed dictionary definitions in a high-resource language with high-quality pre-trained word embeddings (St Arnaud et al., 2017), or multilingual word lists aligned by concepts (Jäger et al., 2017).

4.3 Siamese Convolutional Neural Network

The Siamese convolutional neural network (S-CNN) is a supervised learning model originally proposed by Chopra et al. (2005) for the task of face verification, and applied with some modification to cognate identification by Rama (2016). Our implementation is based on the latter model. The architecture is presented in Figure 1.

The S-CNN is a two-input version of the convolutional neural network (CNN) specialized in processing data with a grid-like topology. CNNs have been very successful in computer vision, and they have also been applied to several NLP tasks, such as text classification (e.g. Zhang et al., 2015).

When applied to NLP tasks, the CNN requires a grid-like representation of the input. In the case of cognate identification, it is convenient to represent a word as a matrix $\mathbf{X} \in \{0,1\}^{|\Sigma| \times n}$ such

Dataset	# cognate	# all pairs	$\|\Sigma\|$
IE-TRAIN	73,238	732,380	329
sma–sme	1,460	11,234 × 47,312	42 (27)
sma–sms	838	11,234 × 29,401	75 (27)
sme–sms	2,188	47,312 × 29,401	77 (38)

Table 2: The datasets used in the experiments. Etymological WordNet is used for training, and other datasets are used for testing (see Table 3 for smaller fine-tuning sets). $|\Sigma|$ is the number of all characters observed in a dataset. The number of overlapping characters is given in parentheses (for language pairs). Languages: South Sami (sma), North Sami (sme), Skolt Sami (sms).

Dataset	# cognate	# all pairs
SAMI-FT	986	100,000
SAMI-FT-TEST	3,500	350,000

Table 3: The small-scale datasets sampled from the Sami vocabularies in Table 2. We use these in experiment 2 to fine-tune the S-CNN and analyse how the number of in-family training pairs affects the performance.

that $\mathbf{X} = [\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_n]$, where each column vector $\mathbf{x}_i \in \{0,1\}^{|\Sigma|}$ is a one-hot vector representing a character in the alphabet Σ. The training data $D = \{(\mathbf{X}_{ai}, \mathbf{X}_{bi}), y_i\}_{i=1}^{N}$ then consists of pairs of words such that $y_i = 1$ if $\mathbf{X}_{ai}$ and $\mathbf{X}_{bi}$ are cognates, and $y_i = 0$ otherwise.

As shown in Figure 1, the S-CNN model is an extension of the CNN: first, one filter $\mathbf{W} \in \mathbb{R}^{p\times q}$ is convolved (cross-correlated) over character sequences of length q from both input matrices $\mathbf{X}_a$ and $\mathbf{X}_b$, producing a feature map for each input matrix. These are run through a rectified linear function, whereafter max-pooling is applied to the results. The number of rectified and max-pooled feature maps produced from each input matrix is equal to the number of filters. We fix the filter height at $p = |\Sigma|$, equal to the size of the alphabet and the height of the input matrix.

The representation vectors $\mathbf{r}_a$ and $\mathbf{r}_b$ are obtained by concatenating all the feature maps into single vectors. These vectors are then merged into one vector $\mathbf{m}$ using some distance metric. We use the absolute vector difference such that $\mathbf{m} = |\mathbf{r}_a - \mathbf{r}_b| = [|r_{a1} - r_{b1}|, |r_{a2} - r_{b2}|, \ldots, |r_{al} - r_{bl}|]^T$, where $l = |\mathbf{r}_a| = |\mathbf{r}_b|$. Finally, the merged vector $\mathbf{m}$ is fed as input to a fully-connected layer, itself connected to the output neuron. The dropout technique of Srivastava et al. (2014) is applied to the fully-connected layer, and the output neuron is activated with the sigmoid function. The output of a trained model can be regarded as a learned similarity metric between pairs of inputs.

5 Experiments

In this section, we present our datasets, experimental setup, training and fine-tuning procedures, and our evaluation scheme.

5.1 Datasets

A summary of the datasets is shown in Table 2. All source data for training and testing is publicly available and we release the exact processed training and test sets for reproducibility[1].

We use the Etymological WordNet (Gerard de Melo, 2014) as our training data for the SVM and S-CNN models. This is a database containing information of etymological origin, cognateness, as well as derivational and compositional links between words. The database consists of word pairs that each belong to one of the aforementioned relations. The database has been mined from Wiktionary, and its entries are mostly from widely spoken Indo-European languages.

Since we are concerned with the identification of cognates across languages, we only use as our training data those word pairs that are either cognates, or where one word is the root of the other. Thus, we exclude derivationally and compositionally linked word pairs from our training set. Furthermore, we filtered out those pairs where both words belong to the same language. In total, there were 73,238 cognate pairs in the filtered training set. In order to train a discriminative classifier, we generated negative examples by randomly pairing unrelated words, so that the ratio of cognate to unrelated word pairs was 10%. We refer to the resulting training set as IE-TRAIN.

[1] All datasets released at https://github.com/soisalon/LRCognates.

As a source of unannotated word lists from low-resource languages, we use a set of three vocabularies from North, South, and Skolt Sami of the Uralic family. We have retrieved these vocabularies from dictionaries compiled by Giellatekno[2]. We filtered out all words with upper-case (proper nouns) or non-alphabetic characters. We retrieved gold-standard cognate sets for evaluation and fine-tuning from Álgu[3], the etymological database for Sami languages. This database contains (positive-only) cognate information for only a subset of all the words in the vocabularies. We refer to this dataset as SAMI-FULL and average results over the three pairs of languages. The evaluation scheme is explained in detail in section 5.3.

In addition, to fine-tune and evaluate models in experiment 2 (see section 5.4), we sample small-scale sets with a higher proportion (1%) of cognates, presented in Table 3, SAMI-FT and SAMI-FT-TEST.

5.2 Training and Fine-Tuning

In our implementation of the S-CNN model, we used ten filters with width $q = 2$ and height $p = |\Sigma|$ (alphabet size). The alphabet was the set of all characters observed in both the training and test datasets, and its size was $|\Sigma| = 336$. We fixed the input matrix width to $n = 20$. For words shorter than this, the input matrices were zero-padded, and longer words were truncated at this length. In the fully-connected layer, we used a dropout rate of 0.5.

We trained the S-CNN model using binary cross-entropy as the loss function, and the Adadelta optimizer (Zeiler, 2012) with initial learning rate $\alpha = 1.0$, decay rate $\rho = 0.95$, and the constant $\epsilon = 1 \cdot 10^{-6}$. The batch size was set at 128, and number of epochs was 50. In fine-tuning (experiment 2), the respective values were 32 and 20. Otherwise, we used the same hyperparameters when fine-tuning the model. We implemented the model using the Keras library with Tensorflow backend [4].

For the SVM implementation, we used the SVM module of the Scikit-learn library for Python (Pedregosa et al., 2011), based on the C-support vector classification implementation of Chang and Lin (2011). We trained the model using a linear kernel and regularization parameter $C = 1$. For probabilistic prediction, the module uses Platt scaling (Platt et al., 1999), which is based on fitting a logistic regression on the initial binary scores using cross-validation.

5.3 Evaluation

A difficulty in evaluating on the Sami datasets is that the set of word pairs annotated as cognates in the Álgu database is known to be far from complete for the vocabularies covered. As a result, there are many word pairs in the vocabularies that are cognates, but are evaluated as unrelated. Measures such as accuracy and precision are therefore not useful for our problem setting, since we do not know whether a given word pair *not* among the annotated cognates is a cognate pair. We can, however, evaluate the *recall* of the known cognate pairs: what proportion of the annotated pairs make it into the set ranked as most likely cognates by the model. We use SAMI-FT-TEST to compute precision-recall curves for the fine-tuned S-CNN, unadapted S-CNN, SVM, and the baseline LD.

Computing scores for all pairs of words between two vocabularies is time consuming. Therefore, when evaluating on whole vocabularies, we only take those words q in vocabulary X that we know have at least one cognate in the other vocabulary Y. Then, we compute scores between each of these words, and all words in the other language. In order to evaluate these scores, we use recall@k averaged over the words q, the queries, and the set of language pairs in the test set. We call this metric the *mean average recall@k*:

$$\text{MAR@}k = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{Q}\sum_{q=1}^{Q}\text{R@}k,$$

$$\text{where R@}k = \frac{\#\text{cognates within top-}k\text{ results}}{\#\text{cognates in }Y},$$

where Q is the number of queries, and L is the number of language pairs. That is, for each word query q, we rank the pairs $(q, y_i)\forall i$ and get the top 100 words y for each q. We then compute the recall@k for $k = 1, \ldots, 100$ for q, that is, counting the cognates found within the k highest-ranked words divided by the total number of cognates for q in Y. For most words q, there is only one, and for some there are several cognates in Y.

[2] The research group of Sami language technology at the University of Tromssa. `http://giellatekno.uit.no/index.eng.html`.

[3] Available at: `http://kaino.kotus.fi/algu/`

[4] Github repository available at: `https://github.com/fchollet/keras`.

5.4 Experimental Setup

We present two experiments. We first train the SVM and S-CNN models on IE-TRAIN. (LD requires no training.)

In **experiment 1**, we apply these three models directly to the three pairs of Sami vocabularies (SAMI-FULL) to measure how well methods trained only on Indo-European data can identify cognates in Sami languages. This tells us how well the methods can exploit information from a different language family. In this experiment, we evaluate the models using the MAR@k metric (see section 5.3).

In **experiment 2**, we fine-tune the S-CNN on a small set of Sami cognates (SAMI-FT), containing example cognate pairs from all three language pairs. We test it on SAMI-FT-TEST to see how much of the performance loss from language transfer can be regained by providing the model with just a small amount of data from the target language family. We also analyse how the performance of the S-CNN improves with the amount of fine-tuning data it is given. In this experiment, we evaluate the models using precision-recall curves.

6 Results

6.1 Experiment 1: Indo-European Models for Sami Cognates

Figure 2 shows the MAR@k curves for Sami cognate identification for the three models trained on Indo-European data: S-CNN (without fine-tuning), SVM, and the baseline LD. The S-CNN outperforms the other approaches by a substantial margin, across values of k. This result suggests that the neural networks in the S-CNN are able to capture aspects of the cognateness relation that transfer across language families more effectively than the hand-designed features of the SVM. The SVM also outperforms LD – unsurprising, since the Levenshtein distance is included among its features.

Since the S-CNN performs best in this experiment, we use it in experiment 2, where we fine-tune the model on the target language family.

6.2 Experiment 2: Fine-Tuning on Target Language Family

Figure 3 shows how the number of cognate pairs used in fine-tuning improves average precision. Naturally, the average precision increases together with the number of cognates used in training. The improvement converges with about 500 training pairs, which is the number used for the fine-tuned model in Figure 4.

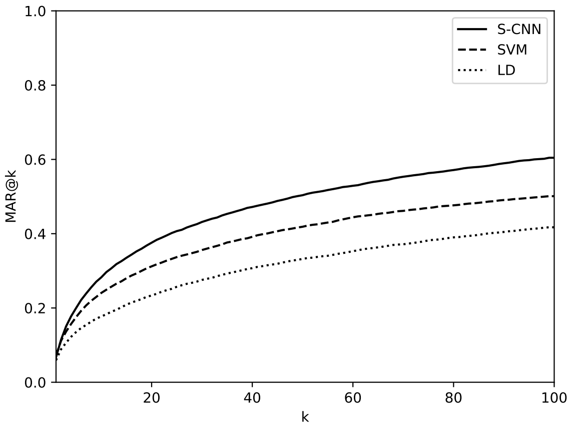


Figure 2: MAR@k for $k = 1 \ldots 100$, for SAMI-FULL, using models trained on IE-TRAIN. One curve is the average over all pairs of Sami languages.

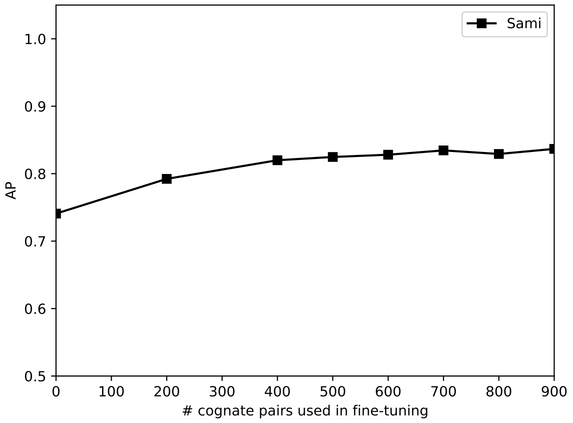


Figure 3: The learning curve of S-CNN fine-tuned on SAMI-FT, having been pre-trained on IE-TRAIN.

Figure 4 shows the precision-recall curves for each approach for the small-scale Sami test set (SAMI-FT-TEST). The corresponding values for average precision are given in Table 4. The pattern of results reflects that in Figure 2: the S-CNN outperforms the other two approaches based on string similarity metrics. The fine-tuned S-CNN substantially outperforms the untuned model. In terms of average precision, the improvement is approximately 11%.

This result tells us that, in addition to learning more general information about cognates that can be carried across language families than the SVM,

Approach	AP
S-CNN + FT	0.825
S-CNN	0.741
SVM	0.608
LD	0.540

Table 4: Average precision in the small-scale Sami test set for each approach.

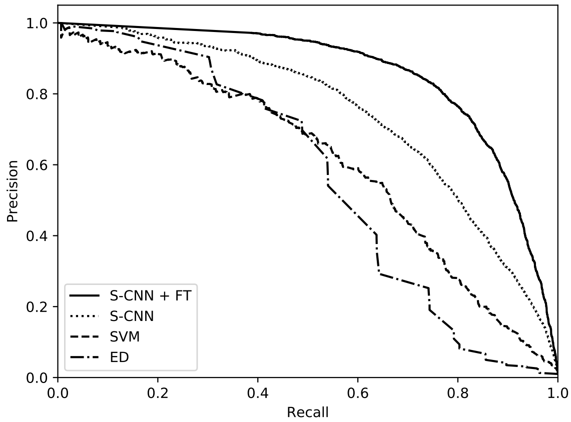


Figure 4: The precision-recall curves for each approach tested on SAMI-FT-TEST. S-CNN + FT was pre-trained on IE-TRAIN and fine-tuned on SAMI-FT. The unadapted S-CNN and SVM were trained on only IE-TRAIN.

the S-CNN is also able to make use of even a small number of annotated examples from the target languages to improve its predictions.

7 Conclusion and Future Work

We have addressed the problem of cognate identification within a set of three truly low-resource Sami languages of the Uralic family. We have examined the extent to which training data from a completely unrelated, higher-resource language family can be utilised for this task. We have taken two approaches to learn a similarity metric for cognateness from Indo-European etymological data, namely an SVM and an S-CNN, both applied to cognate identification in previous work, compared with a Levenshtein distance baseline. In addition, we have compared these with a fine-tuned S-CNN that has access to a small amount of training data in the target language family.

The results of our experiments have shown that the S-CNN is able to generalize more effectively across language families, compared with the SVM. Furthermore, a substantial improvement in performance can be attained by fine-tuning the model with only a small number of cognate examples from the target language set.

In future work, we will investigate whether language transfer for cognate identification can be further improved by making use of unsupervised multilingual character embeddings (Granroth-Wilding and Toivonen, 2019) instead of one-hot encoded characters. This could allow the model to exploit cross-lingual similarities in the usage patterns of symbols, replacing some of the manually encoded knowledge about correspondences across language pairs in previous work without the need to specify features by hand. In addition, due to the incomplete evaluation cognate sets, the experimental set-up could be complemented with a manual evaluation of top cognate suggestions in a manner similar to Hämäläinen and Rueter (2019).

Another avenue for future work is to investigate qualitatively how similar data-driven models generalize across other languages and language families, and how the choice of training language(s) affects performance. With such experimentation, we could gain more insight of what properties of sound change are carried over across families. In addition, we could investigate how the data-driven models presented here perform compared with models with more linguistically-informed handcrafted features.

Acknowledgements

This work has been funded by the Academy of Finland *Digital Language Typology* project (no. 12933481).

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Readability for foreign language learning: The importance of cognates. *ITL-International Journal of Applied Linguistics*, 165(2):136–162.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663.

Michael Bloodgood and Benjamin Strauss. 2017. Using global constraints and reranking to improve cognates detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1983–1992, Vancouver, Canada. Association for Computational Linguistics.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the indo-european language family. *Science*, 337(6097):957–960.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Will Chang, Chundra Cathcart, David Hall, and Andrew Garrett. 2015. Ancestry-constrained phylogenetic analysis supports the Indo-European steppe hypothesis. *Language*, 91(1):194–244.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546. IEEE.

Alina Maria Ciobanu and Liviu P. Dinu. 2014. Automatic detection of cognates using orthographic alignment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 99–105, Baltimore, Maryland. Association for Computational Linguistics.

Gerard de Melo. 2014. Etymological WordNet: Tracing the History of Words. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Luís Gomes and José Gabriel Pereira Lopes. 2011. Measuring spelling similarity for cognate identification. In *Progress in Artificial Intelligence*, pages 624–633, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mark Granroth-Wilding and Hannu Toivonen. 2019. Unsupervised learning of cross-lingual symbol embeddings without parallel data. In *Proceedings of the Society for Computation in Linguistics*, volume 2, pages 19–28.

Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. 2018. Cognate-aware morphological segmentation for multilingual neural translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 386–393, Belgium, Brussels. Association for Computational Linguistics.

Mika Hämäläinen and Jack Rueter. 2019. Finding Sami Cognates with a Character-Based NMT Approach. In *Proceedings of the Workshop on Computational Methods for Endangered Languages*, volume 1, pages 39–45.

Bradley Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilingual lists. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 865–873, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Gerhard Jäger. 2014. Phylogenetic Inference from Word Lists Using Weighted Alignment with Empirically Determined Weights. In *Quantifying Language Dynamics*, pages 155–204. Brill, Leiden, the Netherlands.

Gerhard Jäger, Johann-Mattis List, and Pavel Sofroniev. 2017. Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1205–1216, Valencia, Spain. Association for Computational Linguistics.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295. Association for Computational Linguistics.

Grzegorz Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *International symposium on string processing and information retrieval*, pages 115–126. Springer.

Grzegorz Kondrak. 2009. Identification of Cognates and Recurrent Sound Correspondences in Word Lists. *Traitement Automatique des Langues (TAL)*, 50(2):201–235.

Vladimir I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, 10(8):707–710.

Johann-Mattis List. 2013. *Sequence comparison in historical linguistics*. Ph.D. thesis, Heinrich-Heine-Universität Düsseldorf.

Richard T. McCoy and Robert Frank. 2018. Phonologically Informed Edit Distance Algorithms for Word Alignment with Low-Resource Languages. In *Proceedings of the Society for Computation in Linguistics*, volume 1, pages 102–112.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 301–305, Jeju Island, Korea. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Jelena Prokić, Martijn Wieling, and John Nerbonne. 2009. Multiple sequence alignments in linguistics. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*, pages 18–25. Association for Computational Linguistics.

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1227–1231, Denver, Colorado. Association for Computational Linguistics.

Taraka Rama. 2016. Siamese convolutional networks for cognate identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1018–1027, Osaka, Japan. The COLING 2016 Organizing Committee.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Adam St Arnaud, David Beck, and Grzegorz Kondrak. 2017. Identifying cognate sets across dictionaries of related languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2519–2528, Copenhagen, Denmark. Association for Computational Linguistics.

Peter Turchin, Ilia Peiros, and Gell-Mann Murray. 2010. Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3:117–126.

Matthew D. Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Automatic Detection of Translation Direction

Ilia Sominsky and **Shuly Wintner**
Department of Computer Science
University of Haifa
31905 Haifa, Israel

Abstract

Parallel corpora are crucial resources for NLP applications, most notably for machine translation. The direction of the (human) translation of parallel corpora has been shown to have significant implications for the quality of statistical machine translation systems that are trained with such corpora. We describe a method for determining the direction of the (manual) translation of parallel corpora *at the sentence-pair level*. Using several linguistically-motivated features, coupled with a neural network model, we obtain high accuracy on several language pairs. Furthermore, we demonstrate that the accuracy is correlated with the (typological) distance between the two languages.

1 Introduction

Parallel corpora are used for various purposes, including for training and evaluation of statistical machine translation (SMT) systems (Koehn, 2010). While traditional SMT systems are agnostic with respect to the direction in which the parallel corpora they are trained on were (manually) translated, several studies have shown that taking directionality into account when training SMT systems has a significant effect on the quality of the translation (Kurokawa et al., 2009; Lembersky et al., 2012, 2013; Twitto-Shmuel et al., 2015). In this paper we show the same effect also holds for neural machine translation (NMT) systems.

We address the task of determining the direction of translation given a parallel text; this is cast as a binary classification task. To strain the classifier, we focus on retaining high accuracy when the size of text chunks to be classified is minimal: single sentence pairs. This is an extremely difficult task for humans, in most cases: a single sentence pair often does not reveal any obvious signal of which of the two sentences is the original. It is also a highly challenging task for machines: Figure 1 depicts a few English-French examples of sentence pairs whose translation direction none of our classifiers predicted correctly.

We define sets of features that reflect insights drawn from Translation Studies regarding the special properties of translated texts, and in particular the *asymmetric* nature of translation (Toury, 1980, 1995; Baker, 1993). These include the tendency of translated texts to be simpler (Blum-Kulka and Levenston, 1983; Vanderauwerea, 1985; Baker, 1993; Laviosa, 1998, 2002); the tendency of translators to explicate the source text (Blum-Kulka, 1986; Baker, 1993); the different distributions of various statistical phenomena (e.g., the frequencies of function words or certain syntactic structures) between the source and the translation (Gellerstam, 1986; Blum-Kulka, 1986; Øverås, 1998; Koppel and Ordan, 2011); and *interference* of language constructions from the source to the target (Toury, 1979; Teich, 2003).

The contribution of this paper is manifold. (1) First and foremost, we introduce a method for accurately determining the translation direction of sentence pairs in parallel corpora; the method is based on the introduction of several new, linguistically motivated, types of features for this task. We show that the combination of these features outperforms the previous state-of-the-art in detection of translation direction.[1] Importantly, these features help shed light on the characteristics of translated language. (2) Furthermore, we demonstrate the robustness of our method by evaluating it on several language pairs and on three different

[1] As we explain in Section 2, a direct comparison with the state of the art is problematic as not enough detail is provided in the original publications for us to replicate existing results.

Proceedings of Recent Advances in Natural Language Processing, pages 1131–1140,
Varna, Bulgaria, Sep 2–4 2019.

English→French	Now the question is , who’s going to pay for it all ?
	La question est de savoir qui va payer .
French→English	Admit it and we will understand each other .
	Dites -le moi et on va bien se comprendre .
French→English	We should at least ensure that there is no need to produce many more reports .
	Il ne faudrait tout de même pas qu’ il y ait besoin d’ en faire de nombreux encore .

Figure 1: Some examples of sentence pairs with their translation direction

datasets. (3) We show that detecting the translation direction can indeed be used for improving the quality of both statistical and neural machine translation systems. (4) Finally, from a theoretical perspective, this work corroborates the intuitive hypothesis that the translation detection task is easier when the two languages involved are typologically more distant.

After reviewing related work in the next section, we describe our experimental setup in Section 4, and the features we used in Section 5. The results are presented and discussed in Section 6. We conclude with suggestions for future research.

2 Related Work

The differences between original and translated texts have been a major field of investigation in Translation Studies (Toury, 1980, 1995; Baker, 1995). Translated texts have unique characteristics that set them apart from texts originally written in the same language. These are not necessarily artifacts of poor translation; rather, they reflect different statistical distributions across the two genres. The sub-language of translated texts (in any language) was referred to as *translationese* (Gellerstam, 1986). The unique properties of translationese are attributed to various reasons, some of which are considered “universal” (e.g., translated texts tend to simplify the original message; they tend to use more standard language than originals), while others are related to *interference*, namely the “fingerprints” of the source language found in the translation product.

Distinguishing between original and translated texts is a classic text classification task that has been extensively addressed both with supervised machine learning (Baroni and Bernardini, 2006; van Halteren, 2008; Kurokawa et al., 2009; Koppel and Ordan, 2011; Ilisei et al., 2010; Volansky et al., 2015; Avner et al., 2016) and with unsupervised methods (Rabinovich and Wintner, 2015; Nisioi, 2015; Rabinovich et al., 2016a). The main challenge, as is usually the case in text classification, lies in the choice of features with which text chunks are represented. For the task at hand, features frequently used include function words (FW), character n-grams, part-of-speech (POS) n-grams, special sets of words such as discourse markers, etc. With the right choice of features, accuracies can reach almost ceiling levels, depending on the dataset involved.

However, the classification unit used in all the above-mentioned research was larger chunks of text, typically 2,000 tokens. The accuracy of identifying translationese has been shown to drop significantly when the size of the text chunk used for classification decreases (Rabinovich and Wintner, 2015). One of our goals in this work is to improve the accuracy of translationese detection systems with much smaller text chunks, as available parallel texts are not guaranteed to be long.

Previous research focused on identifying translationese in monolingual texts. However, in realistic scenarios, parallel texts are available and the actual task is to determine the *direction of translation* given texts in *two* languages. For such tasks one can use features drawn from each of the two languages, as well as from the alignments between words and phrases in the two texts. This approach was taken by Eetemadi and Toutanova (2014), who used the Canadian Hansard corpus of parallel texts in English and French.

The motivation stems from the observation that linguistic structures tend to have different distributions in original and translated texts. Therefore, assessing the frequencies of syntactic structures in two parallel texts, especially for text chunks that are aligned with each other across two parallel sentences, may shed light on the direction of the translation. As base structures, Eetemadi and Toutanova (2014) used *minimal translation units (MTUs)*, defined as pairs of source and target word sets that satisfy two conditions: (i) no alignment links exist between distinct MTUs; (ii) MTUs are

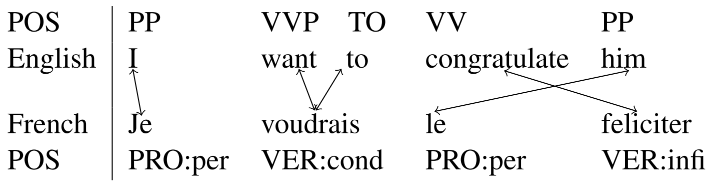


Figure 2: POS-MTUs, English–French

not decomposable into smaller MTUs without violating the previous rule. Once MTUs were identified, each word was replaced by its POS tag, thereby creating POS-MTUs. These are the structures used as features.

As an example, consider the two aligned English–French sentences in Figure 2; they yield the following POS-MTUs: [PP]↔[PRO:per], [VVP, TO]↔[VER:cond], [VV]↔ [VER:infi], and [PP]↔[PRO:per]. More specifically, the POS-MTU [VVP, TO]↔[VER:cond] reflects the fact that English word pairs such as *'want to'* translate to French verbs in the conditional form, e.g., *'voudrais'*. Incidentally, this mapping is much more common, by a factor of 10, in English-to-French translations than in the reverse direction.

As another example, the two aligned English–German sentences depicted in Figure 3 yield the following POS-MTUs: [CD]↔[PIS], [IN]↔[ART], [NP]↔[ADJA], [RB, JJS]↔[ADJA], [NNS]↔[NN]. In particular, the POS-MTU [RBS, JJ]↔[ADJA] reflects the fact that English word pairs such as *'most famous'* translate to German adjectives in the superlative form, e.g., *'berühmtesten'*.

Eetemadi and Toutanova (2014) do not provide sufficient details that would enable replication of their results, but they report 71% accuracy with these features. In a subsequent work, Eetemadi and Toutanova (2015) used Brown clusters (Brown et al., 1992), a method of clustering words according to syntactic and semantic relatedness, instead of POS tags. With *Brown cluster MTUs* as features, they reached 80% precision and 85% recall on the Hansard corpus. This is the present state of the art for this task.

3 Motivation

This work was partly motivated by previous research that demonstrated that *statistical* machine translation can be improved by training on source-translated-to-target corpora rather than target-translated-to-source texts (Kurokawa et al., 2009; Lembersky et al., 2013; Twitto-Shmuel et al., 2015). In this section we verify that such benefits hold also for *neural* machine translation (NMT). We used French–English data from three corpora (Hansard, Europarl and UN; see below). The total data that was available to us consisted of 1.6 million sentences annotated as French original, and 11.7 million sentences annotated as English original. Focusing on translating French to English, we trained three different NMT systems using Marian (Junczys-Dowmunt et al., 2018). In one system (FO), the training material consisted only of French original sentence pairs; in the other (EO), we only used English original sentence pairs; and in the third (MIX), we mixed equal portions of both. In all three cases we used an equal number of sentence pairs (1.6 million). We tested the three NMT systems on a reference set of 10,000 sentences taken from French original data, following the methodology of Lembersky et al. (2013). We evaluated the quality of the resulting NMT systems by comparing BLEU, METEOR and TER scores using MultEval (Clark et al., 2011).

The results, listed in Table 1, clearly corroborate our hypothesis: for the task of French to English translation, training data that were manually translated from French to English yield much better NMT systems than training data that were translated in the reverse direction.

Train Data	BLEU↑	METEOR↑	TER↓
FO	41.0	38.4	46.1
MIX	38.2	36.7	48.5
EO	34.4	35.0	52.8

Table 1: Accuracy of NMT systems with varying configurations of the training material

4 Methodology

Task Given a sentence pair in a parallel corpus, our task is to identify the direction of translation,

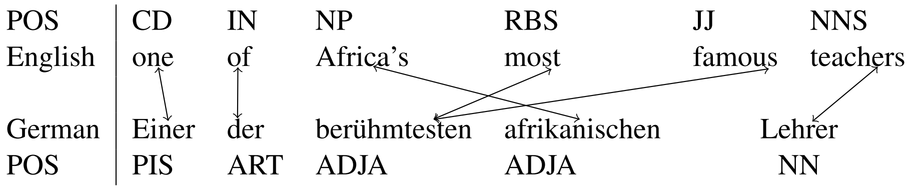


Figure 3: POS-MTUs, English–German

thereby determining the source and the target sentences. Our main challenge is to define a set of features that will yield the best accuracy.

Datasets We used sentence-aligned parallel corpora from three resources: the Canadian parliamentary proceedings (Hansard), with English–French sentence pairs; Europarl (Koehn, 2005), the proceedings of the European Parliament, where English is aligned with French and German; and the UN parallel corpora (Ziemski et al., 2016), in which English is aligned with Arabic, French, German, Russian and Spanish. We used subsets of these corpora in which the direction of translation has been accurately annotated (Kurokawa et al., 2009; Rabinovich et al., 2016b; Tolochinsky et al., 2018). We cleaned the data by removing editor's comments and sentences with fewer than 5 tokens. We then down-sampled the corpora and extracted equally-sized subsets with 50,000 sentence-pairs in each language pair, distributed evenly across translation direction. These are the data we used in all the experiments described below.[2] Details on the available data are presented in Table 2.

Prepprossessing We preprocessed the data as follows. First, all words in the two languages were tagged for part of speech using FARASA (Abdelali et al., 2016) for Arabic and TreeTagger (Schmid, 1995) for the other languages. Second, all the sentence pairs were word aligned using FastAlign (Dyer et al., 2013). With the word alignments we were able to extract the features that will be explained in the next section.

Classification For the task of identifying the translation direction, we implemented various feature sets and used them for training a Logistic Regression classifier (with the implementation of Pedregosa et al. (2011)), mainly because it is faster yet no less accurate than SVM. We performed tenfold cross-validation for evaluation and report accuracy in %. As our datasets are balanced, the trivial baseline is 50%.

Neural network In addition to the classifiers described below, we also approached the task of determining translation direction with a neural network. Our main goal here was to guarantee best performance, even the cost of interpretability. We used a network consisting of one bi-directional Long Short-Term Memory (BiLSTM) layer with 100 units, followed by a fully connected layer with a single output; the loss is defined as binary cross-entropy (the network was implemented with Keras.) The input of the network is the two sentences, where the words are mapped to pre-trained GloVe word embedding vectors of 50 dimensions (we used Pennington et al. (2014) for English and Bojanowski et al. (2017) for the other languages.)

5 Features

We defined several novel features motivated by various insights from Translation Studies. We motivate and explain these feature in this section.

Baseline As a baseline, we implemented some of the features that were suggested by Volansky et al. (2015), including:

POS trigrams We used the frequencies of the 2000 most frequent POS trigrams for each language.

Function words Function words for many languages are available online. We used the frequencies of all the function words in each language (between 160 in Arabic and 600 in German).

Positional token frequency In different languages, the choice of words with which sentences begin is rather different, and is more constrained and formulaic than elsewhere in the sentence (Volansky et al., 2015). A clear example is greetings: parliament speakers may choose to begin their speeches

[2] The only other parallel corpora that we are aware of where the direction of translation is marked are the Dutch Parallel Corpus (Macken et al., 2011), aligning Dutch with English abd French, and EuroParl-UdS (Karakanta et al., 2018), which largely overlaps with our dataset.

	Europarl		UN				Hansard
	EN-FR	EN-DE	EN-FR	EN-ES	EN-RU	EN-AR	EN-FR
EN original	217	225	8100	6100	3600	4087	3377
EN original, cleaned	215	222	6600	5100	2800	3338	2981
EN translated	130	155	773	447	107	88	744
EN translated, cleaned	128	153	683	381	91	65	678

Table 2: Dataset sizes (in thousands of sentence-pairs)

by *'Ladies and gentlemen'*, but this turns out to be much more common in French than in English. We used the frequencies of words that occur in the first, second, penultimate and last positions in the sentences.

MTUs Finally, to compare with the state of the art, we also computed POS-MTUs and Brown Cluster MTUs, as defined by Eetemadi and Toutanova (2014, 2015).

Word rank The *simplification hypothesis* conjectures that translated texts tend to be simpler than originals. As one realization of this hypothesis, we assume that translations would use more common, frequent words than originals. In order to determine how common each word is, we used pre-trained frequency lists in all languages (Michel et al., 2010).

Comparing the actual (frequency-based) ranks of word forms across languages is rather problematic, especially when the morphologies of the languages differ. (e.g., when one language has many more inflected forms per lexeme than the other). Therefore, we split the word frequency lists to seven *bins* that group together words by their frequency, and compared the bins rather than the actual ranks.[3] The first bin includes words whose accumulated frequency is up to 0.25; it includes the most frequent words in each language. The other bins include words with accumulated frequency up to 0.5, 0.7, 0.8, 0.88, 0.95 and all the rest. This facilitates comparison of words in the same frequency brackets across two different languages. This feature defines 14 bins (7 for each language); its actual value is number of words in each bin.

Additionally, we compared the (frequency-based) ranks of aligned word pairs. Given a pair of aligned sentences, consider the difference in rank between each pair of aligned words. We hypothesize that such differences would depend on the translation direction (as rarer words tend to be translated to more common ones). For example, we expect the English *'however'* (ranked 236th) to be typically translated to French *'mais'* (ranked 33rd), but French *'mais'* to be more often translated to English *'but'* (ranked 23rd).

To implement this observation, we defined a histogram representing the values of the differences in rank between pairs of aligned words in each sentence pair. For example, if the English word *'however'* is ranked 236th and its aligned French word *'mais'* is ranked 33rd, we used the value $236 - 33 = 203$. We computed these values for all the aligned words in a sentence-pair; we then used the highest and lowest values as the boundaries of a histogram and split it to 12 bins. For example, if the defined limits of the histogram are: [-100000, -50000, -25000, -8000, -4000, -300, 300, 4000, 8000, 25000, 50000, 100000] and the resulting value from the differences between the words in a sentence pair are -10953, -511, 402, -3159, 4099, 11267, 10535, 80, 4280, 345; then the resulting histogram is: [0, 0, 0, 1, 0, 2, 1, 2, 2, 2, 0 ,0]. The values of this feature for a given pair of sentences are the values of each bin in the resulting histogram.

Lexically-Anchored-POS-MTUs While POS-MTUs identify meaningful linguistic structures, they are too general and may lose important nuances of the correspondences between constructions in the two languages. For example, consider the POS-MTUs [IN]↔[ART] in Figure 3: clearly it is not the case that prepositions in English translate to determiners in German. However, it is reasonable to assume that the English genitive preposition *'of'* will be aligned to a German genitive article such as *'der'*.

To reflect this notion, and define finer, subtler cross-language correspondences, we propose *Lexically-Anchored-POS-MTUs* (LA-POS-MTUs): we only replace *content* words by their POS tag, leaving *function* words intact. The values

[3] The number of bins and their frequency ranges were determined empirically.

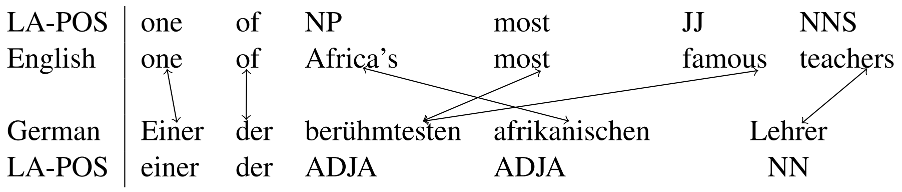


Figure 4: LA-POS-MTUs

of these features are the actual counts of each LA-POS-MTU in the sentences. Similarly to POS-MTUs, they are distributed differently in each of the translation directions.

As an example, consider the LA-POS-MTUs in Figure 4: [one]↔[einer], [of]↔[der], [NP]↔[ADJA], [most, JJ]↔[ADJA], [NNS]↔[NN]. In particular, the LA-POS-MTU [most, JJ]↔[ADJA] reflects the fact that in English, some superlative adjectives can come with the adverb *'most'* or with *'est'* as a suffix, while in German there is only one form: adding a suffix to the adjective. Indeed, the LA-POS-MTU [most, JJ]↔[ADJA] is much more frequent in English to German than in the reverse direction. This is presumably an instance of *interference* of German on the English translation product. While in English there are two ways to form the superlative, and sometimes both are valid (e.g., *'most clever'* and *'cleverest'*), German has only one possible form. When a superlative adjective is translated from German to English, the translator may tend to keep it with the suffix (if possible), rather than splitting it into two words. Hence, this LA-POS-MTU is more frequent in the English to German direction.

Syntactic structure The simplification hypothesis implies that the structure of translated sentences tends to be simpler than that of originals. We therefore parsed the corpus with universal dependencies (Straka and Straková, 2017) and defined several measures that supposedly reflect sentence complexity: the height of the dependency tree; its depth; and the average number of dependents per word. In addition, we used dependency tag trigrams as features, similarly to POS-trigrams.

Back translation Translated texts carry a unique signal; the challenge is to identify this signal at the sentence-pair level, where it may be extremely subtle. The motivation for the back translation feature is to amplify this signal.

To do so, we use machine translation (Google Translate) to translate the sentences again. Given a sentence pair $\langle e_1, f_1 \rangle$, we machine-translate both sentences, yielding the pair $\langle f_2, e_2 \rangle$, where $f_2 = MT(e_1)$ and $e_2 = MT(f_1)$, MT indicating machine translation. Now assume, without loss of generality, that e_1 is the original; hence f_1 is its manual translation, namely $f_1 = HT(e_1)$, where HT indicates human translation. Therefore, $e_2 = MT(f_1) = MT(HT(e_1))$. In other words, e_2 is "twice removed" from e_1, being translated once by a human and once automatically. In contrast, $f_1 = HT(e_1)$ and $f_2 = MT(e_1)$; both f_1 and f_2 are only "once removed" from e_1: f_1 was translated manually and f_2 automatically, but only once. Therefore, we expect f_1 and f_2 (two French sentences) to be closer to each other than e_1 and e_2 (two English sentences) are. This is only the case if f_1 is the translation of e_1; if the translation direction is reversed, we would expect e_1 and e_2 to be closer to each other than f_1 and f_2 are.

To measure the similarity between the two sentences we used three metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and Levenshtein distance (Levenshtein, 1965). Each metric results in two scores: one for the distance between the two English sentences and one for the two French sentences. These six scores were used as features for the classifier.

6 Results

Table 3 depicts the accuracy of 10-fold cross validation evaluation of classifiers reflecting the various features. The "All" column indicates a dataset constructed from the French–English sentence pairs in all the three different corpora; it is therefore a heterogenous dataset, which makes the task much more challenging (Rabinovich and Wintner, 2015). Indeed, the results on this dataset are worst, lower than each individual dataset in isolation. Still, even for this challenging experimental scenario, our best classifier achieves over 72% accuracy. For the Europarl and UN

Corpus	**Europarl**		**UN corpus**				**Hansard**	**All**
Feature set	EN-FR	EN-DE	EN-FR	EN-ES	EN-RU	EN-AR	EN-FR	EN-FR
POS-MTUs	64.4	63.1	63.4	62.6	69.2	76.2	62.7	58.1
LA-POS-MTUs	65.6	66.2	63.4	64.0	68.4	75.2	64.8	59.9
Brwn Clstr MTUs	73.0	67.1	66.4	68.3	71.9	79.0	64.8	60.3
Rank	63.5	64.8	58.0	59.0	60.8	65.2	56.6	56.0
POS-trigrams	65.0	65.7	64.0	63.2	67.0	74.3	64.1	59.6
Function words	65.6	68.0	66.3	66.1	72.3	69.0	66.5	56.6
Pos. token freq.	62.0	64.7	65.9	66.7	76.0	80.8	64.2	61.0
Syntactic structure	64.0	62.0	65.0	63.3	68.6	67.0	61.4	58.8
Back translation	61.2	58.5						
All	**81.0**	**78.1**	**75.6**	**78.0**	**84.5**	**90.1**	**75.1**	**67.9**
BiLSTM	81.0	80.9	79.8	84.8	90.8	89.0	**78.4**	**74.6**
Stacking	**83.0**	**82.3**	**80.3**	**84.9**	**91.1**	**90.0**	76.5	72.1

Table 3: Results: accuracy (%) of predicting the translation direction

datasets, however, our results range between 80% and over 90% accuracy; given the difficulty of the task (refer back to Figure 1), we view this as a significant contribution.

The "All" row indicates the concatenation of all features into one feature vector. Since these features encode different aspects of the relations between the two languages, we believe that they are at least partially independent. Indeed, the results of feature combination support this assumption.

The signal of translationese is indeed subtle, but the results show that many of our basic classifiers are able to detect it, albeit to a small extent. For most language pairs and datasets, each of the feature sets we defined yielded accuracy of over 60%, sometimes over 70%, and reaching 80% in a few cases. Brown cluster MTUs, which were used by the state of the art (Eetemadi and Toutanova, 2015), are indeed a good feature set. MTUs based on Brown clusters turned out to be better than LA-POS-MTUs; presumably, Brown clusters encode lexical semantic information that is helpful for the task. However, they are outdone in more than half of the cases by simpler features such as function words or positional token frequencies.

Back translation turned out to be a less beneficial feature than we have expected on Europarl. As it is a computation-intensive feature, we refrained from computing it on the other datasets.

Combining features together yielded a sizable boost in accuracy, advancing the state of the art to the area of 80-90% accuracy in all cases. The features that we defined are obviously not mutually independent; it therefore makes sense to try some dimensionality reduction method to remove redundant features. We tried several dimensionality reduction methods, with various dimensionalities, but none yielded better results (on the full datasets) than using all features.

As could be expected, the accuracy of the BiLSTM is higher than feature combination in all cases but one; yet we suspect that the features capture phenomena that are not reflected by the neural network. To test that, we used *stacking*. We defined three different classifiers: one with features computed from the English texts only (rank, POS trigrams, function words, positional tokens, and syntactic structure); another with the same features computed from the other language; and a third from the alignment features computed from both languages (the three MTU feature types). We additionally trained the neural network. We then used all four classifiers to predict the direction of translation, and used their confidence scores as features for a stacked classifier, whose prediction is the class we use. The results are listed in Table 3 under "Stacking", and show a small but consistent improvement for all language pairs.

Still, the BiLSTM turned out to be better for the Hansard corpus and for the mixed dataset. We do not have a clear explanation for this outcome. We used paired t-test to determine the statistical significance of the improvement in results between using all the features ("All") and the best results obtained by Stacking. The test yielded p-values <0.001 for all language pairs except English–Arabic. Similarly, comparing the neural network with Stacking in the same way, the test yielded p-values <0.001 in all language pairs except English–Spanish. We thus conclude that the generalizations of the neural network are, at least to some extent, different from the features we defined. In future work, we intend to consider new ways for incorporating linguistically-motivated features in neural network architecture, e.g., along the lines of Strubell et al. (2018).

Finally, observe that the results clearly support our theoretical hypothesis: the accuracy of the classification improves when the two languages involved are more typologically distant. The task is particularly hard for English-French and English-German, and easiest for English-Arabic and English-Russian. We tentatively conclude, therefore, that translationese is more pronounced, and interference is more powerful, when the two languages are more distant. This chimes in with recent results that show the relationships between interference and language typology (Rabinovich et al., 2017).

7 Conclusion

We have shown that linguistically-motivated features, based on Translation Studies insights pertaining to the asymmetry of the translation process, can yield high, state-of-the-art accuracy on the task of translation direction detection. We introduced several novel features and used stacking to produce highly accurate sentence-pair-level classifiers for five language pairs. We also confirmed the hypothesis that this task is harder when the two languages involved are more closely related.

In future work, we intend to provide a deeper analysis of the results, focusing on the constructions whose frequencies differ most across the two languages. We would also like to evaluate our systems cross-domain, as it has been shown (Rabinovich and Wintner, 2015) that the signal of translationese is subtle, and can be overshadowed by signals of the datasets used for training and testing. Finally, and depending on the availability of datasets, we would like to extend the experiments described herein to more language pairs.

Acknowledgements

This research was supported by Grant No. 2017699 from the United States-Israel Binational Science Foundation (BSF), by Grant No. 1813153 from the United States National Science Foundation (NSF), and by grant No. LU 856/13-1 from the Deutsche Forschungsgemeinschaft. We are grateful to Noam Ordan and Ella Rabinovich for helpful discussions and creative ideas. We thank the anonymous RANLP reviewers for their constructive comments. All remaining errors and misconceptions are, of course, our own.

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California. Association for Computational Linguistics.

Ehud Alexander Avner, Noam Ordan, and Shuly Wintner. 2016. Identifying translationese at the word and sub-word level. *Digital Scholarship in the Humanities*, 31(1):30–54.

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. In Mona Baker, Gill Francis, and Elena Tognini-Bonelli, editors, *Text and technology: in honour of John Sinclair*, pages 233–252. John Benjamins, Amsterdam.

Mona Baker. 1995. Corpora in translation studies: An overview and some suggestions for future research. *Target*, 7(2):223–243.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing*, 21(3):259–274.

Shoshana Blum-Kulka. 1986. Shifts of cohesion and coherence in translation. In Juliane House and Shoshana Blum-Kulka, editors, *Interlingual and intercultural communication Discourse and cognition in translation and second language acquisition studies*, volume 35, pages 17–35. Gunter Narr Verlag.

Shoshana Blum-Kulka and Eddie A. Levenston. 1983. Universals of lexical simplification. In Claus Faerch and Gabriele Kasper, editors, *Strategies in Interlanguage Communication*, pages 119–139. Longman.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Sauleh Eetemadi and Kristina Toutanova. 2014. Asymmetric features of human generated translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 159–164. Association for Computational Linguistics.

Sauleh Eetemadi and Kristina Toutanova. 2015. Detecting translation direction: A cross-domain study. In *NAACL Student Research Workshop*. ACL – Association for Computational Linguistics.

Martin Gellerstam. 1986. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, pages 88–95. CWK Gleerup, Lund.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 937–944.

Iustina Ilisei, Diana Inkpen, Gloria Corpas Pastor, and Ruslan Mitkov. 2010. Identification of translationese: A machine learning approach. In *Proceedings of CICLing-2010: 11th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 6008 of *Lecture Notes in Computer Science*, pages 503–511. Springer.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Alina Karakanta, Mihaela Vela, and Elke Teich. 2018. EuroParl-UdS: Preserving and extending metadata in parliamentary debates. In *Proceedings of ParlaCLARIN*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the tenth Machine Translation Summit*, pages 79–86. AAMT.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1326, Portland, Oregon, USA. Association for Computational Linguistics.

David Kurokawa, Cyril Goutte, and Pierre Isabelle. 2009. Automatic detection of translated text and its impact on machine translation. In *Proceedings of MT-Summit XII*, pages 81–88.

Sara Laviosa. 1998. Core patterns of lexical use in a comparable corpus of English lexical prose. *Meta*, 43(4):557–570.

Sara Laviosa. 2002. *Corpus-based translation studies: theory, findings, applications*. Approaches to translation studies. Rodopi.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2012. Language models for machine translation: Original vs. translated texts. *Computational Linguistics*, 38(4):799–825.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2013. Improving statistical machine translation by adapting translation models to translationese. *Computational Linguistics*, 39(4):999–1023.

Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848.

Lieve Macken, Orphée De Clercq, and Hans Paulussen. 2011. Dutch parallel corpus: A balanced copyright-cleared parallel corpus. *Meta*, 56(2):374–390.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Holberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2010. Quantitative analysis of culture using millions of digitized books. *Science*.

Sergiu Nisioi. 2015. Unsupervised classification of translated texts. In *Natural Language Processing and Information Systems: Proceedings of the 20th International Conference on Applications of Natural Language to Information Systems, NLDB*, volume 9103 of *Lecture Notes in Computer Science*, pages 323–334. Springer.

Lin Øverås. 1998. In search of the third code: An investigation of norms in literary translation. *Meta*, 43(4):557–570.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Ella Rabinovich, Sergiu Nisioi, Noam Ordan, and Shuly Wintner. 2016a. On the similarities between native, non-native and translated texts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 1870–1881.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540. Association for Computational Linguistics.

Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics*, 3:419–432.

Ella Rabinovich, Shuly Wintner, and Ofek Luis Lewinsohn. 2016b. A parallel corpus of translationese. In *Proccedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038. Association for Computational Linguistics.

Elke Teich. 2003. *Cross-Linguistic Variation in System and Text: A Methodology for the Investigation of Translations and Comparable Texts*. Mouton de Gruyter.

Elad Tolochinsky, Ohad Mosafi, Ella Rabinovich, and Shuly Wintner. 2018. The UN parallel corpus annotated for translation direction. ArXiv:1805.07697 [cs.CL].

Gideon Toury. 1979. Interlanguage and its manifestations in translation. *Meta*, 24(2):223–231.

Gideon Toury. 1980. *In Search of a Theory of Translation*. The Porter Institute for Poetics and Semiotics, Tel Aviv University, Tel Aviv.

Gideon Toury. 1995. *Descriptive Translation Studies and beyond*. John Benjamins, Amsterdam / Philadelphia.

Naama Twitto-Shmuel, Noam Ordan, and Shuly Wintner. 2015. Statistical machine translation with automatic identification of translationese. In *Proceedings of WMT-2015*.

Ria Vanderauwerea. 1985. *Dutch novels translated into English: the transformation of a 'minority' literature*. Rodopi, Amsterdam.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities*, 30(1):98–118.

Michałl Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Automated Text Simplification as a Preprocessing Step for Machine Translation into an Under-resourced Language

Sanja Štajner
Symanto Research
Nuremberg, Germany
sanja.stajner@symanto.net

Maja Popović
ADAPT Centre, DCU
Dublin, Ireland
maja.popovic@adaptcentre.ie

Abstract

In this work, we investigate the possibility of using fully automatic text simplification system on the English source in machine translation (MT) for improving its translation into an under-resourced language. We use the state-of-the-art automatic text simplification (ATS) system for lexically and syntactically simplifying source sentences, which are then translated with two state-of-the-art English-to-Serbian MT systems, the phrase-based MT (PBMT) and the neural MT (NMT). We explore three different scenarios for using the ATS in MT: (1) using the raw output of the ATS; (2) automatically filtering out the sentences with low grammaticality and meaning preservation scores; and (3) performing a minimal manual correction of the ATS output. Our results show improvement in fluency of the translation regardless of the chosen scenario, and difference in success of the three scenarios depending on the MT approach used (PBMT or NMT) with regards to improving translation fluency and post-editing effort.

1 Introduction

In spite of recent advances in machine translation (MT), the MT into under-resourced languages is still facing a number of problems. First, there is not enough parallel data to build robust phrase-based and neural systems. Second, the majority of those languages (including Serbian) have a very rich morphology and suffer from data sparsity when it comes to less frequently used cases, tenses, etc. Third, there is a number of syntactic differences which are difficult to capture. For English-to-Serbian phrase-based system, a number of language-related problems has been identified so far (Popović and Arčan, 2015). Most of them are related to syntactic differences, e.g. missing verb parts due to distinct structure of certain verb tenses, incorrect prepositions, or incorrect translations of English sequences of nouns. Although the neural approach better handles some grammatical aspects, it still often fails to generate correct inflections, prepositions and translations of the English noun phrases (Popović, 2017).

Text simplification (TS) has the goal of transforming given text or sentence into its simpler variant, while preserving the original meaning. What is considered to be a simpler variant depends on the target application, or the target reader in mind. In the case of simplifying texts for humans, a simpler variant is the one that requires a shorter reading time and leads to better text comprehension scores. In the case of text or sentence simplification used as a preprocessing step for a given natural language processing (NLP) task, e.g. machine translation (MT), information extraction (IE), summarization, and semantic role labeling (SRL), a simpler variant is the one that leads to better performances of that NLP system.

Text simplification was originally proposed as a pre-processing step for machine translation (Chandrasekar, 1994) and later for information extraction and parsing (Chandrasekar et al., 1996). At those early stages, automated text simplification (ATS) was not mature enough to help improving performances of those systems. Instead, the idea was explored only hypothetically, using manual text simplification (Chandrasekar, 1994; Vickrey and Koller, 2008). Evans (2011) later showed that an automated simplification of coordinate structures can improve IE systems.

Later, the focus of the ATS shifted towards text accessibility and better social inclusion, having the main goal of making texts easier to understand by

Proceedings of Recent Advances in Natural Language Processing, pages 1141–1150,
Varna, Bulgaria, Sep 2–4 2019.

various target readers, e.g. people with low literacy levels (Aluísio and Gasperin, 2010), or people with some kind of reading or cognitive impairments, such as aphasia (Devlin and Unthank, 2006), autism (Orăsan et al., 2018), Down's syndrome (Saggion et al., 2015), or dyslexia (Rello, 2012).

In this study, we want to return to the original motivation for text simplification and explore whether the state-of-the-art 'general purpose' ATS system can be used to improve machine translation from English to some under-resourced language, or not. Unlike the previous works, we focus on using fully automated TS output (without any manual corrections), and on filtering out simplifications that are not grammatical and do not preserve the original meaning. Furthermore, we perform experiments using two state-of-the-art MT systems with different architectures, a PBMT and a NMT system. We focus on English-to-Serbian machine translation, taking Serbian as an example of under-resourced languages.

We show that, regardless of the MT architecture (PBMT or NMT) and the strategy for using ATS system as a pre-processing step (without any manual correction, with filtering for its grammaticality and meaning preservation, and with minimal manual correction of the output), the fluency of the translation can be improved. With regards to improving translation adequacy and post-editting effort, our experiments show that the type of the translation architecture (PBMT or NMT), and the strategy for using ATS system, both play a significant role.

2 Related Work

For many language pairs (e.g. English-French, English-Spanish, English-Hindu), attempts were made at rewriting input sentences using paraphrasing or textual entailment to improve the performance of MT systems (Callison-Burch et al., 2006; Mirkin et al., 2009; Aziz et al., 2010; Tyagi et al., 2015; Mirkin et al., 2013a,b). However, they all focus only on out-of-vocabulary words, or difficult to translate shorter n-grams.

Štajner and Popović (2016) went one step further, using lexico-syntactic automatic text simplification systems as a pre-processing step for English-to-Serbian machine translation. In this way, they covered both lexical and syntactic transformations on the source side. The ATS outputs were manually inspected by human editors who were also allowed to do minor revisions (correcting the tense, gender, article, etc.) in order to preserve grammaticality and the original meaning on the source side. For both ATS systems used, it was found that (with this minimal human correction of the simplified output) such a pre-processing step improves fluency of the translations, and reduce the post-editing effort. However, the authors only considered manually post-edited ATS output and made no experiments with the raw (uncorrected) ATS output. Nor did they explore how the grammaticality and meaning preservation of the ATS output might influence the results.

In this work, we use one of the ATS systems used by Štajner and Popović (2016), only the system which does not remove any original information. Unlike Štajner and Popović (2016), who used only a phrase-based MT system for English-to-Serbian translation, we also use the current state-of-the-art neural MT system for that language pair (see Section 3.1). We explore three different scenarios for using ATS as the pre-processing step, in search for fully automatic use of ATS in MT, without human correction of the ATS output (see Section 3). We find that success of the ATS used as a pre-processing step heavily depends on the type of the MT system used (PBMT or NMT).

3 Experimental Setup

We randomly selected 10 original articles from the 100 news articles automatically simplified by the state-of-the-art lexico-syntactic ATS system (Siddharthan and Angrosh, 2014) in the work of Štajner and Glavaš (2017). The ATS system that consists of a rule-based syntactic simplification module and a supervised lexical simplification module built upon the English Wikipedia - Simple English Wikipedia corpus (Coster and Kauchak, 2011).

We further explored three possible scenarios in which ATS can be used as a pre-processing step for MT (Figure 1):

- **Scenario 1 (*Corrected*)**: Automatically simplified sentences are manually corrected before being used as the source sentences for MT, to ensure the preservation of the original meaning and the grammaticality of the MT input;

- **Scenario 2 (*Filtered*)**: Automatically simpli-

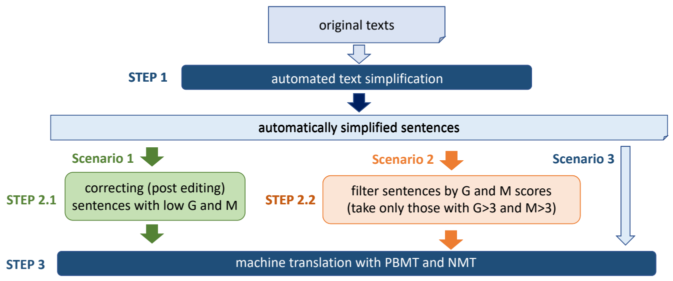

original texts
STEP 1
automated text simplification
automatically simplified sentences
Scenario 1
Scenario 2
Scenario 3
STEP 2.1
correcting (post editing) sentences with low G and M
STEP 2.2
filter sentences by G and M scores (take only those with G>3 and M>3)
STEP 3
machine translation with PBMT and NMT

Figure 1: Workflow. The fully-automated steps are shown in dark blue (steps 1 and 3); fully-manual in green (step 2.1); and those that can be used as either automated or manual, in orange (step 2.2).

fied sentences which did not preserve well the original meaning and/or are ungrammatical are filtered out, and in those cases, the original sentences are used instead of them as the MT input;

- **Scenario 3 (*Automatic*)**: Automatically simplified sentences are used as source sentences for MT without any manual correction or filtering beforehand.

The last scenario (*Automatic*) is troublesome in the context of ATS used as pre-processing step for MT, as one cannot be sure that the original meaning was preserved during automatic simplification. To account for possible changes of meaning, we slightly modify the common procedures for assessing fluency and adequacy of MT output in order to allow those scores to penalize such text simplification errors (see Section 3.3).

3.1 MT Systems

For English-to-Serbian phrase-based and neural machine translation, we use ASISTENT[1] (Arčan et al., 2016), a publicly available web-based MT system offering translation between three South Slavic languages (Croatian, Serbian and Slovenian) and English in both translation directions. Both NMT and PBMT variants were trained on the publicly available data originating from the OPUS website[2] (Tiedemann, 2009) where three domains were available for the Serbian-English language pair: the enhanced version of the SEtimes corpus[3] (Tyers and Alperen, 2010) containing "news and views from South-East Europe", OpenSubtitles[4] as well as KDE localisation documents and manuals, i.e. technical domain. In total, about 20.7M sentences were used for training (20.5M subtitles, 200k news, 30k technical), and 2k sentences from each of the three domains were used for tuning the systems.

The texts which we are translating in this study are external, i.e. they cannot be found in any of the above described corpora.

3.2 Assessment of Quality of ATS Output

The quality of the ATS output was assessed at the sentence level by:

- human assessment of grammaticality (G) and meaning preservation (M) on a 1–5 Likert scale (1 – very bad; 5 – very good).
- measuring the time needed to correct grammaticality and ensure that original meaning is preserved.

The first assessment was used in Scenario 2 (*Filtered*) for filtering simplified sentences according to their G and M scores, while the second assessment was used in Scenario 1 (*Corrected*). The schema of the workflow is presented in Figure 1.

We asked three native English speakers to rate our 130 sentences using the same guidelines as Štajner and Glavaš (2017). The annotators were also provided with several examples for each score. To obtain the final G and M marks used in Scenario 2 (*Filtered*), we averaged the three marks and rounded the results to the closest integers. The average G and M scores were 3.98 and 3.75, respectively. The average pairwise inter-annotator

[1] http://server1.nlp.insight-centre.org/asistent/
[2] http://opus.lingfil.uu.se
[3] http://nlp.ffzg.hr/resources/corpora/setimes/
[4] http://www.opensubtitles.org/

Version	Sentence
Original	Ex-Soviet leader Mikhail Gorbachev says Russian authorities must annul the parliamentary vote results and hold a new election.
Simplification (uncorrected)	Ex-Soviet leader Mikhail Gorbachev says**.** Russian authorities must annul the parliamentary vote results. These authorities hold a new election.
Simplification (corrected)	Ex-Soviet leader Mikhail Gorbachev says **that** Russian authorities must annul the parliamentary vote results. These authorities **must** hold a new election.
Original	A 21-year-old man was arrested on April 30, on suspicion of murder and was released on bail until May 29 pending further enquiries.
Simplification (uncorrected)	A 21-year-old man was arrested on April 30, on suspicion of murder. This man was **followed** until May 29 pending further enquiries.
Simplification (corrected)	A 21-year-old man was arrested on April 30, on suspicion of murder. This man was **released** until May 29 pending further enquiries.

Table 1: Two examples of manual corrections performed on the simplified sentences. Differences between the automatically simplified sentences and their manually corrected versions are shown in bold.

	Adequacy
5	perfectly understandable (regardless of potential poor grammar)
4	understandable with minor ambiguities/differences
3	main gist is preserved but some things are unclear/different from the source
2	difficult to understand and different from the source meaning
1	very bad (regardless to potential grammaticality)
	Fluency
5	perfectly grammatical (regardless of potential meaning loss/change)
4	almost correct – a small number of minor errors
3	a number of grammatical errors although not very heavy
2	many grammatical errors
1	very bad (regardless to potential meaning preservation)

Table 2: Guidelines for assigning adequacy and fluency scores.

agreement was 0.72 and 0.61 (weighted Cohen's κ), for the G and M scores respectively.

In Scenario 1 (*Corrected*), we used the manual corrections already provided by Štajner and Glavaš (2017) for the ten selected articles. During manual corrections, only minimal corrections were performed where necessary to restore the original meaning and grammaticality of the sentences. As the goal of those correction was not to make any further simplifications, and the mistakes were easy to notice, the corrections were very fast (15.0 seconds per sentence). They did not even require a native speaker or trained annotator, but rather someone with a proficiency level of English (Štajner and Glavaš, 2017). Several examples of performed corrections are given in Table 1.

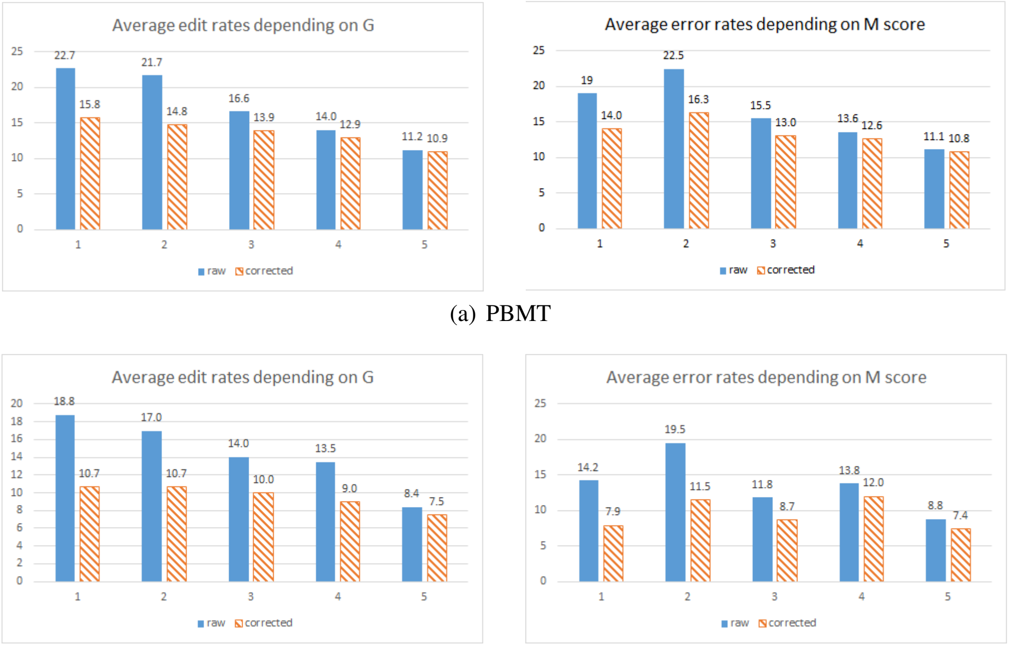
3.3 Evaluation of MT Output

All translation outputs (translations of the original sentences, their raw (uncorrected) automatic simplifications, and their manually corrected automatic simplifications) obtained by the two MT systems (PBMT and NMT), a total of 390 target sentences, were evaluated with respect to three aspects:

- adequacy, i.e. how well the sentence preserves the original meaning;
- fluency, i.e. how grammatical the sentence is;
- technical post-editing effort, i.e. the amount of necessary edit operations to correct the output.

These three evaluation aspects concentrate on three distinct things, which are not necessarily correlated. For example, if the reference translation is *"We will* **not** *finish on time."* and the obtained MT output is *"We will finish on time."*, the adequacy score will be very low (1), fluency perfect (5), and edit distance very low (only one edit).
Each of the tasks has been carried out separately, i.e. the evaluation of adequacy and fluency were carried out in two separate passes, and post-editing was carried out in the third pass.

The guidelines used for assigning adequacy and fluency scores are presented in Table 2.



(a) PBMT

(b) NMT

Figure 2: Average edit rates (y-axis) for corrected TS output (blue full) and uncorrected/raw TS output (orange pattern), depending on the grammaticality (G) and meaning preservation (M) scores (x-axis) after PBMT and NMT.

Raw edit counts and edit rates (raw counts normalised with the segment length) were calculated using Hjerson (Popović, 2011) for:

- the five classes of edits/errors: inflectional error, reordering error, omission, addition and mistranslation
- the sum of edit counts/rates of all classes

Each translated segment was post-edited, and assigned fluency and adequacy scores, while looking into the corresponding source segment, i.e. the English originals were used for evaluating the translations of the originals, while the corresponding simplified and corrected English sentences were used for evaluating the translations of the simplified sentences. This was done to ensure that the change of meaning during ATS (in scenarios with filtered and fully automatic ATS) is penalised. Reference translations were not available.

4 Results

We first explored the influence of grammaticality (G) and meaning preservation (M) scores of the automatically simplified sentences on the post-editing effort needed to correct their translations (Figure 2).[5]

We see that the differences in effort needed to post-edit the translations of automatically simplified sentences (uncorrected) and corrected simplified sentences (corrected) decrease with the increase of grammaticality (G) and meaning preservation (M) scores of the uncorrected simplifications. This supports our initial idea that, instead of correcting the automatically simplified sentences (Scenario 1 in Figure 1), one could filter out the automatically simplified sentences which did not achieve high enough G and M scores, and instead of those, use the original sentences (Scenario 2 in Figure 1). This filtering should ideally be done automatically. Given that here we just look for the proof of concept, we wanted to ensure that we correctly assign G and M scores, and this was thus done manually. However, it is important to note that several systems for automatic assignment of G and M scores to the ATS outputs have been proposed up to date (Štajner et al., 2016).

The initial exploration (Figure 2) indicated that a good cut-off point for filtering bad simplifications would be around G = 3 and M = 3.

[5] For the space constraints, we here present graphs only for the average edit rates, but we also analysed the raw counts and found that they follow the same trends as the avearage edit rates.

(a) PBMT

%	uncorrected		**corrected**		**G, M > 3**		G, M ≥ 3	
	Σer	r.e.c.	Σer	**r.e.c.**	Σer	r.e.c.	Σer	r.e.c.
better	27.7	21.5	**46.2**	**38.5**	**20.0**	**16.9**	24.6	21.5
worse	50.8	49.2	**27.6**	**26.2**	**13.8**	**12.3**	27.7	53.9
same	21.5	29.3	26.2	35.3	66.2	70.8	47.7	24.6

(b) NMT

%	uncorrected		**corrected**		G, M > 3		G, M ≥ 3	
	Σer	r.e.c.	Σer	r.e.c.	Σer	r.e.c.	Σer	r.e.c.
better	13.8	12.3	**46.2**	**36.9**	9.2	6.2	13.8	10.8
worse	67.7	64.6	**37.5**	**32.3**	27.7	26.2	41.6	40.0
same	18.5	23.1	32.3	30.8	63.1	67.6	44.6	49.2

Table 3: Percentage of raw simplified sentences (uncorrected), corrected simplified sentences (corrected) and simplified sentences filtered by high grammatical and meaning scores (G, M > 3, and G, M ≥3) with better/worse/same translations (in terms of edit rate (Σer) and raw edit counts (r.e.c.)) than their original counterparts when translated with phrase-based (a) and neural (b) MT system. Those cases in which more sentences improved than deteriorated are shown in bold.

We further investigated how the distribution of sentences with better, worse, and same post-editing effort differs in the two possible cut-offs: (1) where G and M scores are both greater than 3; and (2) where G and M scores are both greater or equal to 3. The results are presented in Table 3, together with the corresponding results for the fully automated simplification without any filtering (uncorrected) and for the manually corrected simplification output (corrected).

It seems that filtering automatically simplified sentences according to their grammaticality (G) and meaning preservation (M) score can substantially decrease the percentage of sentences whose translation is worse (needs more post editing) than the translation of their original counterparts, if the cut-off point is correctly set. This happens in both MT approaches used (PBMT or NMT), but the decrease is more pronounced in PBMT.

Interestingly, in PBMT, filtering sentences according to their G and M scores (with G, M > 3) results in a lower percentage of sentences with deteriorated translations than correcting the simplification output before translation (corrected). This happens at the cost of increasing the number of sentences with the same MT post-editing effort required, and decreasing the number of sentences with improved translation.

Table 4 shows the differences in fluency and adequacy scores between the translations of the original sentences and the translations of the corrected simplified texts, uncorrected simplified texts, and the filtered (G, M > 3) source sentences.

It can be seen that by using the ATS system in a pre-processing step (marginally) improves the translation adequacy, and only if the simplifications are manually corrected and used in the NMT system.

Using any of the three proposed scenarios (manually corrected simplifications, filtered simplifications, or fully automated simplifications) leads to a higher percentage of sentences with improved rather than those with deteriorated fluency of translation. Interestingly, the percentage of improved sentences is the highest when the uncorrected simplifications are used, but at the cost of the higher percentage of sentences with deteriorated translation adequacy (than in the case of using the manually corrected simplifications). Filtering automatic simplifications according to their grammaticality and meaning preservation substantially decreases the percentage of sentences with both improved and deteriorated translation fluency, but still results in substantially higher percentage of sentences with improved than those with deteriorated fluency.

Several examples of the original English sentences and their simplified versions, together with the scores for the fluency and adequacy of their Serbian translations are presented in Table 5.

(a) PBMT

Difference		Adequacy			**Fluency**	
	corrected	uncorrected	G, M > 3	**corrected**	**uncorrected**	**G, M > 3**
-3	0	1.5	0	0	0	0
-2	3.1	16.9	9.2	0	0	0
-1	21.5	53.8	30.8	15.4	20.0	10.8
0	60.0	23.1	56.9	61.5	53.8	76.9
1	15.4	4.6	3.1	21.5	34.6	12.3
2	0	0	0	1.5	1.5	0
3	0	0	0	0	0	0
Σ positive	15.4	4.6	3.1	**23.0**	**36.1**	**12.3**
Σ negative	24.6	81.2	40.0	15.4	20.0	10.8

(b) NMT

Difference		Adequacy			**Fluency**	
	corrected	uncorrected	G, M > 3	**corrected**	**uncorrected**	**G, M > 3**
-3	1.5	3.1	0	0	0	0
-2	4.6	13.8	6.2	0	3.1	0
-1	20.0	33.8	13.8	13.8	18.5	7.7
0	47.7	38.5	72.3	63.1	52.3	78.5
1	23.1	9.2	6.2	21.5	23.1	12.3
2	3.1	1.5	1.5	0	1.5	0
3	0	0	0	1.5	1.5	1.5
Σ positive	**26.2**	10.7	7.7	**23.0**	**26.1**	**13.8**
Σ negative	26.1	50.7	20.0	13.8	21.6	7.7

Table 4: Distribution of differences in adequacy and fluency scores (in terms of percentages) introduced by the manually corrected simplifications, uncorrected simplifications, and by filtering automatic simplifications by their meaning preservation and grammaticality scores.

5 Summary and Outlook

Going back to the initial motivation for automatic text simplification, using it as a pre-processing step to improve the performance of machine translation systems, we explored how the current state-of-the-art lexico-syntactic automatic text simplification system behaves in this role. We investigated three possible scenarios: (1) using the output of the ATS system as it is; (2) filtering out the automatic simplifications with low grammaticality and meaning preservation scores and using the original sentences instead; (3) minimally correcting the ATS output in order to preserve the original meaning and the grammaticality of the sentence.

The results of our experiments indicated that:

- The success of the ATS systems depends on the type of MT system used (phrase-based or neural).
- In the case of NMT, only the manually corrected ATS output can reduce the post-editing effort, and improve the adequacy scores in translation.
- In the case of PBMT, two scenarios (manual correction of ATS output and maintaining only automatic simplifications with high G and M scores) can reduce the post-editing effort in translation, but none of the three investigated scenarios can improve the adequacy of the translation.
- For both MT approaches, PBMT and NMT, all three scenarios for using ATS improve the fluency of the translation noticeably.
- The uncorrected ATS output improves the fluency of the PBMT translation noticeably more than it improves the fluency of the NMT translation.

We also found that even manually corrected ATS output can deteriorate translation adequacy (in about 15% of the cases) and fluency (in about

(a) PBMT

Ex.	Version	r.e.c/Σer/A/F	Sentence
1	Original	21/75.0/2/2	The cabinet is also expected to demand banks to set aside a further 35 billion euros (27.9 billion pounds) to cover sound loans in their real estate portfolios.
	Uncorrected (G=4, M=5)	18/58.7/1/2	The cabinet is also expected to demand banks to set aside a further 35 billion euros, to cover sound loans in their real estate portfolios. **These** is 27.9 billion pounds.
	Corrected	16/50.0/2/2	The cabinet is also expected to demand banks to set aside a further 35 billion euros, to cover sound loans in their real estate portfolios. **35 billion** euros is 27.9 billion pounds.
2	Original	9/56.7/2/2	The toxic assets now total 184 billion euros, but many fear the hole is even bigger.
	Uncorrected (G=4, M=5)	8/50.8/3/3	The toxic assets now total 184 billion euros. But many fear the hole is even bigger.
	Corrected	7/43.6/4/4	The toxic assets now total 184 billion euros but many fear the hole is even bigger.

(b) NMT

Ex.	Version	r.e.c/e.r./F/A	Sentence
1	Original	5/28.3/5/4	Jeffrey Burrows was killed at his home in Norfolk Square in Brighton, East Sussex, on April 29.
	Uncorrected (G=5, M=3)	3/14.1/4/4	Jeffrey Burrows was killed at his home in Norfolk Square in Brighton, on April 29. Brighton is East Sussex.
	Corrected	2/9.1/5/4	Jeffrey Burrows was killed at his home in Norfolk Square in Brighton, on April 29. Brighton is **in** East Sussex.
2	Original	3/14.3/3/3	Spanish stocks were down 2.1 percent on Friday morning, in line with other European markets, after getting a big boost on Thursday from the banking reform plans.
	Uncorrected (G=5, M=5)	3/14.3/4/4	Spanish stocks were down 2.1 percent on Friday morning, in line with other European markets. This happened after getting a big boost on Thursday from the banking reform plans.

Table 5: Examples of original sentences, automatically simplified sentences (uncorrected), and their manually corrected versions, together with corresponding translation scores in terms of edit operations, adequacy and fluency scores.

25% of the cases), and increases the post-editing effort in translation (in about 26-27% of the cases in PBMT, and about 32-37% of the cases in NMT). This indicates that the general-purpose ATS systems (which were not initially developed for improving MT performances) might not be suitable for this task. Depending on the source-target language combination, it might be better to design a MT-oriented text simplification system, which would target only the sentence structures and vocabulary which poses particular difficulties in that language pair and MT approach.

Our results also showed that there are notable differences in how the three scenarios for using ATS as a pre-processing step influence the performances of the PBMT and NMT (at least for the language pair investigated here). This indicates that the two MT approaches require different pre-processing strategies in order to improve their performances.

6 Acknowledgements

The ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University is funded under the Science Foundation Ireland Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Sandra Maria Aluísio and Caroline Gasperin. 2010. Fostering digital inclusion and accessibility: The porsimples project for simplification of portuguese texts. In *Proceedings of YIWCALA Workshop at NAACL HLT 2010*. pages 46–53.

Mihael Arčan, Maja Popović, and Paul Buitelaar. 2016. Asistent – A Machine Translation System for Slovene, Serbi an and Croatian. In *Proceedings of the Conference on Language Technologies an d Digital Humanities*. Ljubljana, Slovenia, pages 13–20.

Wilker Aziz, Marc Dymetman, Shachar Mirkin, Lucia Specia, Nicola Cancedda, and Ido Dagan. 2010. Learning an expert from human annotations in statistical machine translation: the case of out-of-vocabulary words. In *Proceedings of EAMT*.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of the 2016 Human Language Technology Conference of the North American Chapter of the ACL (HLT-NAACL)*. pages 17–24.

R. Chandrasekar. 1994. *A Hybrid Approach to Machine Translation using Man Machine Communication*. Ph.D. thesis, Tata Institute of Fundamental Research/University of Bombay, Bombay.

Raman Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and Methods for Text Simplification. In *Proceedings of COLING 1996*. pages 1041–1044.

William Coster and David Kauchak. 2011. Simple English Wikipedia: a new text simplification task. In *Proceedings of ACL&HLT*. pages 665–669.

Siobhan Devlin and Gary Unthank. 2006. Helping aphasic people process online information. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*. ACM, New York, NY, USA, Assets '06, pages 225–226.

Richard J. Evans. 2011. Comparing methods for the syntactic simplification of sentences in information extraction. *Literary and Linguistic Computing* 26(4):371–388.

Shachar Mirkin, Lucia Specia, Nicola Cancedda, Ido Dagan, Marc Dymetman, and Idan Szpektor. 2009. Source-language entailment modeling for translating unknown terms. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '09, pages 791–799.

Shachar Mirkin, Sriram Venkatapathy, and Marc Dymetman. 2013a. Confidence-driven Rewriting for Improved Translation. In *Proceedings of the XIV MT Summit, Nice, France*. pages 257–264.

Shachar Mirkin, Sriram Venkatapathy, Marc Dymetman, and Ioan Calapodescu. 2013b. SORT: An Interactive Source-Rewriting Tool for Improved Translation. In *Proceedings of ACL, Sofia, Bulgaria*. pages 85–90.

Constantin Orăsan, Richard Evans, and Ruslan Mitkov. 2018. *Intelligent Natural Language Processing: Trends and Applications*, Springer, chapter Intelligent Text Processing to Help Readers with Autism, pages 287–312.

Maja Popović. 2011. Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output. *The Prague Bulletin of Mathematical Linguistics* 96:59–68.

Maja Popović. 2017. Comparing Language Related Issues for NMT and PBMT between German and English. *The Prague Bulletin of Mathematical Linguistics* 108(1):209–220.

Maja Popović and Mihael Arčan. 2015. Identifying Main Obstacles for Statistical Machine Translation of Morp hologically Rich South Slavic languages. In *Proceedings of the 18th Annual Conference of the European Associati on for Machine Translation (EAMT 2015)*. Antalya, Turkey.

Luz Rello. 2012. Dyswebxia: a model to improve accessibility of the textual web for dyslexic users. In *SIGACCESS Access. Comput.*, ACM, New York, NY, USA, 102, pages 41–44.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. Making It Simplext: Implementation and Evaluation of a Text Simplification System for Spanish. *ACM Transactions on Accessible Computing* 6(4):14:1–14:36.

Advaith Siddharthan and M. A. Angrosh. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. pages 722–731.

Jorg Tiedemann. 2009. News from OPUS – A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In *Advances in Natural Language Processing*, Borovets, Bulgaria, volume V, chapter V, pages 237–248.

Shruti Tyagi, Deepti Chopra, and Iti Mathur. 2015. Classifier based text simplification for improved machine translation. In *Proceedings of International Conference on Advances in Computer Engineering and Applications (ICACEA), Ghaziabad, India*. pages 46–50.

Francis M. Tyers and Murat Alperen. 2010. South-East European Times: A parallel corpus of the Balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and (South-) Eastern European Languages*. Valetta, Malta, pages 49–53.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL&HLT*. volume 344–352.

Sanja Štajner and Goran Glavaš. 2017. Leveraging event-based semantics for automated text simplification. *Expert Systems with Applications* 82:383–395.

Sanja Štajner and Maja Popović. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation (EAMT 2016)*. Riga, Latvia, pages 230–242.

Sanja Štajner, Maja Popović, Horacio Saggion, Lucia Specia, and Mark Fishel. 2016. Shared Task on Quality Assessment for Text Simplification. In *Proceedings of the LREC Workshop on Quality Assessment for Text Simplification (QATS)*. pages 22–31.

Investigating Multilingual Abusive Language Detection: A Cautionary Tale

Kenneth Steimel, Daniel Dakota, Yue Chen, Sandra Kübler
Indiana University
{ksteimel,ddakota,yc59,skuebler}@indiana.edu

Abstract

Abusive language detection has received much attention in the last years, and recent approaches perform the task in a number of different languages. We investigate which factors have an effect on multilingual settings, focusing on the compatibility of data and annotations. In the current paper, we focus on English and German. Our findings show large differences in performance between the two languages.

We find that the best performance is achieved by different classification algorithms. Sampling to address class imbalance issues is detrimental for German and beneficial for English. The only similarity that we find is that neither data set shows clear topics when we compare the results of topic modeling to the gold standard. Based on our findings, we can conclude that a multilingual optimization of classifiers is not possible even in settings where comparable data sets are used.

1 Introduction

The last decade has seen a massive increase in user generated content in social media. While most people are interested in connecting with family and friends and in exchanging experiences, there is an increasing number of posts that cross the line from sharing negative opinions to becoming abusive. Since the data is too massive for manual filtering, automated methods to detect abusive language reliably are required. This has created a novel research area under the titles of abusive language detection, hate speech detection, flame or cyberbullying detection.

While most of the work on abusive language detection has focused on English (Schmidt and Wiegand, 2017; Park and Fung, 2017; Lee et al., 2018), there is some work on other languages, and first attempts have also been made to develop methods that work across different languages (Fehn Unsvåg and Gambäck, 2018).

Our interest also focuses on multilingual abusive language detection. However, before we engage in a full scale investigation of which methods work well across multiple languages, we need to know more about which factors have an effect on multilingual settings, including but not restricted to the compatibility of data and annotations, differences between languages, and topic effects. In the current paper, we focus on two languages, English and German, where we have access to similar data (see section 3 for more information). We approach the following questions as we investigate an approach across the two languages:

1. Do classifiers behave similarly across the two languages? I.e., can we establish the best classifier on one language and then use it successfully for the second language?

2. Can we determine which types of features are necessary for a classifier? Are the types of features and the number of features comparable across the two languages?

3. The data sets are skewed towards non-abusive language, and research in sentiment analysis has shown that over-sampling methods can improve results (Liu et al., 2014). Thus, do over-sampling methods show consistent results across both languages?

4. For tasks related to sentiment analysis, it is often the case that a classifier learns topic information rather that sentiment (Pang et al., 2002). We investigate whether the two languages show similar effects with regard to topics.

Proceedings of Recent Advances in Natural Language Processing, pages 1151–1160,
Varna, Bulgaria, Sep 2–4 2019.

Our results show that the data sets differ in their answers to questions 1–3, only showing similarities with regard to topics, leading us to the the preliminary conclusion that we cannot transfer methodology across languages and data sets when the data sets for the individual languages have been collected opportunistically. Since it is highly unlikely that we can completely replicate the data collection methods from the "source" language, the implications of our findings are far reaching and necessitate further investigation into the issues of multilingual abusive language detection.

The remainder of the paper is structured as follows: We discuss related work in section 2 and the data sets in section 3. Then, we explain our experimental setup and feature sets in section 4. Section 5 presents the results, and section 6 draws conclusions and discusses future work.

2 Related Work

Research on abusive language detection has recently drawn much attention, as several recent shared tasks (Basile et al., 2019; Kumar et al., 2018; Wiegand et al., 2018; Zampieri et al., 2019) demonstrate. So far, research has mostly focused on English. For a comprehensive survey of NLP techniques to detect hate speech see (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018). Here, we will focus on issues relating to problems in multilingual abusive language detection.

There is work on abusive language detection in Arabic, Dutch, and German. For Arabic, Mubarak et al. (2017) developed a data set of abusive language by creating an abusive word list and then splitting their Twitter dump into abusive and non-abusive classes by user, where users are considered abusive if they have used at least one word from the list. They also created an algorithm to extend the list of abusive words.

Cyberbullying detection in Dutch social media was performed by Van Hee et al. (2015). They collected data from Ask.fm and annotated it with 7 categories including 'Threat', 'Insult', and 'Defamation'. For their classifier, they used a bag-of-words approach for word and character n-grams and a sentiment lexicon, with an SVM classifier. They found that sentiment features alone yield poor results, as does a substantial reduction in the number of bag-of-word features due to data sparsity.

The 2018 Germeval shared task focused exclusively on detecting abusive language in German tweets (Wiegand et al., 2018) (for a description of the data set, see section 3). While many of the best systems used neural architectures, the winning system (Montani and Schüller, 2018) used an ensemble method with classifiers trained on a subset of features (such as TF-IDF and character n-grams). The predictions were then combined with a maximum entropy model for a final prediction. They found that strategies such as feature selection, sampling, and extensive preprocessing ultimately reduced performance in their ensemble.

While most work focuses on identifying abusive language in a specific language, Fehn Unsvåg and Gambäck (2018) examined how a single approach can handle abusive language across multiple languages: English, Portuguese, and German. They used a number of existing twitter corpora including the corpus by Waseem and Hovy (2016) for English[1], the one by Ross et al. (2016) for German, and the one by Fortuna (2017) for Portuguese. They also incorporated "user features" (i.e., specific demographic information known about the author of the tweet) along with a standard set of word and character n-gram features using logistic regression. They noted slight improvements but only specific user features contributed improvements to a given language (e.g., network features boosting English and Portuguese).

Several systems for detecting abusive language in Hindi and English were developed as part of the 2018 *Trolling, Aggression and Cyberbullying* shared task (Kumar et al., 2018). This shared task used data from Facebook and Twitter. The systems had to label examples as 'Not Aggressive', 'Covertly Aggressive', and 'Overtly Aggressive'. Modha et al. (2018) achieved the best results on the Hindi side of the shared task using a convolutional neural network with fastText embeddings (Mikolov et al., 2018). Galery et al. (2018) identified abusive language in the English portion of the corpus as their initial survey found code-switching between Hindi and English. To address this code-switching, they used fastText embeddings for both languages and a SVD transformation of these two sets of embeddings to generate multilingual embeddings using sub-word units. These multilingual embeddings were used in a GRU-based recurrent neural network. They found that this approach was not effective on their particular data set due to

[1] This is the same English corpus used in the present work.

English	German
These girls are the equivalent of the irritating Asian girls a couple years ago. Well "done," 7 #MKR	@tagesschau Euere AfD Hetze wirkt. Da könnt ihr stolz sein bei #ARD-Fernsehen (Eng.: @tagesschau Your AfD baiting works. You can be so proud at #ARD-TV)
@ameedjadallah I read the entire Quran. Islam is what is wrong.	@welt Bla bla bla! Lügenpresse verkauft uns mal wider für Dumm! Alles gespieltes Theater!! (Eng.: @welt Blah blah blah! Lying press is taking us for fools again. It's all totally staged!!)

Table 1: Sample abusive tweets for English and German

	Abusive	Non-Abusive
English		
Train	4 460	9 683
Test	496	1 076
Total	4 956	10 759
Total in %	31	69
German		
Train	1 517	2 992
Test	171	329
Total	1 688	3 321
Total in %	34	66

Table 2: Distribution of binary class labels in the English and German data sets

the limited instances of code-switching present.

3 Data Sets

For our work, we chose two data sets that were as similar as possible without creating a new, tightly controlled bilingual data set. For English, we used the publicly available Twitter hate speech data set created by Waseem and Hovy (2016). The original corpus included approximately 16 000 tweets; however we were only able to retrieve 15 715 tweets using the Twitter API, the rest were unavailable or deleted. The English data set was manually annotated with three different labels: 'racism', 'sexism' and 'none', where none refers to non-hate speech. For more information regarding the annotation guidelines, see (Waseem and Hovy, 2016). For classification, we split the data set using 90% (14 143 tweets) as training data and 10% (1 572 tweets) as test set. Every tenth tweet was assigned to the test set and removed from the training set to ensure the data sets were drawn from the whole corpus.

For the German data, we used the 2018 GermEval shared task data set (Wiegand et al., 2018) with the annotations for task 1 of the shared task. The annotations are binary: examples are either labeled as 'offensive' or 'other'. For our work, we use the training set of the shared task, which consists of 5 009 samples, and split it into our training and test sets, taking every tenth example for the test set. We do not use the official GermEval test set since we basically perform optimization experiments, and any scores obtained on the test data should not be directly compared to other research.

Since the English data set consists of a ternary classification while the German set uses a binary classification, we simplify the English data in order to make the data sets consistent across the two languages: We group 'racism' and 'sexism' into an 'abusive' group while the 'none' labels are maintained. Examples of the abusive class are shown in table 1, and a summary of the class distributions in the two data sets is shown in table 2.

The imbalance between classes is clear given the data in table 2. The non-abusive data outnumber the abusive data in a ratio of about 2:1 for both languages. This will negatively affect the performance of the classifiers since statistical classifiers tend to predict the majority class.

4 Methodology

We have developed two pipelines for detecting abusive language in tweets: One pipeline is trained on German data and the other is trained on English data.

Classifiers. For the first set of experiments, we use a range of classifiers including random forest, SVM, XGBoost and a neural network approach. For the former three classifiers, we use scikit-learn (Pedregosa et al., 2011), for the neural network architecture tensorflow's Keras API (Abadi et al., 2016). Based on previous research on related tasks (Park and Fung, 2017; Badjatiya et al., 2017), we experiment with several promising architectures, including fully connected neural networks and convolutional neural networks, along with different word embeddings, with both BERT and Flair embeddings (Devlin et al., 2018; Akbik et al., 2018), stand-alone and stacked respectively.

For the scikit-learn classifiers, we optimized hyper-parameters using grid search. For the neural networks, dropout and batch normalization techniques are applied, and architecture selection and hyper-parameter optimization are done in a non-exhaustive search.

Features. We use simple character n-grams along with stemmed word n-grams and dependency parse-derived features.

Stemming. Since we were unable to identify a good lemmatizer for German Twitter data, we decided to implement a stemmer for both the English and German data, in order to maintain compatibility between the two languages. The YASS stemmer (Majumder et al., 2007) is an unsupervised stemming algorithm that generates a minimum spanning tree of words based upon different string similarity distance metrics, cuts the hierarchy, and then stems a word by replacing the word with the centroid of the cluster the word belongs to. We use the YASS stemming method with a minor modification: While the YASS stemmer replaces all words that belong to a cluster with the cluster centroid, we replace all cluster members with the shortest member of the cluster. Stems are shorter than their morphologically related forms since affixes are not present. However, this step is not a departure from the YASS algorithm, instead it is an adaptation to increase the effectiveness of this algorithm on our particular domain. In addition, numbers, twitter handles, and URLs are removed from the data prior to stemming.

The distance metrics used by Majumder et al. (2007) rely heavily on the suffixing nature of German and English inflectional morphology. While these distance metrics fall flat for non-suffix based inflectional morphology like irregular past tense (primarily ablaut grades in words like ‘sleep’, ‘slept’), they produce less spurious stems compared to other common metrics like the Levenshtein distance. We use distance metric 4 from the YASS stemmer.

Dependency parsing features. To extract dependency features for English, we use the Tweebo parser (Kong et al., 2014), which is designed to parse Twitter data and requires minimal preprocessing to obtain useful parses. Unlike English, German does not possess a Twitter domain specific parser. For this reason, we use the Mate parser pipeline (Björkelund et al., 2010). However, in order to maximize the usefulness of Mate, which expects a more standard text structure, we preprocess the German Twitter data.

Preprocessing steps for parsing include: removing one or more hashtags or retweets after punctuation at the end of tweet as well as removing initial hashtags and retweets, removing the # sign from any hashtag in the middle of the tweet, removing all emojis, and detaching punctuation from words. We also use a base list of abbreviations[2] and add additional ones to ensure that these are kept during the tokenization process.

We extract dependency triples consisting of (dependent, head, label) that occurred a minimum of five times as features. These features are Boolean valued, denoting their presence or absence in a tweet.

Sampling methods. Given the imbalanced nature of the data sets, we investigate sampling techniques to examine whether sampling can yield better performance. We use imbalanced-learn (Lemaître et al., 2017) to perform both under-sampling and over-sampling techniques. More specifically, we use four over-sampling and two under-sampling methods: SMOTE (Chawla et al., 2002) constructs synthetic examples of the minority class by averaging over two randomly chosen minority examples. Borderline SMOTE (Han et al., 2005) focuses on the area around the decision boundary to create new examples, SVM SMOTE (Nguyen et al., 2011) uses an SVM to determine the examples to average while ADASYN (He et al., 2008) concentrates on the areas of the minority class search space that are difficult. Edited Nearest Neighbors (ENN; Wilson, 1972) uses a k-NN approach to identify examples that are untypical for their neighborhood; these are then deleted. One sided selection (Kubat and Matwin, 1997) uses Tomek links to identify and delete noisy examples.

Topic modeling. We train a topic modeler on the two data sets, i.e., we create an LDA (Blei et al., 2003) topic modeler per language and use it to group tweets into two topics. We have preprocessed the tweets in the same manner that was

[2] Taken from Stefanie Dipper’s Perl German Tokenizer and found at `https://www.linguistics.ruhr-uni-bochum.de/~dipper/pub/software/abbrev.lex`

Classifier	Prec	Rec	F-Score
majority class	34.22	50.00	40.63
RF	80.67	74.17	76.08
XGBoost	83.46	78.80	80.49
SVM	82.11	66.58	68.20
NN	34.22	50.00	40.63

Table 3: English results for a range of classifiers.

Classifier	Prec	Rec	F-Score
majority class	32.90	50.00	39.69
RF	66.00	66.50	66.50
XGBoost	68.50	60.00	59.50
SVM	74.41	70.97	72.01
NN	32.90	50.00	39.69

Table 4: German results for a range of classifiers

used for dependency parsing. We then use the top 100 words by frequency to produce two topics.

Evaluation. We report precision, recall, and F_1. Because of the skewing in the data set, all these measures are calculated as the macro average of precision, recall, and F_1, i.e., we calculate the measures per class and then average across both classes.

5 Results

5.1 Classifier Behavior across Languages

Here we investigate whether the classifiers show the same trends across both languages. This will allow us to decide whether we can select the classifier for one language and then keep the selection stable across further languages. For these experiments, we restrict the feature set to using only character n-grams of length 2-7 with a minimum frequency of 3, resulting in 137 434 features for German and 282 507 for English.

The results of the experiments for English are shown in table 3, the results for German in table 4. These results show that classifying all tweets as non-abusive, the majority class, results in a macro-F of around 40% for both languages. All classifiers but the neural networks perform better. All the neural network architecture/embedding combinations predict the majority class throughout.

For the other classifiers, we reach higher F-scores for English than for German: 80.49. vs. 72.01. This is not unexpected, since the English data set is larger than the German one. However, the best English results are reached by XGBoost while the same classifier only reaches 59.50 F for German. The highest results for German are reached by the SVM, which reaches a lower F-score on English (68.20). It is also interesting to see that for English, XGBoost reaches a recall that is more than 12 points higher than the SVM. In contrast, for German, XGBoost's recall is the lowest of all classifiers (except for the neural networks). For the SVM, recall is higher for German than for English, thus going against the general trend, but the difference is much less pronounced (> 4 points).

From these results, we draw the conclusion that we cannot choose a classifier for a new language based on experience with another language.

5.2 Feature Selection across Languages

The next set of experiments is concerned with the question of whether we can use the same features across languages, or if each language requires its own set of informative features. For these experiments, we decided to focus on SVMs since they show good performance and similar trends across the languages in the comparison of classifiers above and since they train much faster than XGBoost. We add two additional feature types into the vectors: stems and dependency features. For German, this results in a total number of features of 148 322 and for English, in 308 323.

We use Information Gain (IG) for feature selection and perform experiments using the set of features with the highest IG in incremental cutoffs. This results in a different amount of features for English and German, but this difference can be explained by the differences in the morphological complexity of the two languages and the data sizes. Results for English are reported in table 5, showing the overall results across both classes and specifically for the abusive class. Results for German are reported in table 6. The last row in each table repeats the results from the previous section, i.e., using all character n-gram features.

For English, a comparison of the two experiments with the character n-grams only as opposed to the full feature set including stems and dependency features shows that adding these features has a minimal negative effect, lowering the F-score from 68.20 to 67.70.

For the experiments on English using feature selection, we see that results with even 2 660 features improve significantly over the all features

		Overall			Abusive		
IG threshold	Num. IG features	Prec	Rec	F	Prec	Rec	F
0.000075	2660	79.38	72.62	74.49	77.95	52.02	62.39
0.00005	4232	80.29	0.74	0.76	78.95	54.44	64.44
0.000025	9305	80.72	74.27	76.18	79.59	55.04	65.08
0.00001	24350	82.26	76.48	**78.36**	81.21	59.27	**68.53**
0.0000075	33187	82.64	75.99	78.04	82.42	57.66	67.85
0.000005	60000	83.06	75.10	77.33	84.00	55.04	66.50
–	all features	81.87	66.18	67.70	87.31	34.68	49.64
–	only char n-grams	82.11	66.58	68.20	87.56	35.48	50.50

Table 5: English results with IG feature selection, overall and for the abusive class.

		Overall			Abusive		
IG threshold	Num. IG features	Prec	Rec	F	Prec	Rec	F
0.005	266	66.18	58.76	58.02	61.97	25.73	36.36
0.004	451	65.54	60.88	61.10	59.18	33.92	43.12
0.003	788	67.59	62.79	63.30	62.14	37.43	46.72
0.002	2 338	66.19	62.74	63.27	59.13	39.77	47.55
0.0016	4 071	67.42	65.19	65.80	59.70	46.78	52.46
0.0014	6 404	68.70	66.23	66.92	61.65	47.95	53.95
0.0011	9 690	69.68	67.40	**68.10**	62.77	50.29	**55.84**
0.0008	16 791	70.21	65.71	66.56	65.49	43.27	52.11
0.0006	26 801	69.62	66.26	67.06	63.71	46.20	53.56
0.0004	48 014	72.84	68.93	**69.95**	68.55	49.71	**57.63**
0.0002	69 541	75.21	72.28	**73.26**	70.80	56.73	**62.99**
0.0001	101 605	74.92	72.13	73.07	70.29	56.73	62.78
–	all features	74.71	71.13	72.20	70.77	53.80	61.13
–	only char n-grams	74.41	70.97	72.01	70.23	53.80	60.93

Table 6: German results with IG feature selection, overall and for the abusive class.

baseline, and they continue their upward trend until around 14 000 features. At this point, however, a downward trend begins, suggesting that for English, a lower number of important features for the SVM classifier is beneficial. The results for the minority class follow the same trend: They also reach their peak at around 14 000 features. This means that the classifier's performance on the minority class is the driving factor (which is partly a result of using the macro-averaged values).

For German, the addition of stems and dependencies has a minimal positive effect, increasing the F-score from 70.01 to 72.20. For the experiments on feature selection, we see a steady upward trend as the number of features increases until we reach around 70 000 features, at which point results start to decrease. Again, the results on the abusive class follow the same trend. This means that, at a certain point, we reach a saturation of features in terms of modeling the abusive class. If we add more features, the classifier suffers from irrelevant features.

When we compare the results across the two languages, we notice a discrepancy in that the stems and dependencies help for German while they are harmful for English. Both effects are minimal. One possible explanation may be that for English, the additional features only increase data sparsity without providing novel information. For German, which is morphologically richer and has freer word order. Introducing the stems and dependencies may help alleviate data sparsity to a certain degree. This requires further investigation.

The second discrepancy concerns the optimal number of features. For German, we achieve our best results using slightly less than half the features; for English, the best results are based on approximately 4.5% of the data. This means that the results differ in terms of absolute numbers and percentage.

	Abusive		Non-Abusive		
Sampling method	Precision	Recall	Precision	Recall	F-score
No sampling	85.49	43.95	78.89	**96.56**	72.45
SMOTE	63.21	**76.21**	**87.89**	79.55	76.31
Borderline SMOTE	62.23	73.99	86.92	79.65	75.48
SVM SMOTE	62.46	73.79	86.82	79.55	75.34
ADASYN	61.19	74.40	86.89	78.25	74.75
Edit nearest neighbors	81.77	57.86	82.88	94.05	**77.94**
One sided selection	**85.60**	44.35	79.01	**96.56**	72.67

Table 7: Results for English using sampling (70 000 features).

	Abusive		Non-Abusive		
Sampling method	Precision	Recall	Precision	Recall	F-score
No sampling	**70.80**	56.73	79.61	**87.84**	**73.26**
SMOTE	58.17	52.05	76.37	80.55	66.67
Borderline SMOTE	60.26	54.97	77.62	81.16	68.42
SVM SMOTE	60.26	54.97	77.62	81.16	68.42
ADASYN	57.32	52.63	76.38	79.64	66.43
Edit nearest neighbors	56.81	**70.76**	**82.58**	72.04	69.98
One sided selection	69.57	56.14	79.28	87.23	72.60

Table 8: Results for German using sampling (69 541 features).

Consequently, we again come to the conclusion that we cannot generalize across languages with regard to which feature types are useful nor to the amount of features that are useful.

5.3 Sampling Methods across Languages

In this section, we look into the effects of sampling methods. Since the problem of abusive language detection is inherently skewed towards non-abusive language, instance sampling on the training set can help make the classifier more sensitive towards the minority class. We investigate the use of over-sampling of the minority class and under-sampling of the majority class using the best performing number of IG features for German (69541) and approximately the number of top IG features for English (70,000). The results of these experiments are shown in table 7 for English and in table 8 for German.

For English, table 7 shows that we reach the highest precision for the abusive class using one-sided selection, an under-sampling method. The highest recall for the abusive class and the highest precision for the non-abusive class are reached by SMOTE, an over-sampling method. The highest recall for the non-abusive class is reached without any sampling. The highest F-score across both methods (77.94) is reached by using edit nearest neighbors, which reaches a somewhat lower precision on the abusive class than one-sided selection, but a recall that is about 12.5 points higher.

For German, table 8 shows a different picture: We reach the highest precision for the abusive class along with the highest recall for the non-abusive class and the highest overall F-score (73.26) in the experiments without any sampling. The highest recall for the abusive class along with the highest precision for the non-abusive class is reached by the edit nearest neighbors under-sampling method.

These results show that we again face a situation in which the two data sets behave completely differently. For English, we reach the best results with an under-sampling method while for German, all of the sampling methods perform worse that not using sampling at all. Additionally, while the experiments without sampling show the same trends – high precision for abusive language and high recall for non-abusive language – across both languages, this is not the case for edit nearest neighbors: Here the English results show a high precision for the abusive class, but the German results show high recall for the same class.

	Abusive		Non-Abusive		
Language	Precision	Recall	Precision	Recall	F-score
English	33.98	53.23	70.82	52.32	52.59
German	36.97	51.46	68.32	54.41	51.80

Table 9: LDA Topic Modeling Classifications

5.4 Topic Behavior across Languages

One important distinction that needs to be better understood during the classification process is whether our classifiers detect abusive language itself, or whether they detect topics that are more likely to induce abusive language. If certain topics are strongly associated with abusive language in the training data then the model obtained may focus on these topic associations and not model abusive language directly. More specifically, if this trend is more pronounced for one language, this may explain some of the differences in results across the two languages that we found in previous sections. In order to investigate this possibility, we perform topic modeling on the tweets of the two data sets. This experiment is based on the assumption that if the topic modeler groups tweets similar to the abusive, non-abusive classes, then we have evidence that the abusive language is closely associated with a content topic. We determine the overlap between the topic modeler and the abusive/non-abusive split by calculating precision, recall, and F_1 between the topic models decisions and the gold standard.

We perform topic modeling with the number of topics set to 2, parallel to the grouping into abusive and non-abusive language. However, the topic modeler does not tell us which of the two topics corresponds to abusive language. Thus we calculate precision and recall for both correspondences and choose the one with the higher F-score.

The outcomes of this comparison are presented in table 9. The table shows that both languages follow the same trend with an F-score slightly higher than chance (52.59 for English and 51.80 for German). For both languages, recall is around 50, and precision is higher for the non-abusive class and lower for the abusive one. We conclude from these results that there is no meaningful overlap between topics and abusive/non-abusive language. I.e., our SVM classifiers do learn characteristics of abusive and non-abusive language rather than characteristics of topics. However, this also means that the differences between the languages found in the previous experiments cannot be explained by differences in associating abusive language with specific topics, and we need to investigate further to determine the source of those differences.

6 Conclusion and Future Work

In this paper, we have started an in-depth investigation into two data sets for abusive language detection, one in English and one in German. While the data sets were collected independently and annotated with different classes, collapsing them into a binary annotation between abusive and non-abusive language resulted in data sets that superficially showed similar characteristics. However, our investigation has shown notable differences: For English, XGBoost provides the best performance, for German SVMs do. Stems and dependencies in addition to character n-grams help for German while they are harmful for English. For German, we need to use slightly less than half of all features while for English, we only need 4.5%. Even though the English data set is larger than the German one, the percentages translate into a much larger feature set for German than for English. Additionally, for English sampling improves results while for German, it does not. One hypothesis, namely that the differences may be related to a stronger topical effect in the abusive tweets, was rejected based on an experiment with a topic modeler.

Moving forward, we will investigate the differences between the German and English data sets in more detail. Finding the causes will allow us to better approach optimization of multilingual abusive language detection systems. The final goal of this work is the development of a system for detecting abusive language that can truly work in a multilingual fashion.

Acknowledgements

We are grateful to Noor Abo Mokh, Leah Schaede and Zuoyu Tian, who helped in initial stages of the project.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. pages 265–283.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*. pages 1638–1649.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*. pages 759–760.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. pages 54–63.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Beijing, China, pages 33–36.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* 16:321–357.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on Twitter hate speech detection. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium, pages 75–85.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)* 51(4):85.

Paula Cristina Teixeira Fortuna. 2017. *Automatic detection of hate speech in text: an overview of the topic and dataset annotation with hierarchical classes*. Master's thesis, Universidade de Porto.

Thiago Galery, Efstathios Charitos, and Ye Tian. 2018. Aggression identification and multi lingual word embeddings. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. pages 74–79.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pages 878–887.

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the 5th IEEE International Joint Conference on Neural Networks*. Hong Kong, pages 1322–1328.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1001–1012.

Miroslav Kubat and Stan Matwin. 1997. Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the 14th International Conference on Machine Learning*. Nashville, TN, volume 97, pages 179–186.

Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. pages 1–11.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on Twitter. In *Proceedings of the Second Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium, pages 101–106.

Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research* 18(17):1–5.

Can Liu, Sandra Kübler, and Ning Yu. 2014. Feature selection for highly skewed sentiment analysis tasks. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*. Dublin, Ireland, pages 2–11.

Prasenjit Majumder, Mandar Mitra, Swapan K Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. YASS: Yet another suffix stripper. *ACM Transactions on Information Systems (TOIS)* 25(4).

Tomas Mikolov, Piotr Bojanowski Edouard Grave, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Sandip Modha, Prasenjit Majumder, and Thomas Mandl. 2018. Filtering aggression from the multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*. pages 199–207.

Joaquín Padilla Montani and Peter Schüller. 2018. TUWienKBS at GermEval 2018: German abusive tweet detection. In *Proceedings of GermEval 2018, 14th Conference on Natural Language Processing (KONVENS 2018)*. Vienna, Austria, pages 45–50.

Hamdy Mubarak, Kareem Darwish, and Walid Magdy. 2017. Abusive language detection on Arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, Canada, pages 52–56.

Hien M. Nguyen, Eric W. Cooper, and Katsuari Kamei. 2011. Borderline over-sampling for imbalanced data classification. *Journal International Journal of Knowledge Engineering and Soft Data Paradigms* 3(1):4–21.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing*. pages 79–86.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*. Vancouver, Canada, pages 41–45.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurosky, and Michael Wojatzki. 2016. Measuring the reliability of hate speech annotations: The case of the European refugee crisis. In *Proceedings of the Workshop on Natural Language Processing for Computer- Mediated Communication (NLP4CMC)*. Bochum, Germany, pages 6–9.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. Valencia, Spain, pages 1–10.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. pages 672–680.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. San Diego, CA, pages 88–93.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the GermEval 2018 shared task on the identification of offensive language. In *Proceedings of GermEval 2018*. Vienna, Austria.

Dennis L. Wilson. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics* 2(3):408–421.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*.

Augmenting a BiLSTM Tagger with a Morphological Lexicon and a Lexical Category Identification Step

Steinþór Steingrímsson[1,2] **Örvar Kárason**[1] **Hrafn Loftsson**[1]
[1]Department of Computer Science, Reykjavik University, Iceland
[2]The Árni Magnússon Institute for Icelandic Studies, Reykjavik, Iceland
{steinthor18,orvark13,hrafn}@ru.is

Abstract

Previous work on using BiLSTM models for PoS tagging has primarily focused on small tagsets. We evaluate BiLSTM models for tagging Icelandic, a morphologically rich language, using a relatively large tagset. Our baseline BiLSTM model achieves higher accuracy than any previously published tagger not taking advantage of a morphological lexicon. When we extend the model by incorporating such data, we outperform previous state-of-the-art results by a significant margin. We also report on work in progress that attempts to address the problem of data sparsity inherent in morphologically detailed, fine-grained tagsets. We experiment with training a separate model on only the lexical category and using the coarse-grained output tag as an input for the main model. This method further increases the accuracy and reduces the tagging errors by 21.3% compared to previous state-of-the-art results. Finally, we train and test our tagger on a new gold standard for Icelandic.

1 Introduction

Bidirectional long short-term memory (BiLSTM) models have in recent years been shown to be effective for various sequential labelling tasks, including Part-of-Speech (PoS) tagging (Ling et al., 2015; Plank et al., 2016).

BiLSTMs are an extension of general LSTMs (Hochreiter and Schmidhuber, 1997) that perform better on sequences where the complete input sequence is available. Two LSTMs are trained on the input sequence, one on its natural reading order and the other on its reverse (Graves and Schmidhuber, 2005). In addition to word embeddings (WE), character embeddings were first used for tagging with BiLSTMs by Dos Santos and Zadrozny (2014). This entails not only examining the sequence of words in a sentence during training but also the sequences of characters within those words.

In this paper we use BiLSTM models, with both word and character embeddings, to train a PoS tagger for a morphologically rich language, Icelandic, with a fine-grained tagset of 565 morphosyntactic tags. Only a small portion of previous work using neural networks for PoS tagging has focused on languages with rich morphology and large tagsets, e.g. Sagot and Martínez Alonso (2017).

Various taggers have been developed for Icelandic: data-driven taggers (Helgadóttir, 2005), a rule-based tagger (IceTagger) (Loftsson, 2008), and a hybrid tagger (Loftsson et al., 2009). Prior to the work presented here, an averaged perceptron tagger, IceStagger (Loftsson and Östling, 2013), was the current state-of-the-art tagger, achieving an accuracy of 93.84% by employing a morphological lexicon and external word embeddings.

This paper presents the first deep neural network tagger for Icelandic. We evaluate three models. First, we confirm the effectiveness of a BiLSTM model for PoS tagging using a fine-grained tagset. Second, we supplement the base model with an external morphological lexicon, thereby obtaining state-of-the-art results. Third, we propose an approach to further increase the accuracy by creating a coarse-grained tagset from the fine-grained one and using the resulting tagset to devise a two-step process. This approach is to our best knowledge novel in the context of neural network tagging. Specifically, we train a separate model on only the lexical category and use the coarse-grained output tag as an input into the main model. Combined, this results in an overall tagging accuracy of 95.15%, which is equivalent to

Proceedings of Recent Advances in Natural Language Processing, pages 1161–1168,
Varna, Bulgaria, Sep 2–4 2019.

an error reduction of 21.3% compared to the previous state-of-the-art. Finally, we train and test our model on a new gold standard for Icelandic, MIM-GOLD. The new standard is larger than the older one, IFD (see Section 2), and contains more diverse texts. We achieve an accuracy of 94.17% on MIM-GOLD.

2 Data

In this section, we describe the data and the tagset used in our work.

The IFD Corpus: The taggers developed for Icelandic so far have all been trained and tested on the Icelandic Frequency Dictionary (IFD) corpus (Pind et al., 1991), a balanced corpus containing about 590 thousand tokens. The IFD corpus was collected in the early 1990s and contains texts from published books, primarily fiction (60%) but also biographies (20%) and scholarly work (20%). As with the other taggers referenced in this paper, we use the so-called *corrected version* of the corpus, with the reduced tagset (565 tags) and ten-fold split from Loftsson et al. (2009).[1] The morphosyntactic tags in this tagset are mnemonic encodings, i.e. character strings where each character has a particular function. The first character denotes the *lexical category*. For each category there is a predefined number of additional characters (at most six), which describe morphological features, like *gender*, *number* and *case* for nouns; *degree* and *declension* for adjectives; *voice*, *mood* and *tense* for verbs, etc. To illustrate, consider the word form *maður* "man". The corresponding tag is *nken*, denoting noun (*n*), masculine (*k*), singular (*e*), and nominative (*n*) case.

The MIM-GOLD Corpus: MIM-GOLD[2] (Loftsson et al., 2010), a subset of the MIM corpus (Helgadóttir et al., 2012), contains a greater diversity of texts than the IFD corpus. In addition to texts from published books, it contains texts from news media, blogs, parliamentary speeches and more. Furthermore, MIM-GOLD contains approximately 1 million running words, about twice as many as IFD. The tagset used in MIM-GOLD consists of the same reduced tagset of 565 tags, mentioned above.

Morphological Lexicon: The Database of Modern Icelandic Inflections (DMII) is a lexicon of about 280 thousand paradigms and close to six million inflectional forms (Bjarnadóttir, 2012). The output from the database used in this project contains word form and morphological features. By incorporating DMII, the average unknown word rate in testing, using the IFD ten-fold split, drops from 6.8% to 1.1% (Loftsson et al., 2011).

3 The Three Models

3.1 Word and Character Embeddings

Word embeddings are vector representations of words based on their context in training data. Adding recurrent character embeddings has been shown to significantly improve performance for handling of unknown words (e.g. Plank et al. 2016; Dos Santos and Zadrozny 2014). For each word, both forward and backward expressions are generated, containing the sequence of characters in the word, as well as word initial and word final markers. This helps the model grasp morphological details.

In our baseline model, which is similar to Plank et al. (2016), both word embeddings and recurrent character embeddings are used as input. The character embeddings for a given word are input into a BiLSTM. The output from the BiLSTM is concatenated to the word embedding and the combined vector input into another BiLSTM, whose output is input into a hidden layer. The hidden layer feeds the output layer, which selects a PoS tag.

3.2 Using Data from an External Morphological Lexicon

Horsmann and Zesch (2017) replicated Plank et al. (2016) using a collection of corpora annotated with fine-grained tagsets of varying sizes, in contrast to the coarse-grained Universal Dependencies (UD) tagset in the previous study (17 tags). The replication confirmed the superior performance of the BiLSTM tagger, also on fine-grained tagsets. Furthermore, they found that the advantages of the BiLSTM tagger over other taggers grow proportionally with the tagset size of the corpus. However, they also claim that for large tagsets of morphologically rich languages, hand-crafted morphological lexicons are still necessary to reach state-of-the-art performance.

Using a morphological lexicon has become common practice for enriching training data for PoS taggers. Hajič (2000) marked the importance

[1] IFD can be downloaded from `http://malfong.is/?pg=ordtidnibok`.

[2] MIM-GOLD can be downloaded from `http://malfong.is/?pg=gull`.

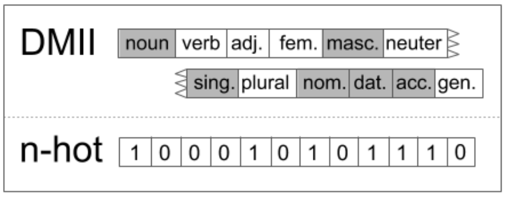


Figure 1: A partial n-hot vector and the corresponding features from DMII. The example shows 12 features, including the active features for the word form *strætó* "bus". The word, a noun, has the same form for nominative, dative and accusative and therefore all corresponding labels are activated. An actual vector in our model has 61 labels, which are either active, `1`, or inactive, `0`.

of this for morphologically rich languages. It was first done for Icelandic in Loftsson et al. (2011).

Sagot and Martínez Alonso (2017) first used morphological lexicons as supplemental input for PoS tagging with BiLSTM taggers and showed that it yields consistent improvement. Following their work, we extend the baseline model by adding an input layer that contains token-wise features obtained from the DMII lexicon (see Section 2). The input vector for a given word is an n-hot vector where each active value corresponds to one of 61 possible labels in the lexicon. An example of an n-hot vector is given in Figure 1.

The vector is concatenated to the two vectors described in the previous section, i.e. the word embedding and the character embedding, and the result is then fed into the BiLSTM layer. Previous taggers using DMII have had to map the information to the IFD tagset. As the tagsets of IFD and DMII are not completely compatible some information has been lost in the mapping process. Our method allows the model to use and learn from all the information encoded in the morphological lexicon, even though it uses a tagset slightly different from our training data.

3.3 Stepwise Tagging Model

When employing a fine-grained tagset with mnemonic encoding, the model does not place different significance on two tags when they differ in lexical category, on one hand, or share a lexical category but differ in morphological features, on the other. A human, however, would consider the former a more significant error than the latter. A PoS tagger is especially prone to such errors when the tagset is large and the amount of training data is insufficient to detect all the subtle differences between labels, as sometimes is the case for under-resourced or domain-specific languages.

To place a higher emphasis on assigning the correct lexical category, we devise a two-step process. First, we simplify the tagset from 565 to 10 tags by using only the first letter of the fine-grained tag mnemonic, i.e. the letter denoting the lexical category. We then train our model on this new coarse-grained tagset, using word and character embeddings as well as the morphological lexicon. This results in a lexical category tagger with very high accuracy, 98.97% in our case. In the second step, the output of that tagger is embedded as a one-hot vector and concatenated to the vectors input into the BiLSTM layer of the main model. This guides the tagger to the correct lexical category and eliminates some of the errors caused by insufficient training data.

This is a work in progress and other morphological features in the tags are promising for evolving this stepwise approach and further increasing overall accuracy. Thus, separate models for detecting gender, number and case agreement, for example, might be considered at each step.

We are not aware of other implementations of stepwise PoS tagging using BiLSTMs, but Horsmann and Zesch (2016) employ such a method in a slightly different setting. They use a Support Vector Machine for training, assume the coarse-grained tags are correct and then have their tagger assign the fine-grained tags based on them. In their Bidir tagger, Dredze and Wallenberg (2008) tag case separately in a second-pass, after running a general first-pass that uses the whole tagset. The second-pass tagger has access to the output of the first-pass, and is permitted to change its case and gender selections.

4 Experiments and Results

4.1 Experimental Setup

Our models were built using DyNet[3] (Neubig et al., 2017). We use the same hyperparameters for all models, SGD training with the initial learning rate of 0.13, which decays 5% in each epoch and runs for 30 epochs. The network has 128-dimensional embeddings for words and 20 for characters. The supplemental embeddings have 61

[3] The Dynamic Neural Network Toolkit, see `http://dynet.io`.

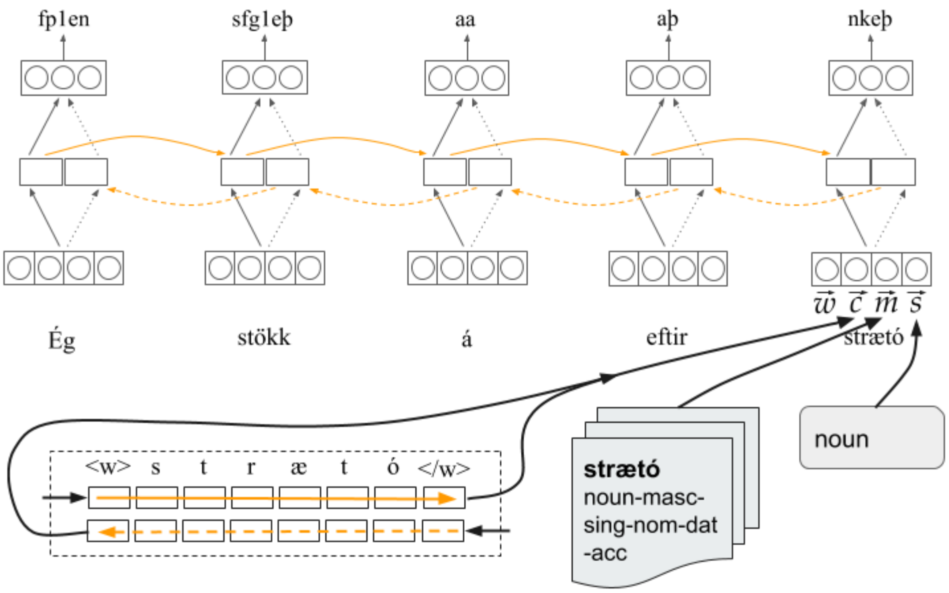


Figure 2: Our full model, employing word embeddings, character embeddings, a morphological lexicon, and the output of the first-pass of the stepwise model. The hidden layer is omitted for simplicity. Figure adapted from (Plank et al., 2016).

dimensions for the lexicon and 10 for the lexical categories. The hidden layer has 32 hidden states.[4]

Our experiments consist of three models:

Baseline: The first model uses word and character embeddings only. This corresponds to the model described in Section 3.1.

DMII: The second model adds external morphological data from DMII to the baseline model by encoding the information in n-hot vectors as described in Section 3.2.

LC: The third and full model then adds the lexical category embeddings created by a coarse-grained tagging step described in Section 3.3. This full model is shown in Figure 2.

	Acc.	Known	Unknown
Baseline	93.25%	95.19%	**66.84%**
+ DMII	94.84%	95.17%	54.61%
+ LC	**95.15%**	**95.48%**	54.06%

Table 1: Accuracy of the three models trained and tested on IFD. Note that when DMII is employed the number of unknown words falls almost 90%, from 4,036 to 476 out of an average total of 58,977 words in the splits.

4.2 Part-of-Speech Tagging Results

The test results for all three models are shown in Table 1, with the full model reaching 95.15% accuracy after 30 epochs. The baseline model (93.25%) already gets close to state-of-the-art results and surpasses existing taggers when not using an external morphological lexicon (cf. Loftsson and Östling 2013).

The substantial gain achieved by using DMII confirms the advantages of using an external morphological lexicon as discussed in Section 3.2. The accuracy gain is considerably higher than the corresponding gain in IceStagger (1.59 vs. 0.88 percentage points).

By employing the stepwise model discussed in Section 3.3 we try to guide the tagger to the highly accurate lexical category given by the coarse-grained tagger. This helps in assigning rare or ambiguous tags in the fine-grained tagset by raising the accuracy of the lexical category, resulting in a further 0.31 percentage point gain.

Note that the baseline model achieves the highest accuracy for unknown words because when adding data from DMII the unknown word ratio drops considerably (see Table 1), from 6.8%

[4] The source code for our implementation is available from https://github.com/steinst/ABLTagger

	Acc.	Known	Unknown
TnT	90.45%	91.82%	71.82%
IceTagger	92.73%	93.84%	**77.47%**
+ DMII	93.48%	93.85%	60.50%
IceStagger	92.82%	93.97%	77.03%
+ DMII	93.70%	94.02%	61.45%
+ DMII,WE	93.84%	94.15%	61.99%
Our model	**95.15%**	**95.48%**	54.06%

Table 2: Comparison to other taggers for Icelandic.

to 0.8%. This is in line with results of previous taggers (see Section 2), further reduction in unknown words is due to us using the latest version of DMII, while previous results were published in 2011. When DMII is employed the remaining unknown words are more likely to be foreign words, typos or to be irregular in some other way and therefore more difficult to tag. This explains the drop in accuracy for unknown words.

4.3 Comparison to Other Taggers

A comparison of our model to other previously published taggers for Icelandic is shown in Table 2. The results for TnT, IceTagger and IceStagger are presented in (Loftsson et al., 2009; Loftsson, 2008; Loftsson and Östling, 2013), respectively. All the reported results are fully comparable as they are based on exactly the same cross-validation split of the IFD corpus, with the exception that the TnT tagger does not employ data from DMII, and has therefore a higher ratio of unknown words.

Our model outperforms all previous taggers by a substantial margin, equaling a 21.3% reduction in errors compared to the highest accuracy (93.84%) obtained by IceStagger. It also has the highest accuracy for known words, i.e. those seen in the training data, including DMII. It should be noted though, that the numbers for accuracy of known and unknown words are not very well comparable between the different models, as using DMII eliminates a substantial part of unknown words, but the ones that remain tend to be more irregular, and can thus be harder to tag correctly.

4.4 Error Analysis

When comparing the most frequent kinds of errors our tagger makes to the errors of IceStagger, two differences stand out. The frequency of sng>sfg3fn and sfg3eþ>sfg1eþ are drastically reduced and are no longer among the ten most frequent kinds of errors (see Table 3). These are verbs that are assigned infinitive mood instead of indicative mood (sn...>sf...) and third person instead of first (sfg3...>sfg1...), respectively. These kinds of errors occur when the subject is far away from the verb itself and the more frequent tag for the word form is selected instead of the correct one. This corroborates that LSTMs are better at handling long-distance dependencies (Linzen et al., 2016) than other methods that have a limited context window during training.

	Our model		IceStagger
No.	Proposed tag > gold tag	Error rate	Proposed tag > gold tag
1.	aþ>ao	3.28%	aþ>ao
2.	ao>aþ	2.99%	ao>aþ
3.	nveo>nveþ	1.80%	nveo>nveþ
4.	nveþ>nveo	1.72%	nveþ>nveo
5.	ao>aa	1.18%	**sng>sfg3fn**
6.	aa>ao	1.09%	ao>aa
7.	nkeo>nkeþ	0.98%	**sfg3eþ>sfg1eþ**
8.	nheo>nhfo	0.92%	aa>ao
9.	nkeþ>nkeo	0.82%	nheo>nhen
10.	ct>c	0.81%	nhen>nheo

Table 3: Ten most frequent kinds of errors.

The remaining kinds of errors in the top ten list are for the most part mistakes in case assignment. For example, prepositions are often wrongly marked as governing accusative instead of dative and vice versa (1 and 2) and there is often a confusion between prepositions and adverbs (5 and 6). The same goes for nouns (7 to 9) and, in addition, they are often assigned the wrong number, i.e. singular instead of plural (8). The last kind of error (10) is caused by a lack of syntactic and contextual information: a conjunction is marked as a relativizer, i.e. conjunction introducing a relative clause.

The nearly even distributions (1+2, 3+4, 5+6, 7+9) at which these kinds of errors occur indicate that there is nothing in the training data to discern which tag to select in these instances. One way forward to try to tackle these errors is to supplement the model further, e.g. with verb subcategorization frames.

	Acc.	Known	Unknown
MIM-GOLD	94.04%	95.13%	68.34%
+ IFD	94.17%	95.62%	68.18%

Table 4: Accuracy when training and testing on MIM-GOLD.

5 Tagging a Different Gold Standard

In the previous sections, we have described the tagging process and compared the results to previous taggers using the same splits on IFD. We have demonstrated that our tagger achieves a significant gain in accuracy over previous taggers. Since the IFD corpus mainly contains literary work (see Section 2), these texts are not necessarily characteristic of texts that have to be tagged for language technology or research purposes. This is one of the reasons why a new gold standard, MIM-GOLD, was built containing more diverse texts (see Section 2). In 2015, Steingrímsson et al. (2015) trained IceStagger on MIM-GOLD, but found it had many inconsistencies and errors. Since then it has been reviewed and corrected and the final version, along with 10-fold splits, was made available in 2018.

We trained our BiLSTM tagger on these splits and measured the accuracy for our full model, employing both DMII and the two-step method. We carried out two experiments. In the first, we only trained and tested on the 10-fold splits for MIM-GOLD, but in the second we added the whole IFD corpus to the training data. As evident from Table 4, there is a substantial drop in accuracy compared to training and testing on IFD (see Table 1). The lower accuracy may, at least partly, be due to a greater variety in texts than before and a larger proportion of unknown words in the MIM-GOLD test set compared to IFD (Steingrímsson et al., 2015).

6 Conclusions and Future Work

We have shown that BiLSTM models with combined word and character embeddings achieve state-of-the-art accuracy in PoS tagging of Icelandic texts. We have also confirmed that BiLSTMs perform well with a fine-grained tagset, such as the one used in the Icelandic corpora, IFD and MIM-GOLD. When dealing with small corpora, as often is the case with under-resourced languages, supplementing the models with external data can be highly beneficial as shown by our experiments.

To deal with the problem of data sparsity, which is more prevalent when using fine-grained tagsets, we devised a stepwise method to guide the tagger in assigning lexical categories. This method is a work in progress – we have pinpointed morphological features that can be independently identified with very high accuracy and are therefore promising candidates for being handled in a separate step in the tagging process. Furthermore, it could be worthwhile to pre-train word embeddings on unlabeled data, such as the Icelandic Gigaword Corpus (IGC) of 1.2 billion words (Steingrímsson et al., 2018), e.g. employing the method described by Wang et al. (2015), which is specifically adapted to BiLSTMs.

The error analysis in Section 4.4 suggests that information on case governance is critical in reducing the most common errors the tagger makes. This could be external data on case governance of verbs and prepositions, or data derived from a method akin to the stepwise method that better discerns this information from the training data.

The final version of a new gold standard, MIM-GOLD, has recently been released and has not been used for training a PoS tagger for Icelandic before. IFD is heavily biased towards literary fiction but MIM-GOLD is a more balanced mix of different text genres and is thus more diverse. The lower accuracy for MIM-GOLD should thus not have been surprising, even though it has more data than IFD. Comparison of error analysis for both gold standards should reveal if there are other factors at play. We suggest that further work on developing PoS taggers for Icelandic texts focuses on this new gold standard.

References

Kristín Bjarnadóttir. 2012. The Database of Modern Icelandic Inflection. In *LREC 2012 Proceedings: Proceedings of Language Technology for Normalization of Less-Resourced Languages, SaLTMiL 8 – AfLaT*. https://www.aflat.org/files/saltmil8-aflat2012.pdf#page=25.

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *Proceedings of the* 31^{st} *International Conference on Machine Learning – Volume 32*. Beijing, China, ICML '14. http://dl.acm.org/citation.cfm?id=3044805.3045095.

Mark Dredze and Joel Wallenberg. 2008. Further results and analysis of Icelandic part of speech tagging. Technical report, Department of Computer and Information Science, University of Pennsylvania.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks* 18(5-6):602–610. https://doi.org/10.1016/j.neunet.2005.06.042.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In 6th *ANLP Conference / 1st NAACL Meeting. Proceedings*. Seattle, Washington. https://dl.acm.org/citation.cfm?id=974318.

Sigrún Helgadóttir, Ásta Svavarsdóttir, Eiríkur Rögnvaldsson, Kristín Bjarnadóttir, and Hrafn Loftsson. 2012. The Tagged Icelandic Corpus (MÍM). In *Proceedings of SaLTMiL-AfLaT Workshop on Language technology for normalisation of less-resourced languages*. Istanbul, Turkey, LREC 2012. https://www.aflat.org/files/saltmil8-aflat2012.pdf#page=79.

Sigrún Helgadóttir. 2005. Testing data-driven learning algorithms for PoS tagging of Icelandic. In H. Holmboe, editor, *Nordisk Sprogteknologi 2004*, Museum Tusculanums Forlag, Copenhagen.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Tobias Horsmann and Torsten Zesch. 2016. Assigning fine-grained PoS tags based on high-precision coarse-grained tagging. In *Proceedings of COLING 2016, 26th International Conference on Computational Linguistics*. Osaka, Japan. http://aclweb.org/anthology/C/C16/C16-1032.pdf.

Tobias Horsmann and Torsten Zesch. 2017. Do LSTMs really work so well for PoS tagging? – a replication study. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark. https://doi.org/10.18653/v1/D17-1076.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal. https://doi.org/10.18653/v1/D15-1176.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics* 4:521–535. http://aclweb.org/anthology/Q16-1037.

Hrafn Loftsson. 2008. Tagging Icelandic text: A linguistic rule-based approach. *Nordic Journal of Linguistics* 31(1):47–72. https://doi.org/10.1017/S0332586508001820.

Hrafn Loftsson, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. 2011. Using a Morphological Database to Increase the Accuracy in POS Tagging. In *Recent Advances in Natural Language Processing*. Hissar, Bulgaria, RANLP 2011. http://www.aclweb.org/anthology/R11-1007.

Hrafn Loftsson, Ida Kramarczyk, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. 2009. Improving the PoS tagging accuracy of Icelandic text. In *Proceedings of the 17th Nordic Conference on Computational Linguistics*. Odense, Denmark, NODALIDA 2009. https://dspace.ut.ee/handle/10062/9736.

Hrafn Loftsson and Robert Östling. 2013. Tagging a Morphologically Complex Language Using an Averaged Perceptron Tagger: The Case of Icelandic. In *Proceedings of the 19th Nordic Conference on Computational Linguistics*. Oslo, Norway, NODALIDA 2013. https://aclanthology.info/papers/W13-5613/w13-5613.

Hrafn Loftsson, Jökull H. Yngvason, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. 2010. Developing a PoS-tagged corpus using existing tools. In Francis M. Tyers Sarasola, Kepa and Mikel L. Forcada, editors, *Proceedings of 7th SaLTMiL Workshop on Creation and Use of Basic Lexical Resources for Less-Resourced Languages*. Valetta, Malta, LREC 2010.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *CoRR* abs/1701.03980. http://arxiv.org/abs/1701.03980.

Jörgen Pind, Friðrik Magnússon, and Stefán Briem. 1991. *Íslensk orðtíðnibók [The Icelandic Frequency Dictionary]*. The Institute of Lexicography, University of Iceland, Reykjavik, Iceland.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany. https://doi.org/10.18653/v1/P16-2067.

Benoît Sagot and Héctor Martínez Alonso. 2017. Improving neural tagging with lexical information. In *Proceedings of the 15th International Conference on Parsing Technologies*. Pisa, Italy. http://aclweb.org/anthology/W17-6304.

Steinþór Steingrímsson, Sigrún Helgadóttir, and Eiríkur Rögnvaldsson. 2015. Analysing Inconsistencies and Errors in PoS Tagging in two Icelandic Gold Standards. In *Proceedings of the* 20^{th} *Nordic Conference of Computational Linguistics*. Vilnius, Lithuania, NODALIDA 2015. https://www.aclweb.org/anthology/W15-1838.

Steinþór Steingrímsson, Sigrún Helgadóttir, Eiríkur Rögnvaldsson, Starkaður Barkarson, and Jón Guðnason. 2018. Risamálheild: A Very Large Icelandic Text Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*. Miyazaki, Japan, LREC 2018. http://www.lrec-conf.org/proceedings/lrec2018/pdf/746.pdf.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. *CoRR* abs/1510.06168. https://arxiv.org/pdf/1510.06168.pdf.

Comparison of Machine Learning Approaches for Industry Classification Based on Textual Descriptions of Companies

Andrey Tagarev[1] and **Nikola Tulechki**[1] and **Svetla Boytcheva**[1,2]
[1]Sirma AI trading as Ontotext, Bulgara
[2]Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences
{andrey.tagarev,nikola.tulechki,svetla.boytcheva}@ontotext.com

Abstract

This paper addresses the task of categorizing companies within industry classification schemes. The dataset consists of encyclopedic articles about companies and their economic activities. The target classification schema is build by mapping linked open data in a semi-supervised manner. Target classes are built bottom-up from DBpedia. We apply several state of the art text classification techniques, based both on deep learning and classical vector-space models.

1 Motivation

The era of big data has made the task of integrating several heterogeneous sources quite common and important. This is a challenging task because, typically, data representation of the same object can significantly differ in different sources, both in level of detail and type of available information. In addition, different ontologies (classification schema) are used for the same concepts in different datasets. Ontology mapping is itself a challenge even for human experts. Thus the problem of concept classification is very important in data integration.

This paper presents a comparison of different techniques for solving the task of company industry classification based on textual descriptions of companies in DBpedia[1]. The problem of text-classification is defined as follows: for a collection of company descriptions $D = \{d_1, d_2, ..., d_n\}$, assign one or more industry categories to each company from a discrete set of labels $C = \{c_1, c_2, ..., c_k\}$.

There are more than fifteen different company industry classifications of varying granularity. Some of the most popular are: Industry Classification Benchmark (ICB)[2], Global Industry Classification Standard (GICS)[3], Thomson Reuters Business Classification (TRBC)[4], and International Standard Industrial Classification of All Economic Activities (ISIC)[5].

2 Text Classification Methods

Text classification methods are widely used in several applications like e-mail spam filtering (Youn and McLeod, 2007), news categorization (Lin and Hauptmann, 2002), in marketing for product reviews (Dang et al., 2009), etc.

The classical methods (Aggarwal and Zhai, 2012) in text classification are based on standard techniques. Usually the texts from the training corpus are transformed into vectors, where distinct words represent features. One of the simplest and most efficient methods is the probabilistic classifier Naïve Bayes (NB) that has really good performance even with few examples and sparse features and works with a ”naïve” assumption for attributes that are conditionally independent. However, usually not all words in a text are independent. Many techniques have been developed (Al-Aidaroos et al., 2010) to overcome the problems caused by the existence of attribute correlations that can lead to classification bias and negatively affect an NB classifier's performance. McCallum and Nigam (McCallum et al., 1998) demonstrate NB classifier applied on ”Industry Sector” data, that contains 6,440 company web pages classified in a hierarchy of 71 industry sectors, with a vo-

[1] https://wiki.dbpedia.org

[2] https://www.ftserussell.com/data/industry-classification-benchmark-icb

[3] https://www.msci.com/gics

[4] https://www.refinitiv.com/en/financial-data/indices/trbc-business-classification

[5] https://unstats.un.org/unsd/publication/seriesM/seriesm_4rev4e.pdf

Proceedings of Recent Advances in Natural Language Processing, pages 1169–1175,
Varna, Bulgaria, Sep 2–4 2019.

cabulary of size 29,964. The reported results for multinomial NB classifier for 20,000 words reach accuracy up to 0.74, and for 1,000 words multivariate Bernoulli model reaches accuracy up to 0.46. Frank and Bouckaert (Frank and Bouckaert, 2006) propose a solution based on multinomial NB that deals with the problem of unbalanced class sizes in Industry Sector dataset with 105 classes, where the largest category has 102 documents, the smallest has 27. They show how by using a centroid classifier and taking into account the significance of different industry sectors classes this method achieves significant gain in some categories. Maximum Entropy classifier used on Industry sector dataset (Nigam et al., 1999) shows better performance than NB and reaches a higher accuracy of 0.788. Ghani (Ghani, 2000) uses an error-correcting codes method and achieves an accuracy up to 0.886 for Industry Sector dataset with a vocabulary size of 10,000.

Support Vector Machines (SVM) (Joachims, 1998) are linear classifiers that, like NB, do not require large training datasets but need more computational time. The most important advantage of SVMs is that they have good performance even for a high dimensional space, such as text classification, where dimensions can be well over 10,000. Rennie and Rifkin (Rennie and Rifkin, 2001) propose application of SVM using one-vs-all and error-correcting output coding for Industry Sector dataset and the results show that this method significantly outperforms NB.

Logistic regression (Genkin et al., 2007) is quite efficient in cases where the dimensions of the feature space surpass the total number of training examples, which is typically the case for text classification datasets.

k-Nearest Neighbor (kNN) Classification method is a proximity-based classifiers that uses distance-based measures for classification task. Usually, such methods use all features in the vector space model which can cause lack of efficiency as not all of them are useful. Several methods for features selection are used, mainly based on the weight of the words in the text. To overcome this problem some modifications of kNN classifier for text documents were proposed, like Weight Adjusted k-Nearest Neighbor Classification (Han et al., 2001). Trstenjak et al.. 2014 present a method for text classification based on kNN and Term frequency inverse document frequency (TF-IDF). Tan (Tan, 2006) demonstrates the performance of kNN and DragPushing strategy based KNN classifier (DPSKNN) methods over subset of 48 sectors of Industry sector dataset (Sector-48 dataset). The Micro-F1 of kNN classifier for Sector-48 dataset is 0.8188 and its Macro-F1 is 0.8235. While DPSKNN shows slightly better performance with for Sector-48 dataset with Micro-F1 0.8544 and Macro-F1 0.8585. In order to emphasize the performance of the methods on common and rare classes, special averages of F1 scores over different classes are used- Micro-F1 - F1 over categories and documents; Macro-F1 - average of within-category F1 values.

Other classification techniques, like Random Forests (Xu et al., 2012), Decision Tree Classifiers (Harrag et al., 2009), Rule–based classifiers, and Conditional Random Fields (CRF) (Lafferty et al., 2001) are also used for this task.

Recent advances in Deep Learning and Transfer Learning introduce more complex methods (Kowsari et al., 2019) with significantly better performance on solving the multi-class multi-label task for industry sectors.

One of the breakthroughs in the area was done by word2vec (Mikolov et al., 2013) that proposes an efficient method for continuous semantics vector representation of words (continuous bag-of-words (CBOW) and skip-grams), by learning from a huge dataset with billions of words. Although its primary purpose is not directly related to text classification, word2vec was used as stepping stone for many other algorithms, because pre-trained word representations are widely used in deep contextual models for word embeddings.

Some more advanced models, like Glove (Pennington et al., 2014) for word representations were proposed, based on so called Global vectors - a new global log-bilinear regression model, where the learning is based on non-zero elements in word-word co-occurrence matrix.

The ELMo (Embeddings from Language Models) (Peters et al., 2018) representation uses deep bidirectional language model (biLM), where each token is assigned a representation which is a function for the whole input sentence.

Skip-thought unsupervised learning model (Kiros et al., 2015) is a generic, distributed sentence encoder that uses robust sentences representation in skip-thought vectors. It uses the idea of continuation of the information in the text and for an encoded sentence, it tries to reconstruct its surrounding sentences.

Because our dataset is based on DBpedia, the performance of neural networks (NN) algorithms over it for text-based classification task is of primary interest for us.

One of the latest algorithms XLNet (Yang et al., 2019) demonstrates the best performance for DBpedia with error 0.62. Where ”classification error” is defined as 1.0 minus classification accuracy. XLNet incorporates ideas from Transformer-XL (Dai et al., 2019). The main advantage of Transformer-XL(Dai et al., 2019) is that it allows the capture of longer-term dependencies and resolves context fragmentation problem.

Other methods with comparable results are Universal Language Model Fine-tuning (ULMFiT), (Howard and Ruder, 2018) with error 0.8 for DBpedia dataset.

Although deep learning algorithms show significant improvement in accuracy they require huge amount of labeled training examples and are computationally expensive in comparison to classical algorithms which do not need such large training datasets.

Some semi-supervised methods (Johnson and Zhang, 2016), (Sachan et al., 2019) and unsupervised methods (Xie et al., 2019) have been proposed for use in combination with supervised models to improve their performance. For example, combination $BERT_Large$+UDA of Unsupervised Data Augmentation (UDA) (Xie et al., 2019) and BERT (Devlin et al., 2018) demonstrate error 1.09 for text classification for DBpedia. The state-of-the-art (SOTA) $BERT_Large$'s error for the DBpedia is 0.64.

In this study we will compare the performance of some deep learning and transfer learning algorithms over DBpedia companies descriptions. We will present experiments with ULMfit and Glove to a baselines of one-hot unigram and one-hot bigram models. These methods were chosen because they are not so computationally expensive in comparison with the others and will serve as a baseline to exploit the potential of the deep learning approach for text-based classification of industry sectors.

3 Dataset

The dataset[6] we used for the experiments is encyclopedic data, consisting of approximately 300,000 textual descriptions of organizations and a classification based on DBPedia classes, which itself is based on Wikipedia. The descriptions are simply the English language abstracts[7] of the the Wikipedia articles about the organizations, and are thus relatively homogeneous in size and style. The *Industries* classification is based on the nature of organization's activity and is generated from the *industry*[8] property of DBPedia. The over 17,500[9] distinct ”industries” are normalized *via* a custom mapping we have developed in an iterative manner guided by the taxonomy's commercial applicability. The end result is a multi–level hierarchy but the experiments described in this paper concern only the 32 top-level classes. For example, the organization ”Bulgaria Air”[10] is, according to DBPedia, an *Airline*[11], which according to our mapping is a sub-industry of *Air Transport*[12], itself a sub-industry of *Transport*[13], the top-level industry in our mapping. Note that, some organizations, such for example, the ”East Japan Railway Company”[14] are directly classified the top-level industry. As is visible in Table 1, the largest classes have well over ten thousand positive examples while many of the smallest have much fewer than a thousand.

4 Experiments and Results

To compare the performance of the methods discussed, we trained a series of algorithms on the same split of our data to get comparable results. We randomly split the training data into three roughly equal parts that were used to carry out three-fold cross validation. In each fold, 60% of the data was used as training data, 7% as valida-

[6] https://gitlab.ontotext.com/trainings/global_datathon/blob/master/data/dt18-ontotext-simple.csv.zip

[7] In Wikipedia the abstracts are the short descriptions between the title and the table of contents of the article

[8] http://dbpedia.org/ontology/industry

[9] DBpedia is rather noisy, over 12,000 of these values are *hapaxes*

[10] http://dbpedia.org/resource/Bulgaria_Air

[11] http://dbpedia.org/resource/Airline

[12] http://dbpedia.org/resource/Air_Transport

[13] http://dbpedia.org/resource/Transport

[14] http://dbpedia.org/resource/East_Japan_Railway_Company

tion data to determine when to stop training and 33% was used as the test data shown to the algorithm only after training is complete. The results reported here are the cumulative performance of the algorithms trained on each fold so each entry in the dataset has been seen as a testing example by one fold one time.

4.1 Linear Baseline

Serving as a baseline, we have a straightforward linear model- a perceptron with no hidden layers. The company descriptions are first processed through a standard NLP pipeline for stopword removal and stemming and then each unigram is converted into a one-hot vector representation[15]. The input for the perceptron is the sum of the unigram vectors.

4.2 Customized Linear Model

The second approach is a customized linear model. It utilizes the same NLP preprocessing of the text and feeds into the same linear perceptron but the features include unigrams and bigrams. Each feature is still represented as a one hot vector as in the baseline model.

4.3 GloVe

The third approach serves as a baseline attempt for incorporating context vectors. It uses the same preprocessing steps as the linear baseline approach (i.e. NLP pipeline for stopword removal and stemming, resulting text is processed into unigrams) but instead of one–hot vectors, GloVe vector embeddings are used. Specifically the 300-dimensional GloVe vectors trained on the large Common Crawl corpus of 840 billion tokens with a vocabulary of 2.2 million words. While there is no additional training of the GloVe embeddings for our specific data, the corpus used to extract context is many orders of magnitude greater than our text data.

The resulting vectors are once again fed into a linear perceptron but because the 300-dimensional resulting vector is much smaller than the one-hot representation in the first two experiments, this approach had much lower training times than the other examined alternatives. Training times were generally under a minute instead of 15-60 minutes.

4.4 ULMfit

Finally, we tested one of the state of the art algorithms for classification with context vectors- ULMfit(Howard and Ruder, 2018) by fast.ai[16]. This is the most sophisticated of the four approaches and the one that varies the most from the initial three.

The first major distinguishing aspect of this approach is the text preprocessing. Rather than the traditional NLP stemming pipeline feeding into one-hot vectors, we use all available company descriptions in order to train a fully custom Language Model based on AWD-LSTM which produces our context vectors directly. For the purposes of these experiments, we used the default settings for the network but there is significant opportunity for in-depth exploration of the language model's performance with various configurations.

The classification training step is implemented as an additional layer added onto an already trained language model. The effect allows relative quick initial training of the context vectors followed by some fine-tuning of the context vectors along the specific classification layer training.

5 Discussion

To compare the performance of the four experiments, let's first look at the class-by-class breakdown of their performance as shown in Table 1. There we can see the F1-score achieved by each algorithm on each of the 32 industries. The second column of the table shows the number of companies of that class and we can see there is a very big discrepancy- from over 76 thousand for the largest class to barely 300 for the smallest class. We can similarly observe that the algorithms achieve good results on the larger classes but their performance degrades and becomes increasingly erratic on the smaller classes.

Looking through the data we can make several other observations. Each algorithm has at least a few classes where they get very poor results and several classes where they achieve the highest results. The difference in results is generally inversely proportional to the size of the class although there are some notable exceptions e.g. the bigram linear model significantly underperforms on the 3rd and 4th largest classes. Overall it is not possible to identify a clearly superior algorithm

[15] binary vectors that are all zero values except for the index corresponding to the word or ngram

[16] https://github.com/jannenev/ulmfit-language-model

Industry	**Size**	**hot-unigram**	**hot-bigram**	**GloVe**	**ULMfit**
Entertainment_and_publishing	76309	0.98	0.98	0.96	0.98
Education	55221	0.98	0.98	0.97	0.99
Travel_and_sport	44768	0.99	0.68	0.90	0.98
Public_sector	26391	0.97	0.63	0.98	0.97
Information_technology	10255	0.82	0.97	0.67	0.80
Transport	10007	0.86	0.99	0.79	0.93
Manufacturing	7757	0.93	0.93	0.72	0.66
Financial_services	6086	0.79	0.87	0.61	0.86
Retail	4464	0.72	0.84	0.66	0.67
Food_and_Beverage	3748	0.64	0.83	0.59	0.83
Nonprofit_organization	3655	0.67	0.69	0.68	0.83
Personal_and_household_goods	3206	0.70	0.76	0.96	0.60
Automotive	2564	0.77	0.78	0.46	0.76
Telecommunications	2500	0.89	0.67	0.73	0.74
Aerospace_and_defense	2425	0.69	0.62	0.76	0.63
Engineering	1758	0.30	0.80	0.84	0.27
Utility	1599	0.46	0.52	0.86	0.68
Commercial_and_professional_services	1268	0.61	0.49	0.53	0.07
Fossil_fuel	1213	0.74	0.54	0.74	0.75
Cultural_heritage	1139	0.69	0.76	0.48	0.90
Pharmaceuticals_and_life_sciences	1062	0.81	0.71	0.72	0.76
Real_estate	920	0.52	0.46	0.41	0.64
Healthcare	915	0.48	0.75	0.57	0.51
Marketing	902	0.44	0.53	0.54	0.36
Conglomerate_(company)	780	0.73	0.74	0.81	0.06
Construction_and_materials	764	0.42	0.72	0.50	0.28
Mining	665	0.72	0.91	0.66	0.69
Justice_and_law	577	0.54	0.90	0.72	0.91
Chemical_industry	526	0.56	0.47	0.48	0.15
Agriculture	359	0.32	0.53	0.39	0.19
Forest_and_paper	328	0.88	0.36	0.39	0.22
Metal	302	0.45	0.37	0.88	0.25

Table 1: Class-by-class comparison of F1 scores between the four algorithms

	hot-unigram			**hot-bigram**			**GloVe**			**ULMfit**		
	Prec	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*	*Prec*	*Rec*	*F1*
Micro	0.943	0.901	0.922	0.944	0.909	0.926	0.913	0.899	0.906	0.956	0.888	0.921
Macro	0.788	0.625	0.689	0.786	0.658	0.712	0.700	0.674	0.686	0.861	0.572	0.641

Table 2: Comparison of overall performance between the four algorithms

by looking at the individual classes although it is worth noting that the bigram linear model performance degraded on some of the largest classes.

If we turn our attention to Table 2, we can examine a micro and macro view of the algorithm performance. The micro-average is obtained by summing up each individual decision of the algorithm while the macro-average is obtained by averaging the scores for each class. The first observation here is that the macro-average recall of the ULMfit is particularly low which makes its macro-average F1 score similarly lower than the others. However, because of the huge class size imbalance demonstrated in Table 1, the macro-average is a poor metric for our particular problem. The conclusion we can draw from this is that ULMfit has achieved some abysmal recall on the smallest classes; a likely cause for this is the lack of stemming in the language model used.

Looking to the micro-averages, the algorithms have achieved much closer performance. The GloVe linear approach is the only one falling significantly behind in F1-score while the hot-bigram model narrowly achieves the best F1-score. To that end there is a clear trade-off, however, with the ULMfit approach having the higher precision while the ngram linear models achieve better recall. As already mentioned, this better recall is likely caused by the stemming in the NLP pipeline which the ULMfit context vectors cannot overcome with the limited size of the corpus.

6 Conclusion and Further Work

The analysis of the results shows that all of the tested approaches produce relatively close results with none emerging as clearly superior to all the others. Overall we observed that ULMfit achieves higher precision while one-hot vector linear models achieve better recall. The linear GloVe vector algorithm achieved inferior results to the other three options overall.

The analysis also shows that the different models have surprisingly varying behavior on the smaller classes indicating that there is still room for improvement in the scores achieved for those smaller classes. ULMfit, while giving comparable results to the linear approaches, presents the best opportunity for that improvement.

There are a few viable directions for exploring further improvement in the performance of the algorithm. One approach would be to work towards improving the reliability of the language model by testing the effect of various parameters or beginning with already trained embedding vectors that are only fine-tuned on our corpus. An alternative direction of experimentation would be to look at the data itself- we know it isn't a true gold standard and expect a relatively large rate of error so it is possible that many of the algorithm "mistakes" are actually errors in the underlying data rather than of the algorithms themselves.

Acknowledgements

The research presented in this paper is partially funded by project The Intelligent Matching and Linking of Company Data project (CIMA), grant BG16RFOP002-1.005-0168-C01 by the European Unions European Regional Development Fund through Operational Programme Innovations and Competitiveness 2014-2020 call Intelligent Specialization. We are grateful to anonymous reviewers for useful comments and suggestions.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, Springer, pages 163–222.

Khadija Mohammad Al-Aidaroos, Azuraliza Abu Bakar, and Zalinda Othman. 2010. Naive bayes variants in classification learning. In *2010 International Conference on Information Retrieval & Knowledge Management (CAMP)*. IEEE, pages 276–281.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* .

Yan Dang, Yulei Zhang, and Hsinchun Chen. 2009. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems* 25(4):46–53.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Eibe Frank and Remco R Bouckaert. 2006. Naive bayes for text classification with unbalanced classes. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pages 503–510.

Alexander Genkin, David D Lewis, and David Madigan. 2007. Large–scale bayesian logistic regression for text categorization. *Technometrics* 49(3):291–304.

Rayid Ghani. 2000. Using error-correcting codes for text classification. In *ICML*. pages 303–310.

Eui-Hong Sam Han, George Karypis, and Vipin Kumar. 2001. Text categorization using weight adjusted k-nearest neighbor classification. In *Pacific-asia conference on knowledge discovery and data mining*. Springer, pages 53–65.

Fouzi Harrag, Eyas El-Qawasmeh, and Pit Pichappan. 2009. Improving arabic text categorization using decision trees. In *2009 First International Conference on Networked Digital Technologies*. IEEE, pages 110–115.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* .

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, pages 137–142.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. pages 3294–3302.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. 2019. Text classification algorithms: A survey. *Information* 10(4):150.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., pages 282–289.

Wei-Hao Lin and Alexander Hauptmann. 2002. News video classification using svm-based multimodal classifiers and combination strategies. In *Proceedings of the tenth ACM international conference on Multimedia*. ACM, pages 323–326.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer, volume 752, pages 41–48.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*. volume 1 (1), pages 61–67.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* .

Jason DM Rennie and Ryan Rifkin. 2001. Improving multiclass text classification with the support vector machine. In *Series/Report no. AIM-2001-026CBCL-210*. http://hdl.handle.net/1721.1/7241.

Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. 2019. Revisiting lstm networks for semi-supervised text classification via mixed objective function. In *AAAI 2019*.

Songbo Tan. 2006. An effective refinement strategy for knn text classifier. *Expert Systems with Applications* 30(2):290–298.

Bruno Trstenjak, Sasa Mikac, and Dzenana Donko. 2014. Knn with tf-idf based framework for text categorization. *Procedia Engineering* 69:1356–1364.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848* .

Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. 2012. An improved random forest classifier for text categorization. *JCP* 7(12):2913–2920.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237* .

Seongwook Youn and Dennis McLeod. 2007. A comparative study for email classification. In *Advances and innovations in systems, computing sciences and software engineering*, Springer, pages 387–391.

A Quantum-Like Approach to Word Sense Disambiguation

Fabio Tamburini
FICLIT - University of Bologna, Italy
fabio.tamburini@unibo.it

Abstract

This paper presents a novel algorithm for Word Sense Disambiguation (WSD) based on Quantum Probability Theory. The Quantum WSD algorithm requires concepts representations as vectors in the complex domain and thus we have developed a technique for computing complex word and sentence embeddings based on the Paragraph Vectors algorithm. Despite the proposed method is quite simple and that it does not require long training phases, when it is evaluated on a standardized benchmark for this task it exhibits state-of-the-art (SOTA) performances.

1 Introduction

The introduction of the *prototype theory* by E. Rosch (1973), one of the most influential theories describing concept organisation at cognitive level, completely changed the perspective in semantics and nowadays most of the studies in computational semantics consider concepts membership and concepts similarity as graded features in a "semantic space".

In Natural Language Processing (NLP) all the recent and fundamental studies on word and sentence embeddings, e.g. (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016; Le and Mikolov, 2014; Sutskever et al., 2014; Kiros et al., 2015; Subramanian et al., 2018; Cer et al., 2018), as well as the older works on word spaces based on co-occurence measures, e.g. see the reviews from (Turney and Pantel, 2010; Baroni and Lenci, 2010), rely on an high-dimensional vector space to represent concepts, through the application of the Harrisian distributional hypothesis, collecting contextual information from text corpora, and measuring their relationships by means of some kind of geometric distance.

The fundamental experiments of Tversky (1977) in cognitive psychology on concept similarity and concept combination challenged the common view to adopt the Classical, Kolmorogovian, Probability Theory (CPT) to explain and model these phenomena. Starting from Tversky's data, a large set of newer experimental studies in cognitive psychology showed a systematic violation of the basic axioms of CPT. For example, the classical problem known as the "pet-fish problem" or "guppy-effect" (Osherson and Smith, 1981) is a typical case of overextension in concept conjunction: if we denote as *pet-fish* the conjunction of the concepts *pet* and *fish* and ask people to rate "guppy" as member of *pet*, *fish* and *pet-fish*, they tend to consider it as a highly typical *pet-fish* but neither a particularly typical *pet* nor *fish*. This violates the CPT axiom stating that the probability of the events conjunction must be less or equal to the probability of the single events. In a similar way scholars in cognitive psychology proposed various experiments showing a systematic violations of CPT axioms in concept disjunction, conceptual negation, decision making and some other relevant cognitive processes: see, for example, (Hampton, 1988; Alxatib and Pelletier, 2011; Aerts et al., 2015). These studies show clearly the impossibility of modeling such cognitive phenomena by using CPT and even by using further elaboration of it such as the fuzzy-set probability theory.

In the same field a growing set of studies in the last decades started to explore the possibility of modeling such cognitive phenomena by using different, more sophisticated, probability theories, in particular Quantum Probability Theory (QPT), the foundational calculus of Quantum Mechanics Theory (QMT). We refer the reader to these books or comprehensive reviews for an in-depth introduction to these approaches in cog-

Proceedings of Recent Advances in Natural Language Processing, pages 1176–1185,
Varna, Bulgaria, Sep 2–4 2019.

nitive psychology (Busemeyer, 2012; Haven and Khrennikov, 2013; Pothos and Busemeyer, 2013; Yearsley et al., 2015; Wendt, 2015; Ashtiani and Azgomi, 2015; Aerts et al., 2016a,b; Haven and Khrennikov, 2018). A very large set of these works showed that, by applying the axioms of QPT and taking advantage from the peculiar phenomena modeled by this calculus, such as *superposition*, *interference* and *entanglement*, it is possible to build complete models which are able to well explain all the experimental data and incorporate in the theory all the deviations from CPT exhibited by human behaviours during cognitive experiments. Even if some studies try to find connections by QPT and brain functionality at neural level (Khrennikov et al., 2018), the use of QPT in this field is simply as an explanation theory useful to model real phenomena in the right way, but none of them is really claiming that our brain is working by applying QPT axioms. That is why it is common to use the term “quantum-like” to describe models making use of this calculus in cognitive psychology.

Even if QMT is one of the most successful theories in modern science, the attempts to apply it in other domains remain rather limited, excluding, of course, the large quantity of studies regarding Quantum Information Processing on quantum computers and Electronics. Only in recent years some scholars tried to embody principles derived from QMT into their specific fields, for example, by the Information Retrieval community (van Rijsbergen, 2004; Zuccon et al., 2009; Melucci and van Rijsbergen, 2011; González and Caicedo, 2011; Melucci, 2015), by the Economics and Finance community (Baaquie, 2018) and by the community studying Quantum Computation and Information Theory (Nielsen and Chuang, 2010; Wilde, 2013). In the machine learning field (Arjovsky et al., 2016; Wisdom et al., 2016; Jing et al., 2017) have used unitary evolution matrices to build deep neural networks obtaining interesting results, but we can observe that their works do not completely adhere to QPT and use unitary evolution operators in a way not allowed by QPT. (Moreira and Wichert, 2018) presented an interesting application of QPT for developing Quantum Bayesian Networks. In recent years, also the NLP community started to look at QPT with interest and some studies using it have already been presented (Blacoe et al., 2013; Liu et al., 2013; Kartsaklis et al., 2016; Basile and Tamburini, 2017).

Given this general framework, and the successful results in explaining cognitive experiments, it seemed natural trying to explore the possibility of applying QPT also in NLP and see if the peculiar properties of this calculus could introduce some benefits for solving NLP tasks.

In this paper we will apply QPT for modeling the Word Sense Disambiguation problem testing our proposal on well-known benchmarks. WSD is an historical task which aims to assign the correct word sense for a polysemous word given a linguistic context. The possible senses for a given word are extracted from a reference sense inventory. We refer the reader to the reviews of (Agirre and Edmonds, 2007; Navigli, 2009; Vidhu Bhala and Abirami, 2014) and to the papers describing the last evaluation results (Navigli et al., 2013; Moro and Navigli, 2015; Raganato et al., 2017a) to get a clear picture of the SOTA for this task.

Computational systems solving such task can be broadly divided into two main groups: knowledge-based systems do not require a sense-annotated corpus for training the model and are usually based on lexical or knowledge resources for performing the disambiguation process; on the contrary, supervised WSD systems require a sense annotated corpus in order to train the model and set up all the parameters. Looking at the previously cited evaluations, supervised WSD systems are able to produce the best results and are currently establishing the SOTA: see, for example, (Zhong and Ng, 2010; Iacobacci et al., 2016; Papandrea et al., 2017; Raganato et al., 2017b; Tripodi and Pelillo, 2017; Luo et al., 2018b,a; Melacci et al., 2018; Uslu et al., 2018). Despite these long time studies, the Most-Frequent-Sense baseline is still a strong algorithm challenging all new proposals, and the best systems results are only few points over that baseline.

2 Quantum Probability Theory

This section aims to introduce the basic background knowledge necessary to understand QPT and the underlying mathematical constructions. A more complete introduction about these topics can be found, for example, in (Nielsen and Chuang, 2010; Busemeyer, 2012). It is important to note that QPT is a probability theory more general than CPT and it includes it completely.

Quantum Events

QPT assigns probability to events as well as classical Kolmogorovian probability theory, but, unlike CPT that defines events as sets, it defines events as subspaces of a multidimensional complex Hilbert space $\mathcal{H} = \mathbb{C}^n$.

Quantum States

In QPT the state of a quantum system is defined, using the Dirac notation[1], as a complex vector $|\psi\rangle \in \mathcal{H}$ with $|\, |\psi\rangle \,| = 1$ and, in its general formulation, it can be expressed as

$$|\psi\rangle = \phi_1 |e_1\rangle + \phi_2 |e_2\rangle + ... + \phi_n |e_n\rangle \qquad (1)$$

where ϕ_j are complex numbers called *probability amplitudes*, $\phi_j = \langle e_j|\psi\rangle$, and $\{|e_j\rangle\}$ is a basis of the Hilbert space $\mathcal{H}$. The state in (1) is called a *superposition* state w.r.t. the basis vectors seen as basic states.

For each event subspace spanned by the vector $|x\rangle$, it is possible to build a projector operator $P_x = |x\rangle\langle x|$ that can project a generic state vector $|\psi\rangle$ onto the subspace corresponding to that event.

Quantum Measurements

In QPT, quantum measurements of a variable (or observable) M are usually represented by a set of measurement operators $\{M_k\}$ where the index indicates one of the possible measurement outcomes and $\sum_k M_k^\dagger M_k = I$ (I denotes the $n \times n$ identity matrix). Applying a measurement on a quantum system when in state $|\psi\rangle$ we can compute the probability of getting a specific result k as

$$P(k) = \langle\psi|M_k^\dagger M_k|\psi\rangle\,. \qquad (2)$$

When we measure a quantum system and an event is observed, the act of measuring it changes the state of the system from the superposed state $|\psi\rangle$ to a new state, it is said that the system *collapses*; this new state is given by

$$|\psi\rangle' = \frac{M_k\,|\psi\rangle}{\sqrt{\langle\psi|M_k^\dagger M_k|\psi\rangle}}. \qquad (3)$$

An important class of measurements is known as projective measurements. These measurements are represented by Hermitian observables that admit a spectral decomposition $M = \sum_k v_k P_k$ where $P_k = |u_k\rangle\langle u_k|$ is the projector onto the eigenvector $|u_k\rangle$ of M with eigenvalue v_k and we can compute the probability of obtaining the result k as $P(k) = \langle\psi|P_k|\psi\rangle = |\,\langle u_k|\psi\rangle\,|^2$. The eigensystem obtained by the spectral decomposition imply the assumption of orthogonality between the eigenvectors and thus force measurements on this orthonormal basis of $\mathcal{H}$. Once applied a measurement the system will collapse and no more uncertainty will remain. A further measurement of the outcome k will result in $P(k) = 1$.

There is also another type of measurements, the Positive Operator-Valued Measurement (POVM). Projective measurements require the assumption of orthogonality and are not well suited to measure non-orthonormal states and compute their probabilities. A POVM is a set of Hermitian positive operators $\{E_i\}$ such that $\sum_i E_i = \sum_i M_i^\dagger M_i = I$ is the only requirement. Note that for projective nonorthogonal operators all M_i can be written as outer products of general, non orthonormal, state vectors and we can introduce any number of operators E_i.

Quantum Interference

Interference is one of the most intriguing phenomena arising only in the domain of quantum systems. The classical double slit experiment is often used as a simple example to introduce this phenomenon, see, for example, (Zuccon et al., 2009). Let us shoot a physical particle towards a screen with two slits A and B and, once passed the screen, the particle hits a detector panel behind the screen in a specific position x. By closing one of the two slits, say B, we can compute the probability that the particle hits the detector at a position x passing through A, $P_A(x) = |\phi_A(x)|^2$, or the reverse, by closing A, $P_B(x) = |\phi_B(x)|^2$ where $\phi_A(x)$ and $\phi_B(x)$ are the probability amplitudes associated with the two events $|e_A\rangle$ and $|e_B\rangle$ forming an orthonormal basis for $\mathcal{H} = \mathbb{C}^2$. By applying the classical probability we can compute $P_{AB}(x)$ when both slits are open and the particle can pass either through A or B as

$$P_{AB}(x) = P_A(x) + P_B(x) = |\phi_A(x)|^2 + |\phi_B(x)|^2 \qquad (4)$$

but experimentally we can note that this equality does not hold and that we have to correct equation (4) by applying the QPT adding an *interference*

[1] In Dirac notation $|.\rangle$ is a column vector, or a *ket*, while $\langle.|$ is a row vector, or a *bra*. Using this notation the inner product between two vectors can be expressed as $\langle x|y\rangle$ and the outer product as $|x\rangle\langle y|$. Then $\langle x| = |x\rangle^\dagger$, where † marks the conjugate transpose operation on vectors or matrices.

term:

$$\begin{aligned} P_{AB}(x) &= |\phi_A(x)|^2 + |\phi_B(x)|^2 + I_{AB}(x) \\ &= |\phi_A(x)|^2 + |\phi_B(x)|^2 + \\ &\quad (\phi_A(x)^* \phi_B(x) + \phi_A(x) \phi_B(x)^*) \end{aligned}$$

In summary, the classical Kolmogorovian rule for addition of probabilities when the event can occur in various alternative ways is violated and we have to apply the QPT in order to completely explain the experiments results.

3 Quantum WSD

3.1 Background

The literature on cognitive psychology gives us precious suggestions about the definitions of the elements involved in our problem and on how operationalise them in the framework of QPT.

Concepts can be seen as quantum states described by state vectors, $|\psi\rangle$, in a complex Hilbert space $\mathcal{H}$.

Specific entities or exemplars or, more appropriately in the WSD domain, polysemous words are viewed as superposed states between the referring senses, or concepts, state vectors. For example the vector for a polysemous word $|W\rangle$ can be expressed as

$$|W\rangle = \phi_1 |S_1\rangle + ... + \phi_m |S_m\rangle \qquad (5)$$

where $\{|S_j\rangle\}$ represents the set of all its possible sense vectors.

A context, as a piece of text in a natural language (e.g. a sentence), provides a specific meaning to a polysemous word collapsing its superposition to one of its possible senses. It is described as a measurement operation projecting the system state into a specific subspace spanned by the linguistic context.

In order to transform these general intuitions into a practical system, the crucial step regards the possibility of generating vector representations of words and senses in the complex domain. The large set of works introducing word and sentence embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016; Le and Mikolov, 2014; Sutskever et al., 2014; Kiros et al., 2015; Subramanian et al., 2018; Cer et al., 2018) produce representations in the real domain while we need similar vectors but in the complex domain. The next section will show how to transform a classical word embedding approach in order to obtain complex word/sentence embeddings.

3.2 Complex Word/Sentence Embeddings

There are some recent work in literature (Trouillon et al., 2016; Li et al., 2018) proposing techniques for computing complex-valued sentence embeddings to solve a specific task by training a Deep Neural Network (DNN) on the problem output classes. Although strictly connected with our work, these studies learn complex embeddings tailored to a specific task, while we are looking for a procedure to learn general complex representations of words and sentences potentially useful for solving a wide range of tasks.

We took inspiration from the unpublished underdocumented code made available by Théo Trouillon[2] extending `word2vec` code[3] (Mikolov et al., 2013) for working on complex numbers and generating complex word embeddings.

In `word2vec` the skip-gram negative-sampling model is trained minimising the objective

$$E = -log\, \sigma(\mathbf{v'}^T_{w_O} \mathbf{v}_{w_I}) - \sum_{w_j \in \mathcal{W}_{neg}} log\, \sigma(-\mathbf{v'}^T_{w_j} \mathbf{v}_{w_I})$$

where σ is the sigmoid function, w_O is the output word (a positive sample taken from the real context), $\mathbf{v}'_{w_O}$ is its corresponding vector taken from the output weight matrix and $\mathbf{v}_{w_I}$ is the value of the hidden layer that, for skip-gram models, is equivalent to the input word (w_I) vector taken from the input weight matrix.

For extending this model to work with complex numbers we have to transform the input and output weight matrices from real to complex values and adapt the objective function consequently. Unfortunately there are no studies, up to our knowledge, handling directly complex losses and most of the more recent attempts to work with complex neural networks (Trabelsi et al., 2018; Scardapane et al., 2018; Sarroff, 2018) transform the complex loss into a real one by applying some function f to the network output. We can then adapt the objective function as following

$$\begin{aligned} E' &= -log\, \sigma\Big(f\left(\langle \mathbf{v}'_{w_O} | \mathbf{v}_{w_I} \rangle\right)\Big) - \\ &\quad \sum_{w_j \in \mathcal{W}_{neg}} log\, \sigma\Big(f\big(-\langle \mathbf{v}'_{w_j} | \mathbf{v}_{w_I} \rangle\big)\Big) \end{aligned}$$

where $\mathbf{v}'_{w_x}$ and $\mathbf{v}_{w_x}$ are now complex vectors and

[2] https://github.com/ttrouill/imwords.git

[3] https://code.google.com/archive/p/\word2vec/

$f : \mathbb{C} \rightarrow \mathbb{R}$ can be defined as

$$f(z) = \Re(z) + \Im(z) = \frac{z + \overline{z}}{2} + \frac{z - \overline{z}}{2i}.$$

where $\overline{z}$ is the complex conjugate of z.

E' is still a real-valued loss function and, by leveraging the Wirtinger calculus to extend the complex derivative (well defined only for holomorphic functions) to non-holomorphic functions and real-valued analytic functions, we can calculate and backpropagate all the gradients needed to update the network weights:

$$\frac{\partial E'}{\partial \mathbf{v}'_{w_j}} = (1+i) \left[\sigma \left(\langle \mathbf{v}'_{w_j} | \mathbf{v}_{w_I} \rangle \right) - t_j \right] \mathbf{v}_{w_I}$$

$$\frac{\partial E'}{\partial \mathbf{v}_{w_I}} = (1-i) \left[\sigma \left(\langle \mathbf{v}'_{w_j} | \mathbf{v}_{w_I} \rangle \right) - t_j \right] \mathbf{v}'_{w_j}$$

where $t_j = 1$ if $w_j = w_O$ (positive sample), $t_j = 0$ if $w_j \in \mathcal{W}_{neg}$ (negative samples) and i is the imaginary unit.

Le and Mikolov (2014) proposed an extension to the `word2vec` model to build vectors referred to a generic piece of text (phrases, sentences, paragraphs or entire texts). They called this extension "Paragraph Vectors" (PVs). Following their paper and the suggestions given by T. Mikolov for implementing PVs[4], we extended the code accordingly, producing complex paragraph vectors (cPVs) for fragments of texts longer than a word. As in the cited paper, it was sufficient to insert a fake word at the beginning of the paragraph and training it together with all the other words forming the paragraph to obtain, at the end of the training process, reliable dense vector representations for the paragraph in the complex domain as well as complex vector representations for words (cWV). Although this vectors are not derived using QPT and we used the Dirac notation only for consistency, they will enable us to use such complex vectors as the basic elements in our WSD algorithm based on QPT.

3.3 The WSD Model

The proposed model for WSD relies heavily on an external lexical resource for getting all the glosses and examples connected to a specific meaning. WordNet (Miller, 1995), BabelNet (Navigli and Ponzetto, 2012) or other lexical resources providing a large set of senses with their glosses and examples can be used for our purpose.

Given the general considerations made in Sections 3.1 and the complex vector representations introduced in 3.2, we can list the ingredients for our WSD recipe in the following way:

- the target word W to be disambiguated will be represented as the corresponding cWV, namely $|W\rangle$;
- the subspace of $\mathcal{H}$ connected with the sense S, has to be build by combining all the glosses and examples provided by the external lexical resource, seen as the corresponding cPVs, and all the disambiguated contexts extracted from the training set belonging to this specific sense, again seen as cPVs. The whole set of vectors $\{|\mathcal{G}_j\rangle\}$ belonging to a specific sense S are, in general, non-orthogonal each other and thus cannot form a proper basis to define the subspace connected with the sense. A standard procedure to obtain an orthonormal basis spanning the same subspace of a specific set of vectors is based on the Singular Value Decomposition and it is available in any linear algebra library. Given the orthonormal basis spanning the same space of $\{|\mathcal{G}_j\rangle\}$, say $\{|\mathcal{O}_i\rangle\}$, we can build the projector over the subspace spanned by $\{|\mathcal{G}_j\rangle\}$ relative to the sense S as

$$P_S = \sum_i |\mathcal{O}_i\rangle\langle\mathcal{O}_i| \qquad (6)$$

- the context subspace will be represented by all the cPVs corresponding to the sentences belonging to the context and the projector P_C to this subspace can be computed following the same procedure as in the previous point.

Having defined such ingredients, the disambiguation process consists in projecting the word state $|W\rangle$ onto the context subspace by applying the quantum measurement operation of (3)

$$|W_C\rangle = \frac{P_C\,|W\rangle}{\sqrt{\langle W|P_C^{\dagger}P_C|W\rangle}}$$

and then compute, by applying another measurement on the new state $|W_C\rangle$, which of the possible

[4] https://groups.google.com/d/msg/word2vec\-toolkit/Q49FIrNOQRo/J6KG8mUj45sJ

senses of W, $\{S_k\}$, exhibits the maximum similarity with $|W_C\rangle$ or, in other words, the projection of $|W_C\rangle$ over S_k has the maximum probability:

$$\overline{S} = \underset{S_k}{\operatorname{argmax}}\, P(S_k) = \underset{S_k}{\operatorname{argmax}}\, \langle W_C | P^{\dagger}_{S_k} P_{S_k} | W_C \rangle$$

where P_{S_k} is the projector obtained by equation (6) for sense S_k.

4 Experiments

We made two kind of experiments: the first is aimed to evaluate if the proposed procedure to learn complex word/sentence embeddings from texts produces effective results, while the second is devoted to the specific evaluation of our Quantum WSD system (QWSD). Both experiments rely on standard, largely-used evaluations benchmarks.

4.1 Complex Embedding Evaluation

Producing complex sentence embeddings for getting the best performance is not the main focus of this work. We simply need word/sentence representations in the complex domain in order to use QPT to develop our new approach to WSD. Thus, the evaluation of the cPVs is simply devoted to be certain that the cVPs are reliable dense representations of our glosses and contexts sentences.

To test the cPVs we adopted the benchmark proposed by (Conneau and Kiela, 2018) to evaluate sentence embeddings focusing on the five Semantic Textual Similarity (STS) tasks. We chose to apply only these tasks because we are not interested in a complete evaluation, but only in getting a broad idea if our cPVs were reliable enough to build our QWSD model.

Table 1 shows the evaluation results. The performances of our model in the STS tasks are in line with the other basic method for producing sentence embeddings. For a fair comparison, cPVs have the same number of parameters as the other methods, thus, considering that the experiments in (Conneau and Kiela, 2018) were made with vectors of 300 dimensions, our result is referred to complex embeddings with 150 dimensions.

4.2 QWSD Evaluation

In order to evaluate the proposed method to solve the WSD problem we relied, as most of the recent studies, on the standardized evaluation proposed by (Raganato et al., 2017a) for English all-words WSD. This benchmark is based on two corpora for training the systems, namely SemCor (Miller et al., 1994) and OMSTI (Taghipour and Ng, 2015) and five test corpora taken from Senseval/SemEval evaluation campaigns: Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007), SemEval-2013 (Navigli et al., 2013) and SemEval-2015 (Moro and Navigli, 2015).

Model			**STS**		
	’12	**’13**	**’14**	**’15**	**’16**
GloVe BoW	0.52	0.50	0.55	0.56	0.51
fastText BoW	0.58	0.58	0.65	0.68	64.3
SkipThought-LN	0.31	0.25	0.31	0.31	-
InferSent	0.59	0.59	0.70	0.71	0.72
Char-phrase	0.66	0.57	0.75	0.76	-
ELMo (Orig.5.5B)*	0.55	0.53	0.63	0.68	0.60
USE (DAN)*	0.59	0.59	0.68	0.72	0.70
USE (Transf.)*	0.61	0.64	0.71	0.74	0.74
PVs (300)	0.53	0.61	0.66	0.69	0.65
cPVs (150)	0.53	0.61	0.65	0.69	0.64

Table 1: Evaluation of sentence representations on the STS benchmarks as the average of Pearson correlations. Systems marked with * use embeddings bigger than 300 dim. Data were taken from (Conneau and Kiela, 2018) and (Perone et al., 2018). At the end the results of the cPVs and the standard PVs in the real domain.

We compare our evaluation results with all the systems already evaluated by (Raganato et al., 2017a) and with the new studies presented in the last two years (Papandrea et al., 2017; Zhong and Ng, 2010; Iacobacci et al., 2016; Raganato et al., 2017b; Luo et al., 2018b,a; Melacci et al., 2018; Uslu et al., 2018).

Most of the studies cited before required complex training phases and they use the SemEval-2007 dataset as the validation set, thus, although we did not need to use it in this way, it has to be excluded from the test sets for evaluating WSD systems. Moreover, most of the previous results were obtained by using only SemCor as training set and we stick to this practice to enable a complete comparability of the various results. The standard metric for this task is the F-score.

The setting for our experiments is very simple:

- we collected the glosses and the examples for a given sense, a Wordnet synset, from BabelNet v3.7.

- with regard to the creation of cPVs, by following the unsupervised procedure described in 3.2, we created a single corpus formed by all the sentences contained in the British

National Corpus[5] joined with the BabelNet glosses and examples for the various senses and all the training and test sets contexts to be used during the evaluation. It is important to underline that we connected the training set context with the correct target word sense, but for the test set we simply connected the contexts to their instance id without any explicit link to the correct results. In other words, the fake words we inserted for generating the cPVs are the correct WordNet sense id string for the training context and the test instance id string for the test contexts. In this way we can retrieve the cPVs when needed without compromising the evaluation. We used the hyperparameters setting proposed in the Mikolov's post cited before without any parameter optimisation (emb. size = 400, win. size = 10, neg. samples = 5, sub-sampl. = 1e-4, iter. = 20, min. word freq. = 5).

- the disambiguation procedure is deterministic and does not have any parameter to tune. We only introduced a limit to the number of senses for each target word equal to 20.

4.3 Results

Table 2 shows the results obtained by the proposed system (QWSD) compared with the results obtained by the SOTA systems on the evaluation framework proposed by (Raganato et al., 2017a). QWSD, despite its simplicity, obtained very good results, not far from those obtained by the best systems on the same benchmark, and it exhibits the best performances in the last evaluation datasets, namely SemEval 2013 and 2015, the best performance when classifying polysemous nouns and the second best for adjectives.

5 Discussion and Conclusions

After the influential paper from Reimers and Gurevych (2017) it is clear that we should report the mean and standard deviation of various runs with the same setting in order to get a more accurate picture of the real systems performances. We had no possibility to reproduce all the results from the other systems because some of them lack of a public code, others do not work well or it is not sufficiently clear how to set up them and for others the results we obtained using the public code and the described parameters are so different from the published ones that, to be fair with the colleagues, we prefer not to take any position on it and thus we put in Table 2 our best result, as some other studies did. In any case, to be consistent with the community trend, we made ten different experiments to generate the cPVs and repeated the training procedure accordingly. Our results over the ten runs are very similar to the best one: 70.1±0.22. The problem of results reproducibility for empirical studies is becoming rather serious (Wieling et al., 2018).

But, why our method is working well? A possible explanation for these results could be traced back to the interference phenomenon. A superposition of state vectors is usually a valid state vector even if they are not orthogonal, thus we can consider the vector representing a polysemous word, $|W\rangle$ as in (5), as a superposition state. As showed by (Khrennikov and Basieva, 2014; Aliakbarzadeh and Kitto, 2016) this can still produce the interference phenomenon even if these vectors are non-orthogonal. The presence of the interference term when computing the word sense probability over the context subspace might explain the good results we obtained. This point deserves further studies in order to verify this idea and understand how to use this term to drive the disambiguation process and obtain even better results.

Our Quantum WSD system relies on complex vector representations for words and sentences; in this study, for ease of experimentation, we tested our proposal by using a simple extension of PVs to the complex domain, but in literature there are techniques to build better word/sentence embeddings that could be extended to the complex domain; the field of complex DNN is very active.

Another interesting idea worth to be explored regards the possibility of solving the disambiguation process for all ambiguous words in the sentence as a single process (Tripodi and Pelillo, 2017) by using specific properties of QMT.

We feel it is worth spending few words on the simplicity of the proposed system. The only training phase regards the production of cPVs and, as well as the standard `word2vec` application, is based on a very simple feedforward neural network employing very few non-linearities. The disambiguation phase based on QPT is fully deterministic and involves few linear algebra operations, namely matrix multiplications and orthogonalisation procedures. Looking at the perfor-

[5] http://purl.ox.ac.uk/ota/2554

System	Test datasets				All Test datasets				
	SE2	**SE3**	**SE13**	**SE15**	**Noun**	**Verb**	**Adj**	**Adv**	**ALL**
Most Frequent Sense baseline	65.6	66.0	63.8	67.1	67.6	49.6	73.1	80.5	65.5
IMS (Zhong and Ng, 2010)	70.9	69.3	65.3	69.5	70.4	56.1	75.6	82.9	68.9
IMS+emb (Iacobacci et al., 2016)	71.0	69.3	67.3	71.3	71.8	55.4	76.1	82.7	69.7
IMS-s+emb	72.2	70.4	65.9	71.5	71.9	56.9	75.9	84.7	70.1
Bi-LSTM+att+lex (Raganato et al., 2017b)	72.0	69.4	66.4	72.4	71.6	57.1	75.6	83.2	69.9
Bi-LSTM+att+lex+pos	72.0	69.1	66.9	71.5	71.5	57.5	75.0	83.8	69.9
supWSD (Papandrea et al., 2017)	71.3	68.8	65.8	70.0	-	-	-	-	69.1
supWSD+emb	72.7	70.6	66.8	71.8	-	-	-	-	70.6
supWSD-s+emb	72.2	70.3	66.1	71.6	-	-	-	-	70.1
GAS (Linear) (Luo et al., 2018b)	72.0	70.0	66.7	71.6	71.7	57.4	76.5	83.5	70.1
GAS (Conc)	72.1	70.2	67.0	71.8	72.1	57.2	76.0	84.4	70.3
GAS_ext (Linear)	72.4	70.1	67.1	72.1	71.9	58.1	76.4	84.7	70.4
GAS_ext (Conc)	72.2	70.5	67.2	72.6	72.2	57.7	76.6	**85.0**	70.6
CAN^w (Luo et al., 2018a)	72.3	69.8	65.5	71.1	71.1	57.3	76.5	84.7	69.8
CAN^s	72.2	70.2	69.1	72.2	73.5	56.5	76.6	80.3	70.9
HCAN	72.8	70.3	68.5	72.8	72.7	58.2	**77.4**	84.1	71.1
fastSense (Uslu et al., 2018)	73.5	**73.5**	66.2	73.2	-	-	-	-	71.7
IMSC2V+PR (Melacci et al., 2018)	73.8	71.9	68.2	72.8	73.1	77.1	60.6	83.5	71.8
IMSC2V+sSyn	**74.2**	71.8	68.1	72.8	71.9	76.2	57.6	83.2	**71.9**
IMSC2V+sSyn+PR	74.1	71.6	68.1	72.8	73.1	**77.3**	60.2	83.8	71.8
QWSD	70.5	69.8	**69.8**	**73.4**	**73.6**	54.4	77.0	80.6	70.6

Table 2: Results obtained by the proposed system (QWSD) compared with the SOTA (F-score). The first four columns show the results for the different test sets, while the last five the performances on all the four test sets joined together analysed w.r.t. the different parts of speech.

mances in Table 2 it is clear that the results are very near, and in some case better, than those obtained by system based on intricate DNN structures that require long training processes and a careful parameter tuning. This paper presents the results of a basic quantum system for WSD and the results are very encouraging; more work in this direction could drive to even better systems.

Codes and data are freely available[6].

References

D. Aerts, J. Broekaert, L. Gabora, and S. Sozzo. 2016a. Generalizing prototype theory: A formal quantum framework. *Frontiers in psychology* 7:418.

D. Aerts, J. Broekaert, L. Gabora, and S. (Eds.) Sozzo. 2016b. *Quantum Structures in Cognitive and Social Science*. Frontiers Media, Lausanne.

D. Aerts, S. Sozzo, and T. Veloz. 2015. Quantum structure of negation and conjunction in human thought. *Frontiers in Psychology* 6:1447.

E. Agirre and P. Edmonds. 2007. *Word Sense Disambiguation: Algorithms and Applications*. Springer Publishing Company.

M. Aliakbarzadeh and K. Kitto. 2016. Applying povm to model non-orthogonality in quantum cognition. *Lecture Notes in Computer Science* 9535:284–293.

S. Alxatib and J. Pelletier. 2011. On the psychology of truth-gaps. In R. Nouwen, R. van Rooij, U. Sauerland, and H. Schmitz, editors, *Vagueness in Communication*. Springer, Berlin Heidelberg, pages 13–36.

M. Arjovsky, A. Shah, and Y. Bengio. 2016. Unitary evolution recurrent neural networks. In *Proc. of the ICML 2016*. pages 1120–1128.

M. Ashtiani and M. Abdollahi Azgomi. 2015. A survey of quantum-like approaches to decision making and cognition. *Mathematical Social Sciences* 75:49–80.

B.E. Baaquie. 2018. *Quantum Field Theory for Economics and Finance*. Cambridge University Press.

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36:673–721.

I. Basile and F. Tamburini. 2017. Towards quantum language models. In *Proc. of EMNLP 2017*. pages 1840–1849.

W. Blacoe, E. Kashefi, and M. Lapata. 2013. A quantum-theoretic approach to distributional semantics. In *Proc. of HLT-NAACL 2013*. pages 847–857.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

J.R. Busemeyer. 2012. Introduction to quantum probability for social and behavioral scientists. In L. Rudolph and J. Valsiner, editors, *Qualitative Mathematics For the Social Sciences*, London: Routledge, pages 75–104.

[6] https://github.com/ftamburin/QWSD

D. Cer, Y. Yang, S. Kong, et al. 2018. Universal sentence encoder for english. In *Proc. of EMNLP 2018*. pages 169–174.

A. Conneau and D. Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *Proc. of LREC 2018*.

P. Edmonds and S. Cotton. 2001. Senseval-2: Overview. In *Proc. of SENSEVAL-2*. pages 1–5.

F.A. González and J.C. Caicedo. 2011. Quantum latent semantic analysis. In *Advances in Information Retrieval Theory, LNCS, 6931*. pages 52–63.

J.A. Hampton. 1988. Disjunction of natural concepts. *Memory & Cognition* 16(6):579–591.

E. Haven and A. Khrennikov. 2013. *Quantum Social Science*. Cambridge University Press, Cambridge.

E. Haven and A. (Eds.) Khrennikov. 2018. *Applications of Quantum Mechanical Techniques to Areas Outside of Quantum Mechanics*. Frontiers Media.

I. Iacobacci, M.T. Pilehvar, and R. Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proc. of ACL 2016*. pages 897–907.

L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S.A. Skirlo, M. Tegmark, and M. Soljacic. 2017. Tunable efficient unitary neural networks (EUNN) and their application to RNN. In *Proc. of ICML 2017*.

D. Kartsaklis, M. Lewis, and L. Rimell. 2016. *Proc. of the 2016 Workshop on Semantic Spaces at the Intersection of NLP, Physics and Cognitive Science*.

A. Khrennikov and I. Basieva. 2014. Quantum model for psychological measurements: From the projection postulate to interference of mental observables represented as positive operator valued measures. *NeuroQuantology* 12(3):324–336.

A. Khrennikov, I. Basieva, E.M. Pothos, and I. Yamato. 2018. Quantum probability in decision making from quantum information representation of neuronal states. *Scientific Reports* 8(1):16225.

R. Kiros, Y. Zhu, R.R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In C. Cortes et al., editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 3294–3302.

Q.V. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. of ICML'14*. pages 1188–1196.

Q. Li, S. Uprety, B. Wang, and D. Song. 2018. Quantum-inspired complex word embedding. In *Proc. of The Third Workshop on Representation Learning for NLP*. pages 50–57.

D. Liu, X. Yang, and M. Jiang. 2013. A novel classifier based on quantum computation. In *Proc. of ACL 2013*. pages 484–488.

F. Luo, T. Liu, Z. He, Q. Xia, Z. Sui, and B. Chang. 2018a. Leveraging gloss knowledge in neural word sense disambiguation by hierarchical co-attention. In *Proc. of EMNLP 2018*. pages 1402–1411.

F. Luo, T. Liu, Q. Xia, B. Chang, and Z. Sui. 2018b. Incorporating glosses into neural word sense disambiguation. In *Proc. ACL 2018*. pages 2473–2482.

S. Melacci, A. Globo, and L. Rigutini. 2018. Enhancing modern supervised word sense disambiguation models by semantic lexical resources. In *Proc. of LREC 2018*.

M. Melucci. 2015. *Introduction to Information Retrieval and Quantum Mechanics*. The IR Series 35. Springer, Berlin Heidelberg.

M. Melucci and K. van Rijsbergen. 2011. Quantum mechanics and information retrieval. In M. Melucci and R. Baeza-Yates, editors, *Advanced Topics in Information Retrieval*, Springer, Berlin Heidelberg, pages 125–155.

T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges et al., editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119.

G.A. Miller. 1995. Wordnet: A lexical database for english. *Communiations of the ACM* 38:39–41.

G.A. Miller, M. Chodorow, S. Landes, C. Leacock, and R.G. Thomas. 1994. Using a semantic concordance for sense identification. In *Proc. of HLT'94*. pages 240–243.

C. Moreira and A. Wichert. 2018. Are quantum-like bayesian networks more powerful than classical bayesian networks? *Journal of Mathematical Psychology* 82:73–83.

A. Moro and R. Navigli. 2015. Semeval-2015 task 13: Multilingual all words sense disambiguation and entity linking. In *Proc. of SemEval'15*. pages 288–297.

R. Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2):1–69.

R. Navigli, D. Jurgens, and D. Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Proc. of SemEval'13*. pages 222–231.

R. Navigli and S.P. Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193:217–250.

M.A. Nielsen and I.L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press, New York.

D.N. Osherson and E.E. Smith. 1981. On the adequacy of prototype theory as a theory of concepts. *Cognition* 9(1):35 – 58.

S. Papandrea, A. Raganato, and C. Delli Bovi. 2017. Supwsd: A flexible toolkit for supervised word sense disambiguation. In *Proc. of EMNLP 2017*. pages 103–108.

J. Pennington, R. Socher, and C.D Manning. 2014. Glove: Global vectors for word representation. In *Proc. EMNLP 2014)*. pages 1532–1543.

C.S. Perone, R. Silveira, and T.S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *CoRR* abs/1806.06259.

E.M. Pothos and J.R. Busemeyer. 2013. Can quantum probability provide a new direction for cognitive modeling? *Behavioral and Brain Sciences* 36(3):255–274.

S. Pradhan, E. Loper, D. Dligach, and M. Palmer. 2007. Semeval-2007 task-17: English lexical sample, SRL and All words. In *Proc. of SemEval'07*. pages 87–92.

A. Raganato, J. Camacho-Collados, and R. Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proc. of EACL 2017*. pages 99–110.

A. Raganato, C. Delli Bovi, and R. Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proc. of EMNLP 2017*. pages 1156–1167.

N. Reimers and I. Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proc. of EMNLP 2017*. pages 338–348.

E. Rosch. 1973. Natural categories. *Cognitive Psychology* 4:328–350.

A.M. Sarroff. 2018. *Complex Neural Networks for Audio*. Ph.D. thesis, Dartmouth College, Hanover, NH.

S. Scardapane, S. Van Vaerenbergh, A. Hussain, and A. Uncini. 2018. Complex-valued neural networks with nonparametric activation functions. *IEEE Transactions on Emerging Topics in Computational Intelligence*. pages 1–11.

B. Snyder and M. Palmer. 2004. The english all-words task. In *Proc. of SENSEVAL-3*. pages 41–43.

S. Subramanian, A. Trischler, Y. Bengio, and C.J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *Proc. of ICLR 2018*.

I. Sutskever, O. Vinyals, and Q.V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS'14*. pages 3104–3112.

K. Taghipour and H.T. Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proc. of CoNLL 2015*. pages 338–344.

C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. Felipe Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C.J. Pal. 2018. Deep complex networks. In *Proc. of ICLR 2018*.

R. Tripodi and M. Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics* 43(1):31–70.

T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML 2016*. pages 2071–2080.

P.D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research* 37:141–188.

A. Tversky. 1977. Features of similarity. *Psychological Review* 84(4):327–352.

T. Uslu, A. Mehler, D. Baumartz, A. Henlein, and W. Hemati. 2018. FastSense: An efficient word sense disambiguation classifier. In *Proc. of LREC 2018*.

K. van Rijsbergen. 2004. *The Geometry of Information Retrieval*. Cambridge University Press, New York.

R. V. Vidhu Bhala and S. Abirami. 2014. Trends in word sense disambiguation. *Artificial Intelligence Review* 42(2):159–171.

A. Wendt. 2015. *Quantum Mind and Social Science: Unifying Physical and Social Ontology*. Cambridge University Press.

M. Wieling, J. Rawee, and G. van Noord. 2018. Reproducibility in computational linguistics: Are we willing to share? *Computational Linguistics* 44(4):641–649.

M.M. Wilde. 2013. *Quantum Information Theory*. Cambridge University Press.

S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas. 2016. Full-capacity unitary recurrent neural networks. In D.D. Lee et al., editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 4880–4888.

J.M. Yearsley, E.M. Pothos, J.A. Hampton, and A.B. Duran. 2015. Towards a quantum probability theory of similarity judgments. *LNCS* 8951:132–145.

Z. Zhong and H.T. Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proc. of ACL 2010 Demos*. pages 78–83.

G. Zuccon, L.A. Azzopardi, and K. van Rijsbergen. 2009. The quantum probability ranking principle for information retrieval. In L.A. Azzopardi et al., editors, *Advances in Information Retrieval Theory*, Springer, Berlin Heidelberg, pages 232–240.

Understanding Neural Machine Translation by Simplification: The Case of Encoder-free Models

Gongbo Tang[1] **Rico Sennrich**[2,3] **Joakim Nivre**[1]

[1]Department of Linguistics and Philology, Uppsala University
[2]School of Informatics, University of Edinburgh
[3]Institute of Computational Linguistics, University of Zurich
firstname.lastname@{lingfil.uu.se, ed.ac.uk}

Abstract

In this paper, we try to understand neural machine translation (NMT) via simplifying NMT architectures and training encoder-free NMT models. In an encoder-free model, the sums of word embeddings and positional embeddings represent the source. The decoder is a standard Transformer or recurrent neural network that directly attends to embeddings via attention mechanisms. Experimental results show (1) that the attention mechanism in encoder-free models acts as a strong feature extractor, (2) that the word embeddings in encoder-free models are competitive to those in conventional models, (3) that non-contextualized source representations lead to a big performance drop, and (4) that encoder-free models have different effects on alignment quality for German→English and Chinese→English.

1 Introduction

Neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) has emerged in the last few years and has achieved new state-of-the-art performance. However, NMT models are black boxes for humans and are hard to interpret. NMT models employ encoder-decoder architectures where an encoder encodes source-side sentences and an attentional decoder generates target-side sentences based on the outputs of the encoder. In this paper, we attempt to obtain a more interpretable NMT model by simplifying the encoder-decoder architecture. We train encoder-free models where the sums of word embeddings and sinusoid embeddings (Vaswani et al., 2017) represent the source. The decoder is a standard Transformer (Vaswani et al., 2017) or recurrent neural network (RNN) that attends to embeddings via attention mechanisms.

As motivation for our architecture simplification, consider the attention mechanism[1] (Bahdanau et al., 2015; Luong et al., 2015), which has been introduced to extract features from the hidden representations in encoders dynamically. Attention and alignment were initially used interchangeably, but it was soon discovered that the attention mechanism can behave very differently from traditional word alignment (see Ghader and Monz, 2017; Koehn and Knowles, 2017). One reason for this discrepancy is that the attention mechanism operates on representations that potentially includes information from the whole sentence due to the encoder's recurrent or self-attentional architecture. Intuitively, bypassing these encoder layers and attending word embeddings directly could lead to a more alignment-like, and thus predictable and interpretable behavior of the attention model.

By comparing encoder-free models with conventional models, we can better understand the working mechanism of NMT, figure out which components are more crucial, and learn lessons for improvement. Experimental results show that there is a significant gap between the two models. We focus on exploring what leads to the big gap.

As the embeddings in encoder-free Transformers (*Trans-noEnc*) are only influenced by attention mechanisms, without the help of encoders, we hypothesize that the quality of embeddings leads to the gap between Transformers and *Trans-noEnc* models. Thus we conduct both qualitative and quantitative evaluations of the embeddings from Transformers and *Trans-noEnc* models. We also hypothesize that the attention distribution in *Trans-noEnc* is not spread out enough for extract-

[1] We refer to the encoder-decoder attention mechanism unless otherwise specified.

Proceedings of Recent Advances in Natural Language Processing, pages 1186–1193,
Varna, Bulgaria, Sep 2–4 2019.

ing contextual features. However, we find that word embeddings and attention distributions are not the major reasons causing the distinct gap. We further explore NMT encoders. We find that even NMT models with one layer encoder get significant improvement compared to encoder-free models which indicates that non-contextualized source representations lead to the evident gap.

In encoder-free models, the attention attends to source embeddings rather than hidden representations fused with the context. We hypothesize that encoder-free models generate better alignments than default models. We evaluate the alignments generated on German→English (DE→EN) and Chinese→English (ZH→EN). We find that encoder-free models improve the alignments for DE→EN but worsen the alignments for ZH→EN.

2 Related Work

2.1 Understanding NMT

The attention mechanism has been introduced as a way to learn an alignment between the source and target text, and improves encoder-decoder models significantly, while also providing a way to interpret the inner workings of NMT models. However, Ghader and Monz (2017) and Koehn and Knowles (2017) have shown that the attention mechanism is different from a word alignment. While there are linguistically plausible explanations in some cases – when translating a verb, knowledge about the subject, object etc. may be relevant information – other cases are harder to explain, such as an off-by-one mismatch between attention and word alignment for some models. We suspect that such a pattern can be learned if relevant information is passed to neighboring representations via recurrent or self-attentional connections.

Ding et al. (2017) show that only using attention is not sufficient for deep interpretation and propose to use layer-wise relevance propagation to better understand NMT. Wang et al. (2018) replace the attention model with an alignment model and a lexical model to make NMT models more interpretable. The proposed model is not superior but on a par with the attentional model. They clarify the difference between alignment models and attention models by saying that that the alignment model is to identify translation equivalents while the attention model is to predict the next target word.

In this paper, we try to understand NMT by simplifying the model. We explore the importance of different NMT components and what causes the performance gap after model simplification.

2.2 Alignments and Source Embeddings

Nguyen and Chiang (2018) introduce a lexical model to generate a target word directly based on the source words. With the lexical model, NMT models generate better alignments. Kuang et al. (2018) propose three different methods to bridge source and target word embeddings. The bridging methods can significantly improve the translation quality. Moreover, the word alignments generated by the model are improved as well.

Our encoder-free model is a simplification and only attends to the source word embeddings. We aim to interpret NMT models rather than pursuing better performance.

Different from previous work, Zenkel et al. (2019) introduce a separate alignment layer directly optimizing the word alignment. The alignment layer is an attention network learning to attend to source tokens given a target token. The attention network can attend to either the word embeddings or the hidden representations or both of them. The proposed model significantly improves the alignment quality and performs as well as the aligners based on traditional IBM models.

3 Experiments

In addition to training Transformer and *Trans-noEnc* models, we also compare *Trans-noEnc* with NMT models based on RNNs (*RNNS2S*). We train *RNNS2S* models without encoders (*RNNS2S-noEnc*), without attention mechanisms (*RNNS2S-noAtt*), and without both encoders and attention mechanisms (*RNNS2S-noAtt-noEnc*) to explore which component is more important for NMT. We also investigate the importance of positional embeddings in *Trans-noEnc*.

3.1 Experimental Settings

We use the *Sockeye* (Hieber et al., 2017) toolkit, which is based on MXNet (Chen et al., 2015), to train models. Each encoder/decoder has 6 layers. For *RNNS2S*, we choose long short-term memory (LSTM) RNN units. Transformers have 8 attention heads. The size of embeddings and hidden states is 768. We tie the source, target, and output embeddings. The dropout rate of embeddings and Transformer blocks is set to 0.1. The dropout rate of RNNs is 0.2. All the models are trained with a single GPU. During training, each mini-batch contains 2,048 tokens. A model checkpoint is saved every 1,000 updates. We use *Adam* (Kingma and Ba, 2015) as the optimizer. The initial learning rate is set to 0.0001. If the performance on the validation set has not improved for 8 checkpoints, the learning rate is multiplied by 0.7. We set the early stopping patience to 32 checkpoints.

The training data is from the WMT15 shared task (Bojar et al., 2015) on Finnish–English (FI–EN). We choose *newsdev2015* as the validation set and use *newstest2015* as the test set. All the BLEU (Papineni et al., 2002) scores are measured by *SacreBLEU* (Post, 2018). There are about 2.1M sentence pairs in the training set after preprocessing. We learn a joint BPE model with 32K subword units (Sennrich et al., 2016). We employ the models that have the best perplexity on the validation set for the evaluation. We set the beam size to 8 during inference.

To test the universality of our findings, we conduct experiments on DE→EN and ZH→EN as well. For DE→EN, we use the training data from the WMT17 shared task (Bojar et al., 2017). We use *newstest2013* as the validation set and *newstest2017* as the test set. We learn a joint BPE model with 32k subword units. For ZH→EN, we choose the CWMT parallel data of the WMT17 shared task for training. We use *newsdev2017* as the validation set and *newstest2017* as the test set. We apply Jieba[2] to Chinese segmentation. We then learn 60K subword units for Chinese and English separately. There are about 5.9M and 9M sentence pairs in the training set after preprocessing in DE→EN and ZH→EN, respectively.

3.2 Results

Table 1 shows the performance of all the trained models. Encoder-free models (*NMT-noEnc*s) perform rather poorly compared to conventional NMT models.[3] It is interesting that *Trans-noEnc* obtains a BLEU score similar to the *RNNS2S* model. Even though the attention networks only attend to the non-contextualized word embeddings, *Trans-noEnc* still performs as well as the *RNNS2S* by paying attention to the context with multiple attention layers. Tang et al. (2018a) find that the superiority of Transformer models is attributed to the self-attention network which is a powerful semantic feature extractor. Given our results, we conclude that the attention mechanism is also a strong feature extractor in *Trans-noEnc* without self-attention in the encoder.

Model	Param.	PPL	BLEU
Transformer	104.4M	9.6	18.9
Trans-noEnc	71.4M	11.7	15.9
RNNS2S	91.5M	14.9	15.9
RNNS2S-noEnc	64.3M	25.2	12.5
RNNS2S-noAtt	90.3M	33.3	8.2
RNNS2S-noAtt-noEnc	63.1M	53.7	4.1
Trans-noEnc-noPos	71.4M	26.6	7.1

Table 1: The performance of NMT models. PPL is the perplexity on the development set. BLEU scores are evaluated on *newstest2015*. "Param." denotes the number of parameters.

The attention mechanism improves encoder-decoder architectures significantly. However, there are no empirical results to clarify whether encoders or attention mechanisms are more important for NMT models. We compare *RNNS2S-noAtt*, *RNNS2S-noEnc*, and *RNNS2S-noAtt-noEnc* to explore which component contributes more to NMT models.[4] In Table 1, *RNNS2S-noEnc* performs much better than *RNNS2S-noAtt*. Moreover, the gap between *RNNS2S-noEnc* and *RNNS2S-noAtt-noEnc* is distinctly larger than the gap between *RNNS2S-noAtt* and *RNNS2S-noAtt-noEnc*. These results hint that attention mechanisms are more powerful than encoders in NMT.

The positional embedding is also very important to Transformers which holds the sequential information. We are interested in the extent to which the positional embedding affects the translation performance. We further simplify the model by removing the positional embedding in the source (*Trans-noEnc-noPos*). *Trans-noEnc-noPos* has a dramatic drop in BLEU score. It is even worse than *RNNS2S-noAtt*. This result indicates that positional information is indeed crucial for Transformers.

[2] https://github.com/fxsjy/jieba

[3] We also trained a *Transformer* with less parameters (64.3M). The *Transformer* still achieved a significantly better BLEU score (18.2) than *Trans-noEnc* which means that the number of parameters is not the primary factor in this case.

[4] Because the encoders and decoders in Transformers are only connected via attention, we only conduct this experiment on *RNNS2S* models.

Word	Neighbors	
	Transformer	*Trans-noEnc*
more	less, better, greater, most, **further**	less, greater, better, **fewer**, most
for	to, in, on, of, **with**	to, in, of, on, **towards**
ole (not)	olekaan (not the), **kykene** (unable to), kuulu (part of), **pysty** (upright), **ollut** (been)	olekaan, kuulu (part of), **ei** (no/not), **ene** (a suffix), **liity** (sign up)
Arvoisa (honorable)	arvoisa, Arvoisat (honorable), **arvoisaa**, **arvoisan** (honorable), hyvät (honorable)	arvoisa, **arvoisat**, hyvät, Arvoisat, **Hyvä** (honorable)

Table 2: Neighbors of the selected word embeddings. Bold words are distinct neighbors.

4 Analysis

Trans-noEnc is obviously inferior to *Transformer* but we are more interested in investigating what causes the performance gap. In this section, we will test our hypotheses on embedding quality and attention distributions.

4.1 Embeddings

Word embeddings are randomly initialized by default and learned during training. As the embeddings in *Trans-noEnc* are only updated by attention mechanisms, we hypothesize that embeddings in *Trans-noEnc* are not well learned and therefore affect translation performance. We test our hypothesis by (1) evaluating the embeddings in the two models manually and (2) initializing *Trans-noEnc* with the learned embeddings in *Transformer* as pre-trained embeddings.

Qualitative Evaluation We select the 150 most frequent tokens from the vocabulary and then manually evaluate the quality of embeddings by comparing the 5 nearest neighbors.

The quality of English word embeddings is quite good based on the output of neighbors. Finnish word embeddings are not as good as English word embeddings. Table 2 exhibits four examples, two English words, “more”, “for” and two Finnish word, “ole” (not), “Arvoisa” (honorable). The neighbors of “more” in *Transformer* and *Trans-noEnc* are all quite related words, including comparatives and “most” which is the superlative of “more”. The words “further” and “fewer” are more different neighbors but both are related to “more”. For the Finnish word “ole” (not), both models have negative words as neighbors, but there are different unrelated words as well. We can see that the qualities of neighbors in two embedding matrices are close. We cannot easily distinguish which embedding matrix is better based on the neighbors.

Quantitative Evaluation In addition to the qualitative evaluation, we also conduct a quantitative evaluation. We first employ the learned embeddings from *Transformer* to initialize the embedding parameters in *Trans-noEnc*. The pre-trained embeddings can be either fixed or not fixed during training. Table 3 gives the BLEU scores of these models. The pre-trained embeddings slightly improve the BLEU score.

Embeddings	Random	Fixed	Not-fixed
BLEU	15.9	16.1	16.2

Table 3: BLEU scores of *Trans-noEnc*s with different embedding initialization. “Random” means no pre-trained embeddings. “Fixed” and “Not-fixed” denote using pre-trained embeddings.

The evaluation reveals that the embeddings from *Trans-noEnc* are competitive to those of *Transformer*. Thus, we can rule out differences in embedding quality as the main factor for the performance drop.

4.2 Attention Distribution

The attention networks in *Trans-noEnc* only attend to word embeddings. To better capture the sentence-level context, the attention networks need to distribute more attention to the context. We test our hypothesis that the attention distributions in *Trans-noEnc* are not as distributed as those in *Transformer*. If the attention distributions in *Transformer* are more spread out than those in *Trans-noEnc*, it means that smaller weights are distributed to contextual features by *Trans-noEnc*.

$$E_{At}(y_t) = -\sum_{i=1}^{|x|} At(x_i, y_t) \log At(x_i, y_t) \quad (1)$$

We use attention entropy (Equation 1) (Ghader and Monz, 2017) to measure the concentration of the attention distribution at timestep t. We then average the attention entropy at all the timesteps as the final attention entropy. x_i denotes the ith source token, y_t is the prediction at timestep t, and $At(x_i, y_t)$ represents the attention distribution at timestep t. The attention mechanism in Transformer has multiple layers, and each layer has multiple heads. In each layer, we average the attention weights from all the heads.

Figure 1 shows the entropy of attention distributions in both models. The attention distributions are consistent with the finding in Tang et al. (2018b) that the distribution gets concentrated first and then becomes distributed again. *Transformer* has lower entropy, which potentially is because the contextual information has been encoded in the hidden representations. The attention entropy of *Trans-noEnc* is clearly higher than that of *Transformer* in each attention layer. The attention in *Trans-noEnc* tends to extract features from source tokens more uniformly which indicates that the attention mechanism compensates for the fact that embeddings are non-contextualized by distributing attention across more tokens.

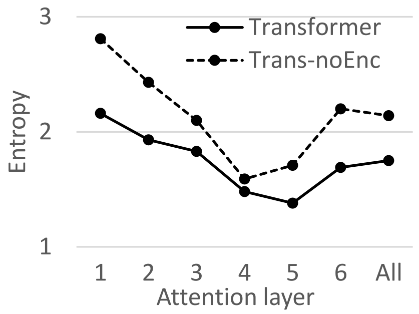


Figure 1: The attention entropy of each attention layer and the entire attention mechanism.

4.3 Encoders

We have shown that embeddings and attention distributions are not the primary reasons causing the gap between *Transformer* and *Trans-noEnc*. Therefore, we move to explore encoders.

Encoders are responsible for providing source hidden representations to the decoder. Encoder-free models have to use word embeddings to represent source tokens without the help of encoders. Thus, the source-side representations probably lead to the performance gap.

We train NMT models with different encoder layers. Table 4 displays the performance of Transformer models that have different layers in the encoder. It is clear that even the model with only a 1-layer encoder outperforms *Trans-noEnc* (0-layer) by 1.7 BLEU points, which accounts for 56.7% of the performance gap. The results seem to show that source-side hidden representations are crucial in NMT.

Layers	Param.	PPL	BLEU
0	71.4M	11.7	15.9
1	76.9M	10.3	17.6
3	87.9M	9.9	18.4
5	98.9M	9.5	18.6
6	104.4M	9.6	18.9

Table 4: The performance of Transformer models that have different layers in the encoder, including the perplexity (PPL) on the development set and the BLEU scores on *newstest2015*.

It has been shown that encoders could extract syntactic and semantic features in NMT (Belinkov et al., 2017a,b; Poliak et al., 2018). In the meantime, contextual information is encoded in hidden representations as well. Hence we conclude that the quality of source representations is the main factor causing the big gap between *Transformer* and *Trans-noEnc*.

In Table 5, our additional experiments on DE→EN and ZH→EN confirm that models with contextualized representations are much better. Transformer models always outperform *Trans-noEnc* models substantially.

Lan.	*Trans-noEnc*	*Transformer*	Impr.
DE→EN	29.5	32.6	10.5%
ZH→EN	18.5	20.9	13.0%

Table 5: The improvement (Impr.) of employing encoders in *Trans-noEnc*s on DE→EN and ZH→EN.

5 Alignment

The weights of the attention mechanism can be interpreted as an alignment between the source and target text. We further explore whether encoder-free models have better alignments than default models. We evaluate the alignments on two manually annotated alignment data sets. The first one has been provided by RWTH,[5] and consists of 508 DE→EN sentence pairs. The other one is from Liu and Sun (2015) and contains 900 ZH→EN sentence pairs. We apply alignment error rate (AER) (Och and Ney, 2003) as the evaluation metric.

Following Luong et al. (2015); Kuang et al. (2018), we also force the models to produce the reference target words during inference to get the alignment between input sentences and their reference outputs. We merge the subwords after translation following the method in Koehn and Knowles (2017).[6] We sum the attention weights in all attention heads in each attention layer.[7] Given a target token, the source token with the highest attention weight is viewed as the alignment of the current target token (Luong et al., 2015). However, a source token maybe aligned to multiple target tokens and vice versa. Therefore, we also align a source token to the target token that has the highest attention weight given the source token. Experimental results show that the bidirectional method achieves higher alignment quality.

Figure 2 displays the evaluation results. The alignment in the fourth attention layer achieves the best performance. Therefore, we only compare the alignments in the fourth layer. In DE→EN, the encoder-free model has a lower AER score (0.41) than the default model (0.43) which accords with our hypothesis. However, in ZH→EN, the alignment quality of the encoder-free model (0.46) is worse than that of the default model (0.43). The effect on alignment quality is not clear-cut for encoder-free models given limited language pairs.

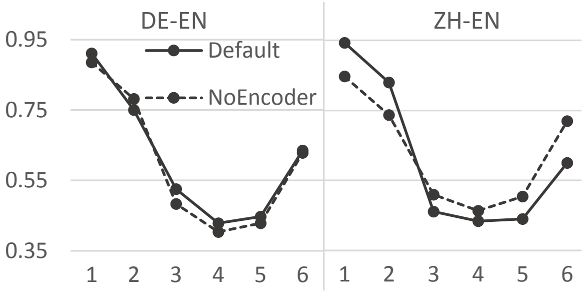


Figure 2: The AER scores of alignments in different attention layers on DE→EN and ZH→EN.

6 Conclusion

To better understand NMT, we simplify the attentional encoder-decoder architecture by training encoder-free NMT models in this paper. The non-contextualized source representations in encoder-free models cause a big performance drop, but the word embeddings in encoder-free models are shown competitive to those in default models. Also, we find that the attention component in encoder-free models is a powerful feature extractor, and can partially compensate for the lack of contextualized encoder representations.

Regarding the interpretability of attention, our results do not show that the attention mechanism in encoder-free models is consistently more alignment-like: only attending to source embeddings improves the alignment quality on DE→EN but makes the alignment quality worse on ZH→EN.

Acknowledgments

We thank all reviewers for their valuable and insightful comments. Gongbo Tang is mainly funded by the Chinese Scholarship Council (grant number 201607110016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA. https://arxiv.org/abs/1409.0473.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 861–872. http://aclweb.org/anthology/P17-1080.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 1–10. http://www.aclweb.org/anthology/I17-1001.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang,

[5] https://www-i6.informatik.rwth-aachen.de/goldAlignment/

[6] (1) If an input word is split into subwords, we sum their attention weights. (2) If a target word is split into subwords, we average their attention weights.

[7] Following Tang et al. (2018b), we tried maximizing the attention weights as well but got worse alignment quality.

Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics, Copenhagen, Denmark, pages 169–214. http://aclweb.org/anthology/W17-4717.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Lisbon, Portugal, pages 1–46. http://aclweb.org/anthology/W15-3001.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In *Proceedings of the Workshop on Machine Learning Systems in Neural Information Processing Systems 2015*. http://arxiv.org/abs/1512.01274.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, pages 1150–1159. https://doi.org/10.18653/v1/P17-1106.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, pages 30–39. http://www.aclweb.org/anthology/I17-1004.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *arXiv preprint arXiv:1712.05690* http://arxiv.org/abs/1712.05690.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1700–1709. http://www.aclweb.org/anthology/D13-1176.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, California, USA. https://arxiv.org/abs/1412.6980.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, Vancouver, Canada, pages 28–39. http://www.aclweb.org/anthology/W17-3204.

Shaohui Kuang, Junhui Li, António Branco, Weihua Luo, and Deyi Xiong. 2018. Attention focusing for neural machine translation by bridging source and target embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 1767–1776. http://aclweb.org/anthology/P18-1164.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. Austin, Texas, USA, pages 2295–2301.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Toan Nguyen and David Chiang. 2018. Improving lexical choice in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, USA, pages 334–343. http://aclweb.org/anthology/N18-1031.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51. http://www.aclweb.org/anthology/J03-1002.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://www.aclweb.org/anthology/P02-1040.

Adam Poliak, Yonatan Belinkov, James Glass, and Benjamin Van Durme. 2018. On the evaluation of semantic phenomena in neural machine translation using natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*

(Short Papers). Association for Computational Linguistics, New Orleans, Louisiana, USA, pages 513–523. https://doi.org/10.18653/v1/N18-2082.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Belgium, Brussels, pages 186–191. http://aclweb.org/anthology/W18-6319.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Neural Information Processing Systems 2014*. Montréal, Canada, pages 3104–3112. https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018a. Why self-attention? a targeted evaluation of neural machine translation architectures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, pages 4263–4272. http://aclweb.org/anthology/D18-1458.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2018b. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. Association for Computational Linguistics, Belgium, Brussels, pages 26–35. http://aclweb.org/anthology/W18-6304.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 6000–6010. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Weiyue Wang, Derui Zhu, Tamer Alkhouli, Zixuan Gan, and Hermann Ney. 2018. Neural hidden markov model for machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 377–382. https://www.aclweb.org/anthology/P18-2060.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2019. Adding interpretable attention to neural translation models improves word alignment. *arXiv preprint arXiv:1901.11359* https://arxiv.org/abs/1901.11359.

Text-Based Joint Prediction of Numeric and Categorical Attributes of Entities in Knowledge Bases

V Thejas
BITS Pilani, India

Abhijeet Gupta and **Sebastian Padó**
IMS, University of Stuttgart, Germany

Abstract

Collaboratively constructed knowledge bases play an important role in information systems, but are essentially always incomplete. Thus, a large number of models has been developed for Knowledge Base Completion, the task of predicting new attributes of entities given partial descriptions of these entities. Virtually all of these models either concentrate on numeric attributes (`<Italy,GDP,2T$>`) or they concentrate on categorical attributes (`<Tim Cook,chairman,Apple>`).

In this paper, we propose a simple feed-forward neural architecture to jointly predict numeric and categorical attributes based on embeddings learned from textual occurrences of the entities in question. Following insights from multi-task learning, our hypothesis is that due to the correlations among attributes of different kinds, joint prediction improves over separate prediction.

Our experiments on seven FreeBase domains show that this hypothesis is true of the two attribute types: we find substantial improvements for numeric attributes in the joint model, while performance remains largely unchanged for categorical attributes. Our analysis indicates that this is the case because categorical attributes, many of which describe membership in various classes, provide useful 'background knowledge' for numeric prediction, while this is true to a lesser degree in the inverse direction.

1 Introduction

Collaboratively constructed knowledge bases (CCKBs) such as WikiData (Vrandečić and Krötzsch, 2014), YAGO (Suchanek et al., 2008), FreeBase (Bollacker et al., 2008) or DBPedia (Bizer et al., 2009), capture world knowledge in the shape of a graph structure where nodes denote *entities* and edges denote *attributes* (Hitzler et al., 2009). Their collaborative construction importantly enables them to avoid the scaling problems encountered by expert-constructed knowledge bases. Thus, CCKBs have come to play an important role in information systems, forming the basis for a wide range of natural language processing applications (Hovy et al., 2013) such as question answering (Berant et al., 2013; Krishnamurthy and Mitchell, 2015) or representation learning for entities (Toutanova et al., 2015; Yaghoobzadeh et al., 2018).

The most crucial shortcoming of CCKBs is their incompleteness (Min et al., 2013; West et al., 2014) – not just with respect to the entities that they cover, but also with respect to the attributes present for entities that are nominally covered. This is not surprising: When a contributor to a knowledge base adds an entity, they will probably concentrate on the most salient attributes (e.g., for a scientist, *field* or *affiliation*), while other attributes (such as *parents* or *place of birth*) may be added later or never. This realization has led to a large boost to work in the area of *knowledge base completion*, that is, the prediction of attributes of entities that are currently missing from the CCKB (Bordes et al., 2013; Socher et al., 2013; Min et al., 2013; Guu et al., 2015; Gupta et al., 2017).

These methods, however, overwhelmingly concentrate on *categorical* attributes, that is, attributes whose values are themselves entities in the knowledge graph. As an example, consider the attribute *capital* that maps a *country* onto a *city* which is itself an entity (*Mexico – Mexico City, UK – London*). A prominent approach to the prediction of categorical attributes is as an operation in embedding space, which explains the popularity of embedding-based approaches for this task.

Much fewer studies has considered the prediction of *numerical* attributes of entities in CCKBs (Davidov and Rappoport, 2010; Gupta et al., 2015),

Proceedings of Recent Advances in Natural Language Processing, pages 1194–1202,
Varna, Bulgaria, Sep 2–4 2019.

Attribute	Value
latitude	41.90 N
longitude	12.49 E
GDP_per_capita::2015	29,957.8 US$
fertility_rate::2010	1.46
capital	Rome
containedBy	Western_Europe
containedBy	Europe
member_of	G8
member_of	European_Union

Table 1: Sample of numeric and categorical FreeBase attributes for *Italy*.

such as the attribute *GDP-1990* which maps a *country* onto a number denoting its gross domestic product in the year 1990. For many entities in CCKBs, numeric attributes actually form the majority of the attributes for entities. Still, these attributes are often seen as secondary because their values are not 'proper' entities but numeric constants that themselves do not possess interesting attributes.

In this paper, we investigate the hypothesis that *joint prediction of numeric and categorical attributes* can improve prediction quality for both attribute types. As a motivating example, consider the sample of both numeric and categorical attributes listed for the country *Italy* in the FreeBase CCKB (Bollacker et al., 2008), shown in Table 1. It is clear that, as assumed by most models concentrating on categorical attributes, these attributes correlate with one another, and therefore the presence of one attribute can serve as evidence for the presence of another attribute. For example, containment in Western Europe implies containment in Europe, and is correlated with membership in the European Union. However, similar correlations arguably hold between categorical and numeric attributes. For example, the high GDP per capita constitutes evidence for Italy's membership in the G8 political forum, or vice versa, membership in the European Union and the G8 points towards a high GDP per capita. Similarly, Italy's latitude and longitude (defined by FreeBase to be the capital's geolocation) determine Rome as the country's capital, and vice versa.

Concretely, in this paper we adopt two previous embedding-based models for the individual prediction of numeric and categorical attributes from textual data, respectively. We define a novel simple joint model that predicts both attribute types concurrently (Section 2) and evaluate these models on a sample of seven FreeBase domains (Section 3) and find that prediction improves substantially for numeric attributes, but remains constant for categorical attributes (Section 4). Our analysis indicates that this is the case because numeric attributes that are difficult to predict from text-based embeddings are still often correlated with categorical attributes, and that can thus profit from joint training, while this is not true for categorical attributes.

2 Predicting Numeric and Categorical Attributes from Text

The majority of methods to predict attributes for entities in CCKBs are based on techniques from representation learning. Specifically, they us distributed representations (i.e., vectors, also called *embeddings*) to represent the entities, and sometimes also the attributes. Embeddings can be built from different sources, such as the knowledge bases themselves (Bordes et al., 2013; Guu et al., 2015; Lin et al., 2015), from text corpora that mention these entities (Socher et al., 2013; Krishnamurthy and Mitchell, 2015), or from both (Toutanova et al., 2015; Yaghoobzadeh et al., 2018).

In this paper, we use two prediction models that build on embeddings that were built from text corpora, following the widely successful assumption that text corpora implicitly contain a large amount of world knowledge that can be extracted by observing the contexts in which words are used (the so-called distributional hypothesis) (Firth, 1957; Miller and Charles, 1991; Turney and Pantel, 2010; Mikolov et al., 2013). The formulation of attribute prediction on top of precomputed embeddings enables us to use rather simple supervised neural model which are generally considered the state of the art for computational models in natural language processing.

2.1 Numeric Prediction

The first model is a feed-forward neural network, shown on the left-hand side of Figure 1. It builds on a study that used a logistic regression model to predict the values of numeric values, scaled to the interval (0;1) (Gupta et al., 2015) to avoid the excessive influence of outliers that linear regression is sensitive to. The model uses an n-dimensional entity embedding as its input which is mapped through a $\tanh$ nonlinearity onto an h-dimensional hidden layer, which in turn maps onto an $|N|$-dimensional output layer (where $|N|$ is the number of the numeric attributes) using a sigmoid nonlinearity. In other words, each unit in the output layer corresponds to one numeric attribute, and the model predicts all numeric attributes simultaneously.

We use a variant of the mean cross-entropy loss function commonly used for logistic regression. Let $a \in A$ denote an attribute, $E(a) = \text{Tr}(a) \cup \text{Val}(a) \cup \text{Ts}(a)$ the set of entities for this attribute, partitioned into training, validation, and test sets, and $v_a(e)$ and $\hat{v}_a(e)$ the gold and predicted values for entity e, respectively. Then

$$\begin{aligned} L_{\text{num}} = -\sum_{a \in A} & \frac{1}{|A||Tr(a)|} \\ & \sum_{e \in Tr(a)} \big(v_a(e) \log \hat{v}_a(e) + \\ & \qquad (1 - v_a(e)) \log(1 - \hat{v}_a(e)) \big) \end{aligned} \tag{1}$$

Even though simple, this model shows good performance in predicting numeric attributes of entities in CCKBs, since distributed representations implicitly capture a large amount of world knowledge (Gupta et al., 2015) and the hidden layer enables the model to exploit correlations among numeric attributes.

2.2 Categorical Prediction

The second model (Gupta et al., 2017) is another feed-forward neural network, shown on the right-hand side of Figure 1. Again, it uses a precomputed n-dimensional entity embedding as its input. Since this model predicts only the value of one categorical attribute at one time, this embedding is complemented by a representation of the attribute, realized as a one-hot vector whose dimensionality is the number of categorical attributes $|C|$.[1] Again, the input is first mapped onto an h-dimensional hidden layer and then onto an output layer, passing through a $tanh$ nonlinearity in both steps.

In this model, the output layer is n-dimensional, like the input, and actually represents an embedding of the attribute value. For example, given the embedding for *Italy* and the attribute *capital* as input, the model should predict the embedding for *Rome*. To map the output of the model back onto an explicit entity, we perform a nearest-neighbor retrieval in the space of precomputed embeddings, which is feasible with specialized indexes (Babenko and Lempitsky, 2016).

The loss function we use for this model is a contrastive variant of mean squared error (MSE) loss: we minimize the MSE between the prediction and the correct embedding while maximizing the MSE between the prediction and a sample of confounders. Since MSE can be understood as (squared) Euclidean distance, this loss function pushes the predicted embedding towards the correct embedding and away from confounders:

$$\begin{aligned} L_{\text{cat}} = \sum_{a \in A} & \frac{1}{|A||Tr(a)|} \\ & \sum_{e \in Tr(a)} \big((v_a(e) - \hat{v}_a(e))^2 - \\ & \mu \sum_{e' \in \text{NN}(k, \hat{v}_a(e), Y - \{e\})} (v_a(e') - \hat{v}_a(e))^2 \big) \end{aligned} \tag{2}$$

The notation is the same as in Equation (1). Additionally, $\text{NN}(k, x, X)$ is a function that returns the k nearest neighbors of x in the set X, and μ a weight that trades off the positive and negative parts of the loss against each other. In this model, we do not need an indicator function as in the numeric attribute model, since the loss in this model is defined over seen attributes.

2.3 Joint Prediction

The similar structure of the two models described directly above makes it easy to define a joint model for the prediction of categorical and numeric attributes, shown in Figure 2. The new architecture re-uses the input layer from the categorical model, which subsumes the simpler architecture of the numeric one. It uses the same type of hidden layer, to which both the numeric output layer and the categorical output layer are attached. The nonlinearities are the same as in the individual models. Since the input to the model is still an entity embedding plus a categorical attribute, as in the categorical model, the model essentially predicts the numeric attributes of the entity "on the side".

Correspondingly, the loss function of this model is a weighted average of the losses of its parts:

$$L_{\text{joint}} = \alpha L_{\text{cat}} + (1 - \alpha) L_{\text{num}} \tag{3}$$

where α is the relative weight of the categorical loss. For the extreme values of $\alpha = 1$ and $\alpha = 0$, the joint model reverts to its component models.

[1] We experimented with learning a distributed representation of the attributes, but did not achieve better results.

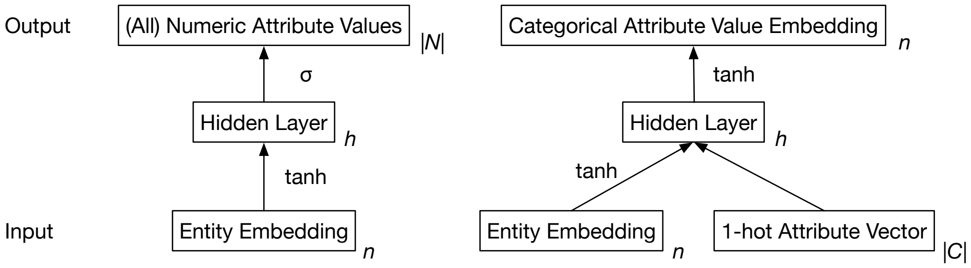


Figure 1: Individual model architectures. Prediction of numeric attributes (left-hand side) and categorical attributes (right-hand side). Subscripts in italics indicate dimensionality of layers (n: dimensionality of embedding space; h: dimensionality of hidden layer; $|N|$: number of numeric attributes, $|C|$: number of categorical attributes)

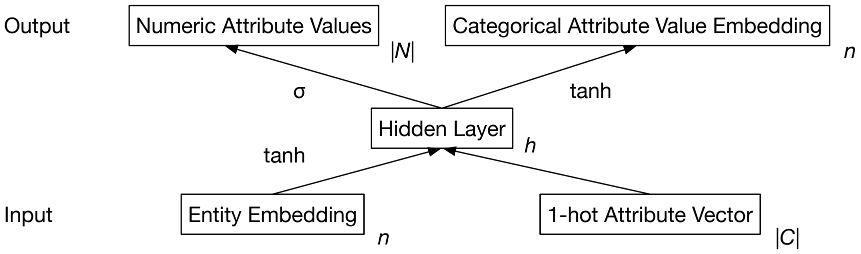


Figure 2: Joint model architecture for the simultaneous prediction of categorical and numeric attributes. Subscripts in italics indicate dimensionality of layers (n: dimensionality of embedding space; h: dimensionality of hidden layer; $|N|$: number of numeric attributes, $|C|$: number of categorical attributes)

Our hypothesis is that intermediate values of α will improve prediction quality for the two types of attributes. We expect this to be the case since joint training can be seen as an instance of multi-task learning, which is known to often positively impact the quality of the learned intermediate representations (Zhang and Yang, 2017). Note that this effect is not guaranteed, since we introduce competition among the two output layers, which may deteriorate the output of the 'losing' layer.

2.4 Discussion

Note that all three model assume that all entities share a common set of attributes: in the numeric model, these determine the shape of the output layer, and in the categorical mode, they determine the shape of the attribute input layer. While it is still possible to train global models, CCKBs are typically organized into top-level *domains* that share little to no attributes. For example, *people* (which have, e.g., birth and death dates) or *organizations* (which have e.g., personnel, turnover, profit numbers) have no attributes in common. Consequently, in the remainder of the paper, we adopt a *domain*-specific approach, learning and evaluating separate models for each domain.

3 Experimental Setup

3.1 Dataset and Embeddings

To our knowledge, there are no existing datasets that include both numeric and categorical attributes. For example, the widely used FB15K and WN18 datasets (Bordes et al., 2013) focus exclusively on categorical attributes. For this reason, we construct our own dataset which we make freely available on DANS at URL `https://doi.org/10.17026/dans-zxp-t7tf`.

We construct the dataset on the basis of the FreeBase CCKB (Bollacker et al., 2008). As sketched above, we proceed by domains and extract entities and attributes for six of the most populous top-level FreeBase domains (*animal, book, citytown, country, employer, organization, people*).

Since we build on pretrained embeddings for the entities in question, we only include entities if they are covered by the largest existing pretrained embedding space for proper names. This is the "Google News" embedding space that used a 100G token news corpus to compute embeddings specifically for FreeBase entities (Mikolov et al., 2013).[2] The embeddings are computed with the Word2Vec

[2] `https://code.google.com/p/word2vec/`

skip-gram algorithm, 1000 dimensions. Similarity, categorical attributes are only included if we have embeddings for both the entity and the value.

Finally, we split all domains into training, validation, and test sets (60%–20%–20%). The split is applied to each attribute type: at validation and test time, our models face no unseen attribute types, but unseen instances for each attribute. In the numeric and joint models, this means that the model will encounter 'incomplete' numeric output layers since some attributes of a given entity may be reserved for testing (cf. the left-hand side of Figure 1 as well as Figure 2). This does not hurt the model, though: The objective function, Equation (1), only ranges over attributes present in the training data.

Table 2 shows descriptive statistics for the resulting dataset. We consider just over 5000 entities for a total of 269 categorical attribute types and 1041 numeric attribute types.[3] Note that the domains differ considerably with regard to their numbers of entities, numbers of attributes, and relative prevalence of categorical and numeric attributes. For example, the *country* domain has the highest number of attributes, and about ten times as many numeric as categorical attributes. This reflects the large number of time series recorded for countries. In contrast, the *organization* domain has much fewer attributes overall, and more categorical than numeric attributes (e.g., location, founders, officers, business sector).

3.2 Evaluation

Categorical Attributes. As explained above in Section 2.2, we apply nearest neighbor mapping to the embedding output of the model to map its prediction back onto an entity symbol. Following earlier work (Gupta et al., 2017), we perform Information Retrieval-style ranking evaluation, mean reciprocal rank (MRR) (Manning et al., 2008). Reusing the notation from Sec. 2 and writing ra for rank, we define MRR as

$$\frac{1}{|T|}\sum_{a\in A}\sum_{e\in \mathrm{Ts}(a)}\frac{1}{\mathrm{ra}(\hat{v}_a(e), \mathrm{NN}(\infty, v_a(e), v_a(\mathrm{Ts}(a))))}$$

For each entity-attribute pair (e, a), MRR computes the (reciprocal) rank of the model's prediction $\hat{v}_a(e)$ in the nearest neighbor list of the true value $v_a(e)$. These values are averaged over all datapoints in the test set Ts.

Intuitively, MRR describes how close, on average, the predictions are to the correct one in terms of ranks: an MRR of 0.5 means that they are the second-nearest neighbors, an MRR of 0.3 means that they are the third-nearest neighbors, and so on. Thus, higher MRR values indicate better performance. We report results at the domain level as well as micro-averaged MRR for the complete dataset.

Numeric Attributes. For numeric attributes, we use the so-called normalized rank score (NRS). NRS is a variant of Spearman's correlation coefficient that takes into account both how correctly the entities in the test set are ranked with respect to each numeric attribute, and how consistent the predictions are with regard to the training set (Frome et al., 2013). We choose this evaluation over a numeric error-based one because it is more robust to outliers and sets a more realistic target for the prediction of numeric attributes (Gupta et al., 2015). NRS is defined as

$$\sum_{a\in A}\frac{1}{|A||\mathrm{Ts}(a)|}\operatorname*{med}_{e\in\mathrm{Ts}(a)}\{|\mathrm{ra}(\hat{v}_a(e), E(a)) - \mathrm{ra}(v_a(e), E(a))|\}$$

NRS measures divergence from the gold standard ranking. It has range [0;1], with smaller numbers indicating better performance: 0.2, for example, means that the prediction is, on average, off by 20% of the ranks. As before, we report the statistic for each domain, plus a micro-averaged NRS for the complete dataset.

3.3 Hyperparameters

Individual Models. We trained the two individual models and the joint model using AdaDelta optimization method, using the best parameters according to the literature (Zeiler, 2012), namely $\rho = 0.95$ and $\epsilon = 10^{-6}$. We trained until convergence or for at most 300 iterations with early stopping. All hyperparameters were explored on the validation set. We explored h, the size of the hidden layer, by setting it to values between 200 and 3000 with a step size of 200. We found h=2000 to yield good results for both models and adopted this number. In the model for categorical attributes, we followed earlier work (Gupta et al., 2017) by using just a single nearest neighbor for the negative part of the loss (k=1) and setting μ to 0.6.

Joint Model. To build the joint model, we retained the hyperparameter settings of the two individual models. We explored values of α between

[3] We removed attributes that were not populated for any entities in our entity set.

Domain	# Entities (train/val/test)	$\lvert C \rvert$	$\lvert N \rvert$
Animal	279/93/93	22	118
Book	16/5/6	8	2
Citytown	1783/594/595	57	62
Country	155/53/51	79	698
Employer	720/140/141	50	55
Organization	187/63/62	36	32
People	85/28/29	25	76
Sum	3225/976/977	277	1043

Table 2: Data set statistics. $|C|$: number of categorical attribute types. $|N|$: number of numeric attribute types

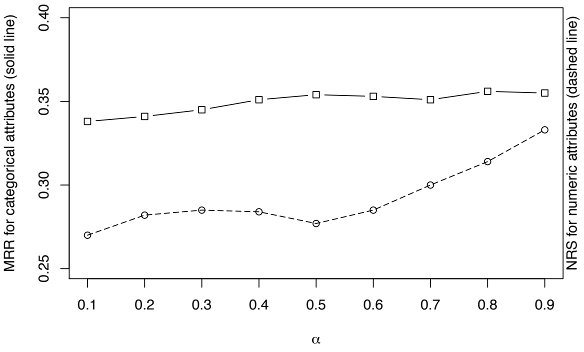


Figure 3: Hyperparameter exploration. Impact of different values of α on the *animal* domain for categorical (solid) and numeric (dashed) attributes (validation set).

0.1 and 0.99 on the validation set of the *animal* domain. The results for the joint model are shown in Figure 3. As expected, there is a trade-off between the two objectives: Results for categorical prediction improve for high values of α, where the model focuses on these attributes. Conversely, results for numeric prediction improve when the model pays more attention to these attributes, for low values of α (recall that lower NRS values are better). We chose $\alpha = 0.6$ as a value that gives both models a chance to profit from the joint setup.

3.4 Inference

Regarding inference in the models, the two individual prediction models are trivial, and so is the categorical part of the joint model:To predict the value of a categorical attribute for an entity, the numeric output can simply be ignored. To predict the value of a numeric attribute of entity, however, different inference procedures are possible. We used the simplest one, namely activating a random categorical attributes to query a numeric attribute (cf. Figure 2). We did not observe meaningful variance across the choice of different categorical attributes.

3.5 Baselines

We use two baseline models from previous studies. For categorical attributes, our baseline model ignores the entity. For each attribute, it predicts the frequency-ordered list of all values seen in the training set (Frequency Baseline). We also report on a baseline that simply models each attribute as a linear operation in embedding space (Mikolov et al., 2013; Bordes et al., 2013) defined as the centroid of all difference vectors for a given attribute between entities and their values for this attribute (Linear Baseline).

For numeric attributes, our baseline model predicts the mean value of the attribute seen in the training set (Mean Baseline).

4 Results and Discussion

Numeric Attributes. Table 3 shows the results as averaged normalized rank scores (NRS) for each domain as well as macro-averaged (Avg) scores for the complete test set. Recall that for NRS lower values are better.

We find that that the joint model (which predicts numeric and categorical attributes at the same time) yields substantially better results than the individual model on all domains, ranging between 0.03 (for *animal* and *people*) and 0.1 (*books*). Average performance on all domains improves from 0.3 by 0.07 to 0.23. In turn, the individual model outperforms the baseline on all domains except *people*, corresponding to a similar improvement by 0.07.

We see the best results of the joint model for *citytown* and the worst results for *people*. These numbers correlate with the numbers of entities present

	Domains							Mean
	Animal	Book	City	Country	Employer	Organization	People	
Joint Model	**0.284**	**0.276**	**0.211**	**0.293**	**0.215**	**0.225**	**0.387**	**0.229**
Individual Models	0.317	0.382	0.288	0.376	0.289	0.300	0.421	0.300
Mean BL	0.370	0.434	0.366	0.416	0.364	0.421	0.394	0.373

Table 3: Test set results on numeric attributes per domain (normalized rank score; lower is better). Best result for each domain marked in boldface.

	Domains							Mean
	Animal	Book	City	Country	Employer	Organization	People	
Joint Model	0.330	0.244	0.198	**0.105**	**0.118**	**0.096**	**0.352**	0.193
Individual Models	**0.331**	**0.256**	**0.202**	**0.105**	0.116	**0.096**	**0.352**	**0.195**
Linear BL	0.215	0.217	0.084	0.046	0.085	0.090	0.250	0.101
Frequency BL	0.247	0.225	0.045	0.018	0.028	0.022	0.173	0.064

Table 4: Test set results on categorical attributes per domain (mean reciprocal rank; higher is better). Best result for each domain marked in boldface.

for these domains (Table 2): citytown is the largest domain, with almost 1800 entities in the training set, while people is the smallest domain with 85 training entities. Thus, we surmise that the differences in performance are not due to inherent differences in the features to be predicted, but reflect the different amounts of training data available.

Categorical Attributes. Table 4 shows the results as mean reciprocal rank (MRR) scores for each domain as well as macro-averaged scores (Avg) for the complete test set. For MRR, higher values are better.

The joint and the individual model outperform both baselines for all domains, and outdistance them substantially in average performance. However, the two informed models show almost identical performance overall, with average MRRs of 0.193 and 0.195, respectively. For three domains (*country, organization, people*), they perform equally well. For one domain (*employer*), the joint model performs better, and for three domains (*animal, book, city*), the individual model does better. The results of the two models are generally extremely close to one another, with the largest difference between the models (of 0.012) appearing for *book*, the smallest domain where we would expect the largest variance. Overall, we attribute these differences to random fluctuation.

The performance of all models is markedly different across domains, with low results close to 0.1 for *organization* and *country* and high results over 0.3 for *animal* and *people*. This is in line with earlier results for categorical attributes that identified the predominance of one-to-many relations across domains as important predictor of performance (Gupta et al., 2017).

Discussion. Probably the most surprising outcome of our experiment is the asymmetry that we observe: Joint modeling achieves respective improvements over individual modeling for numeric attributes, but not for categorical attributes. In other words, the prediction of numeric attributes profits from the availability of categorical attributes, but not vice versa.

One potential explanation is a suboptimal setting of the α parameter (Equation (3)) that would let the joint model pay too much attention to the numeric attributes. However, Figure 3 indicates that this not the case: the change in performance on categorical attributes is relatively minor across values of α, in particular compared to the chance in performance on numeric attributes.

To search for alternative explanations, we performed a qualitative analysis of the models' predictions. What we observe is that many numeric attributes of entities have a relatively low degree of *contextual support* (Gupta et al., 2015), that is, the values of these attributes do not correlate well with salient textual characteristics of the occurrences of the entity name. For example, in the *animal* domain, attributes like 'life span' or 'litter size' describe relatively detailed properties of animal species. Such attributes are unlikely to be represented well in embeddings learned in an unsupervised manner from newspaper text. As another example, the *country* domain contains attributes like 'diesel price' or 'gender balance among members of parliament' that arguably suffer from the same problem.

We believe that the prediction of such attributes can profit from information added to the hidden layer by the categorical part of the objective (Fig. 2), since specific values of categorical attributes provide informative priors. For animals, life span and litter size differ, for example, among different animal classes, orders, etc.; for countries, fuel prices or gender equality are correlated with categorical attributes such as membership in organisations (OPEC, Nordic Council).

For categorical attributes, an earlier study (Gupta et al., 2017) found that difficulty arose both from lack of contextual support and from list-valued attributes. In contrast to the numeric side, though, lack of support for categorical attributes can often not be compensated by access to numeric information. An examples, consider attributes such as 'disputed territories' from the *country* domain, or 'supplier' from the *organization* domain; arguably the values of these attributes is so specific that numeric information cannot help. Nor can the fundamental problem of list-valued attributes be alleviated by numeric information. Instead, this would require a fundamentally different prediction mechanism that supports list-valued attributes (Lin et al., 2015).

5 Conclusion

This paper is located in the area of knowledge base completion, that is, the task of complementing knowledge bases with missing relations, which is particularly pressing for collaboratively constructed knowledge bases. We focus on an understudied subproblem of knowledge base completion, namely the prediction of numeric, as opposed to categorical, attributes.

We assume a text-based approach that uses corpus-derived entity embeddings as the basis for attribute prediction. Building on top of two existing models for categorical and numeric attributes, the first contribution of this paper is a joint model for the prediction of these two attribute types. The second contribution is an empirical evaluation of separate vs. joint modeling on a novel dataset, where we find that numeric attributes profit substantially from a joint model, while categorical attributes do not. A qualitative analysis of the predictions indicates that there is indeed an asymmetry: in cases where the values of numeric attributes are difficult to predict from text-based embeddings, categorical information about the entity can often serve as a prior, whereas difficult-to-predict categorical attributes are often so specific that numeric attributes do not help.

To the best of our knowledge, this paper presents the first joint model for the prediction of numeric and categorical attributes. The joint model that we present is a straightforward combination of individual models for the two attribute types, both of which are purely text-based. A first step to improve the models would be to learn, or at least fine-tune, text-based embeddings in a task-specific manner, in order to enable the embeddings to pay attention to infrequent context cues that are nevertheless highly informative for particular attributes. On a more fundamental level, embeddings can be made to take both textual evidence and the structure of the knowledge base into account, as has been demonstrated for categorical attributes (Toutanova et al., 2015; Yaghoobzadeh et al., 2018). Finally, a direction for future research that would address in particular the difficulties in predicting categorical attribute could be the development of a neural architecture that explicitly accounts for list-valued attributes.

References

Artem Babenko and Victor S. Lempitsky. 2016. Efficient indexing of billion-scale datasets of deep descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063, Las Vegas, NV.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, pages 1533–1544, Seattle, WA.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia – A crystallization point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of ACM SIGMOD*, pages 1247–1249, Vancouver, Canada.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795, Lake Tahoe, NV.

Dmitry Davidov and Ari Rappoport. 2010. Extraction and approximation of numerical attributes from the web. In *Proceedings of ACL*, pages 1308–1317, Uppsala, Sweden.

J. R. Firth. 1957. A synopsis of linguistic theory 1930–55. In *Studies in Linguistic Analysis*, pages 1–32. The Philological Society, Oxford.

Andrea Frome, Greg Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In *Proceedings of NIPS*, pages 2121–2129, Lake Tahoe, NV.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedings of EMNLP*, pages 12–21, Lisbon, Portugal.

Abhijeet Gupta, Gemma Boleda, and Sebastian Padó. 2017. Distributed prediction of relations for entities: The easy, the difficult, and the impossible. In *Proceedings of STARSEM*, pages 104–109, Vancouver, BC.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*, pages 318–327, Lisbon, Portugal.

Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. 2009. *Foundations of semantic web technologies*. CRC Press.

Eduard Hovy, Roberto Navigli, and Simone Paolo Ponzetto. 2013. Collaboratively built semi-structured content and artificial intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Jayant Krishnamurthy and Tom M Mitchell. 2015. Learning a compositional semantics for freebase with an open predicate vocabulary. *Transactions of the Association for Computational Linguistics*, 3:257–270.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, Austin, TX.

Chris Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, Lake Tahoe, NV.

G Miller and W Charles. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6(1):1–28.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of NAACL-HLT*, pages 777–782, Atlanta, Georgia.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934, Lake Tahoe, CA.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6:203–217.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of EMNLP*, pages 1499–1509, Lisbon, Portugal.

Peter D Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of ACM*, 57(10):78–85.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of WWW*, pages 515–526, Seoul, Korea.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2018. Corpus-level fine-grained entity typing. *Journal of Artificial Intelligence Research*, 61:835–862.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. In *CoRR, abs/1212.5701*.

Yu Zhang and Qiang Yang. 2017. An overview of multi-task learning. *National Science Review*, 5:30–43.

SenZi: A Sentiment Analysis Lexicon for the Latinised Arabic (Arabizi)

Taha Tobaili[1], **Miriam Fernandez**[1], **Harith Alani**[1],
Sanaa Sharafeddine[2], **Hazem Hajj**[3], **and Goran Glavaš**[4]

[1]Knowledge Media Institute, The Open University
[2]Department of Computer Science and Mathematics, Lebanese American University
[3]Department of Electrical and Computer Engineering, American University of Beirut
[4]Data and Web Science Group, Universität Mannheim

{taha.tobaili, miriam.fernandez, h.alani}@open.ac.uk,
sanaa.sharafeddine@lau.edu.lb, hh63@aub.edu.lb, goran@informatik.uni-mannheim.de

Abstract

Arabizi is an informal written form of dialectal Arabic transcribed in Latin alphanumeric characters. It has a proven popularity on chat platforms and social media, yet it suffers from a severe lack of natural language processing (NLP) resources. As such, texts written in Arabizi are often disregarded in sentiment analysis tasks for Arabic. In this paper we describe the creation of a sentiment lexicon for Arabizi that was enriched with word embeddings. The result is a new Arabizi lexicon consisting of 11.3K positive and 13.3K negative words. We evaluated this lexicon by classifying the sentiment of Arabizi tweets achieving an F1-score of 0.72. We provide a detailed error analysis to present the challenges that impact the sentiment analysis of Arabizi.

1 Introduction

Arabizi, a portmanteau of *Arabic* and *Englizi* (English), is a written form of dialectal Arabic (DA) often used by Arabic speakers for informal communication in messaging applications and on social media enabling them to type Arabic words using Latin letters (Yaghan, 2008). Arabizi lacks a consistent orthography and reflects the various dialects of Arabic, which differ from one Arab region to another and from the formal Modern Standard Arabic (MSA) phonetically, morphologically, and syntactically.

Studies show that Arabizi has reached 12% of the Latin script tweets in Lebanon and 25% of the Latin script tweets in Egypt (Tobaili, 2016). It is a common way of communication among the youth (Keong et al., 2015; Muhammed et al., 2011; Allehaiby, 2013) and has been actively used during relevant events in the Arab world such as the Arab spring (Basis-Technology, 2012). Despite the growth of this newly born written language, Arabic sentiment analysis approaches often disregard Arabizi text due to the challenges it poses and the scarcity of NLP resources to process it (Bies et al., 2014).

In this work we contribute to the sentiment analysis of Arabizi by creating a new Arabizi sentiment lexicon (SenZi) for the Lebanese dialect. We annotated a 3.4K Arabizi Twitter dataset to evaluate the lexicon and to train an Arabizi language identifier. We used this identifier to create an Arabizi corpus of 1M public Facebook comments. We widened the coverage of SenZi by enriching it with inflectional and orthographic forms for each sentiment word using word embeddings on the corpus reaching 11.3K positive and 13.3K negative words. All resources and detailed description are made public and freely accessible on the project's webpage[1].

The rest of the paper is structured as follows: Section 2 explains the nature of Arabizi and the challenges it poses. Section 3 reviews the related work. Section 4 presents the annotated datasets and the compiled corpus. Section 5 presents the pipeline for creating SenZi. Section 6 presents the sentiment analysis and results. Section 7 discusses the contributions and limitations of this work. Finally, Section 8 concludes the paper.

2 Background and Challenges

Arabizi privileged its users to transcribe their mother tongue dialect in Latin script at their comfort, free of grammar and orthographic rules. This section dissects the formation of Arabizi, the linguistic issues associated with it, and the challenges it poses onto NLP.

2.1 Background

Arabizi naturally inherits the rich morphology of Arabic but introduces an inconsistent orthography

[1] https://project-rbz.kmi.open.ac.uk

Proceedings of Recent Advances in Natural Language Processing, pages 1203–1211,
Varna, Bulgaria, Sep 2–4 2019.

and codeswitching.

Morphology: Arabic is an inflectional language where a given word may have a wide range of inflectional forms to express gender, tense, case, number, or perspective. Each of these inflectional forms could be written with different pronoun affixes. An Arabic lemma is inflected by the attachment of clitics, prefixes, and suffixes, the insertion of infixes, or the deletion or replacement of some letters, resulting in a deep morphological shift. For example, the following dialectal words are few inflectional forms of the word زكي *zake / smart*: *azkiya, zakeya /smart-people* (regular and irregular plural forms), *tetzeka, tetzeke, tetzeko / you-are-outsmarting* (masculine, feminine, and plural), *azka / smarter-than*, and *ma azkek, azkekon, azkeke / how-smart-you-are!* (masculine, feminine, and plural).

Orthography: Arabic is rich in guttural phonemes. It contains two voiceless fricatives خ ح, two voiced fricatives ع غ, a voiced plosive ق, and a glottal stop ء. It also contains distinct consonants with similar phonemes known as soft and emphasised or heavy consonants. Arabic contains five pairs of light and heavy consonants: *q*: ك ق, *t*: ط ت, *d*: د ض, *s*: ص س, and *th*: ذ ظ. This is even exacerbated in Levant Arabic where the ق *q* is pronounced as a glottal stop ء, both ظ and ذ *th* (as in *them*) as ز *z*, and the ث *th* (as in *thrill*) as س *s*. Additionally, there are short and long vowel letters. Short vowels are the diacritics, marks above or below the letters as in كَتَبَ, but they are not scripted in most social texts as in كتب, because a native reader would comprehend the text without the diacritics. These factors had lead to an inconsistent orthography in the transcription of Arabic in Latin script. Moreover, users map the Arabic phonemes with Latin alpha numeral in accordance with their dialect, some transcription standards of the region, and their individual choice of letters. For example:

1. Dialect: The guttural ق *q* is expressed as a guttural *g* in Gulf Arabic but a glottal stop ء in some Levantine Arabic dialects, therefore it is mapped with the number 2 in Levant dialect Arabizi e.g., قلبي *qalbi / my heart* in MSA, *galbi* in Gulf Arabizi, and *2albi* or *albi* in Levant Arabizi.
2. Transcription Standards: Some transcriptions became normalised among Arab regions, such as mapping the guttural consonants غ and خ with the numerals 8 and 5 in some countries like Egypt and Jordan (Aboelezz, 2009; Allehaiby, 2013; BIANCHI, 2012) while *gh* and *kh* are more common in Lebanon (Sullivan, 2017).
3. Choice of Letters: It is up to every user's personal choice whether to transcribe some, all, or none of the vowel phonemes either because the short vowels are diacritics or the text is informal and readable without the vowel letters. The following words are few orthographic forms of the word حبيبي *habibi / darling* or *my-love*: *7abibi, 7bb, 7bbi, 7abebe, 7bibi, hbb, or habb*.

Codeswitching: Arabizi users constantly switch between Arabizi and Latin script languages, mainly English and French. For example, *Hi! kifak, cava?*, a common trilingual greeting from Lebanon. Codeswitching may occur within individual sentences or within conversations, posing a challenge for data collection and analysis.

2.2 Challenges

Lexical Sparsity: As mentioned earlier (in Section 2.1), Arabizi words can derive a large range of inflectional forms and each form can be transcribed in several orthographic variants. This leads to a high degree of lexical sparsity. Therefore, sentiment lexicons with one or few forms for each sentiment word are insufficient to capture the high number of inflections and variants that could be derived from each Arabizi word.

Word Ambiguity: Apart from words that are naturally polysemous, transcribing Arabic phonemes that have no equivalent in English Latin script may lead to ambiguity. Ambiguous words are generated by:

1. Transcribing a short Arabic vowel phoneme, a diacritic, as a vowel letter in Latin script. For example, transcribing the word ضيعة (short vowel *a*, a diacritic originally) / *village* as *day3a* becomes ambiguous with ضايعة (long vowel *a*) / *lost* or *confused*.
2. Transcribing one Latin script letter for two distinct Arabic letters. This is common for the soft and heavy consonants. For example, transcribing درب (soft *d*) / *route* as *dareb* be-

comes ambiguous with ضرب (heavy *d*) / *hit*.

Transliteration: Transliteration in this context is the automatic conversion of Arabizi into Arabic script. With the heterogeneity of Arabic dialects and the inconsistency of orthography, transliteration can not be achieved in a straightforward Latin to Arabic mapping. The most accurate transliterators are online tools that generate a list of possible transliterations for every input word such as Yamli[2] and Google Input Tools[3]. These tools are designed to help Arabic users output MSA text by typing in Latin script Arabic word by word. Transliterating whole Arabizi texts to Arabic produces orthographic errors. Google Translate for example, detects Arabizi, converts it to Arabic, and translates it to the target language. It translated the tweet *da5l jamelik w hadamtik / Oh-my (expression), your-beauty and your-humour (feminine)* to *inside your camel and demolished* due to the dialect (Lebanese), choice of letters, and word ambiguity.

3 Literature Review

Recent efforts in creating lexical resources for Arabic focus mainly on MSA (Badaro et al., 2014; Eskander and Rambow, 2015; Al-Twairesh et al., 2016) and DA (Abdul-Mageed and Diab, 2014). Several works that analysed sentiment from Arabic social data filtered out Arabizi completely from their datasets (Al-Kabi et al., 2013, 2014; Duwairi and Qarqaz, 2014), missing the sentiment from a considerable portion of the population in general and the youth in specific.

To the best of our knowledge, (Duwairi et al., 2016; Mataoui et al., 2016; GUELLIL et al., 2018) are the only works that looked into sentiment analysis for Arabizi. All three papers proposed to transliterate Arabizi to Arabic. (Duwairi et al., 2016) transliterated Jordanian dialect Arabizi to Arabic using their own transliterator without evaluating the transliterations. (Mataoui et al., 2016) transliterated Algerian dialect Arabizi as part of their sentiment analysis for Algerian Arabic pipeline. They used Google Translate without evaluating the transliterations as well. (GUELLIL et al., 2018) created a rule-based transliterator that generates several transliterations per word, then used a language model based on a large corpus to select the best transliteration. Thus, minimising the error of transliteration but maximising the complexity of the task.

The following papers propose sophisticated works for transliterating Egyptian (Darwish, 2014; Al-Badrashiny et al., 2014; Eskander et al., 2014) and Algerian dialect Arabizi (Guellil et al., 2017). The limitations of these works include transliterating datasets manually, hand-crafting rules to preprocess and normalise Arabizi, and mapping Arabizi with Arabic heuristically.

In this work we aim to advance the state of the art in Arabic sentiment analysis by analysing Arabizi directly, without the need to filter, preprocess, or transliterate it.

4 Data Collection and Annotation

This section describes the collection and annotation of Twitter datasets to evaluate SenZi and train an Arabizi identifier. It then describes the creation of a Facebook corpus that was used (in Section 5.2) to enrich SenZi with inflectional and orthographic forms.

4.1 Annotated Twitter Datasets

Collection: We used the Twitter stream API to collect live tweets that have geographic coordinates lying within the region of Lebanon. We collected 177K tweets intermittently over the period of one year, between 2016 and 2017. We filtered out the Arabic tweets that were identified by Twitter as Arabic: ar (80K). The remaining dataset contains 97K tweets in Latin script such as, Arabizi, English and French or codeswitched tweets among these languages. Twitter misidentified Arabizi and codeswitched tweets as (ht, tr, in, hi, pt, nl, ct, ey) where some stand for (Haitian, Turkish, Hindi, Portuguese, and Dutch). To accurately identify the Arabizi tweets, we resorted for a manual annotation task.

Preprocessing: We removed the URLs, hashtags, mentions, and non-ASC characters and deleted duplicated tweets and tweets that lack an alphabet, obtaining a filtered dataset of 66K tweets.

Annotation: We selected 30K tweets randomly and created a user friendly annotation platform that displays these tweets in different random order for every user. The platform asks each user:

1. Is the tweet written mostly in Arabizi?
   (*yes, no, I don't know*)

[2] http://www.yamli.com

[3] http://www.google.com/inputtools/try

2. What is the sentiment of the tweet? (*positive, negative, neutral, I don't know*)

We assigned this task to three Lebanese undergraduate students and guided them to:

1. Count the number of Arabizi words in codeswitched tweets to determine if these tweets are mostly written in Arabizi.
2. Consider the sentiment that the tweets infer regardless of present expressions e.g., *haha tabashna bl exam* / *haha we failed the exam* is a negative tweet.
3. Answer *I don't know* for ambiguous tweets.

A screenshot of the developed annotation platform is presented in Figure 1. Further details about the platform and the annotation process can be found on the project's webpage.

Results: Table 1 presents the annotation of the 30K tweets for Question 1: *Is the tweet written mostly in Arabizi?* We had a total of 4.3K *yes*, 27.6K *no*, and 641 *I don't know*. We applied Fleiss' Kappa (Fleiss, 1971) to measure the agreement among the annotators scoring a substantial agreement of 0.74 (Landis and Koch, 1977).

Table 1: Arabizi Annotation of 30K Tweets

Tweets	Arabizi	Not Arabizi	IDK	Kappa
30K	4.3K	27.6K	641	0.74

From the 4.3K Arabizi tweets, there were (3.4K tweets) where at least two answers match for *Arabizi-yes* and (2.2K Tweets) where all three answers match for *Arabizi-yes*. We balanced the 2.2K Arabizi with a 2.2K non-Arabizi tweets to create an Arabizi identification (AI) dataset.

Table 2 presents the annotation of the (3.4K tweets) for Question 2: *What is the sentiment of the tweet?* We had a total of 1.2K *positive*, 1.4K *negative*, 2.1K *neutral*, and 172 *I don't know* scoring a fair agreement of 0.33 Fleiss' Kappa.

Table 2: Sentiment Annotation of the (3.4K Tweets)

Tweets	Pos	Neg	Neutral	IDK	Kappa
3.4K	1.2K	1.4K	2.1K	172	0.33

From the 3.4K Tweets, there were (2.9K Tweets) where at least two answers match for the sentiment of the Arabizi tweets. They consist of 801 *positive*, 881 *negative*, 1.2K *neutral*, and 7 *I don't know*. We balanced an 800 positive with 800 negative tweets to create the sentiment analysis (SA) dataset.

As a result, we had two datasets:

1. AI Dataset: 4.4K Tweets (2.2K Arabizi and 2.2K not Arabizi).
2. SA Dataset: 1.6K Arabizi Tweets (800 positive and 800 negative).

We used the AI Dataset to train the Arabizi identifier (in Section 4.2) and the SA Dataset to evaluate SenZi (in Section 6.1).

4.2 Automatic Arabizi Identification

We used the AI Dataset (from Section 4.1) to train a Support Vector Machine (SVM) classifier with the tweets' unigrams as input features. We shuffled the dataset and split it into 10 folds for cross validation. The average of the classification results for all folds are presented in Table 3.

Table 3: Arabizi Identification
4.4K Tweets: 2.2K Arabizi - 2.2K non-Arabizi

Recall	Precision	F1-score	Accuracy
0.93	0.97	0.95	0.95

4.3 Facebook Corpus

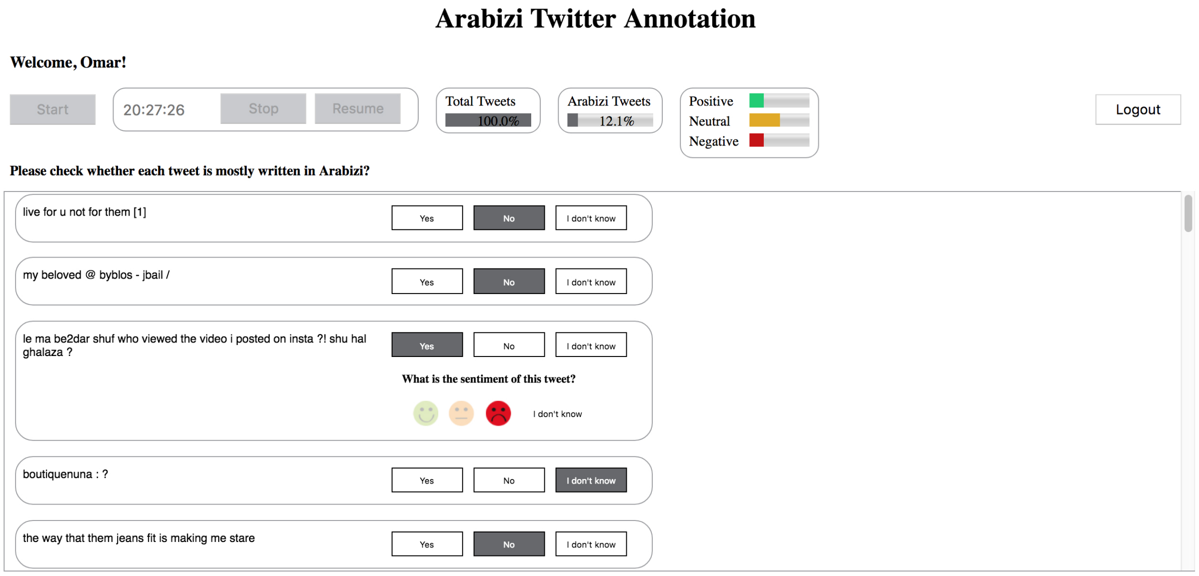
We created an Arabizi corpus of 1M Facebook public comments collected from 49 popular and active Lebanese pages[4]. The pages vary in genre such as, news, comedy, and politics. We used this corpus (in Section 5.2) to create a word embeddings space to discover inflectional and orthographic forms of SenZi's sentiment words.

Collection: We wrote a script that uses Facebook API to iterate over all posts (texts, images, and videos) in a public page and extract all the comments and replies from every post. It collects all Latin script comments and replies in reverse chronological order up to the very first post posted by the page. As we extracted the comments and their replies, we skipped comments that contain Arabic text. We ran the script over the 49 public pages in 2017 resulting in a 2.2M Latin script Facebook comments.

Preprocessing: We removed the URLs, mentions, and media attachments and deleted duplicated comments and comments that lack an alphabet, reducing the comments to 2.1M.

Identification: We used the trained Arabizi identifier (from Section 4.2) to identify the Arabizi text, obtaining a corpus of 1M Arabizi Facebook comments.

[4] The list of pages can be found at the project's webpage.



20:27:26

Total Tweets 100.0%

Arabizi Tweets 12.1%

Please check whether each tweet is mostly written in Arabizi?

live for u not for them [1] — Yes [✓ No] I don't know

my beloved @ byblos - jbail / — Yes [✓ No] I don't know

le ma be2dar shuf who viewed the video i posted on insta ?! shu hal ghalaza ? — [✓ Yes] No I don't know

What is the sentiment of this tweet? — 🙂 😐 [✓ ☹] I don't know

boutiquenuna : ? — Yes No [✓ I don't know]

the way that them jeans fit is making me stare — Yes [✓ No] I don't know

Figure 1: Annotation Platform

5 SenZi

This section presents the pipeline of building SenZi. It was built in two phases:

1. Lexicon Generation: Using existing resources to generate an initial list of Arabizi sentiment words.
2. Lexicon Expansion: Expanding this sentiment word list using the created Facebook corpus (from Section 4.3).

5.1 Lexicon Generation

Resources: We started with two English sentiment lexicons and one Lebanese dialect word list as seeds to build SenZi. We chose two of the most common English sentiment lexicons in the literature: Hiu and Liu[5] (2K positive and 4.8K negative) and the MPQA[6] (2.7K positive and 4.9K negative) (Wilson et al., 2005). LivingArabic is a list of 7.1K Lebanese dialect words compiled by the Living Arabic project[7]. We generated the Lebanese Arabizi sentiment words in five steps:

1. Combine the existing English lexicons.
2. Translate to Arabic.
3. Select the dialectal sentiment words.
4. Combine the resulting Arabic lexicons.
5. Transliterate to Arabizi.

5.1.1 Combination: English Lexicons

We took the union of Hiu Liu and MPQA to have a list of 2.7K positive and 5.1K negative words. We call this list HL-MPQA.

5.1.2 Translation

We used an online dictionary *bab.la*[8] to translate HL-MPQA to MSA. We wrote a script that inputs every word from HL-MPQA into *bab.la* and copies the single-word translations keeping the multi-word expressions for a future work. We generated 4.2K positive and 5.2K negative unique MSA words. We call this list HL-MPQA-Ar.

5.1.3 Selection

HL-MPQA-Ar: Since we aimed to create a Lebanese dialect lexicon, we needed to filter HL-MPQA-Ar from terms that are not common to the Lebanese dialect. We asked a Lebanese graduate student to select the dialectal sentiment words. The student selected 537 positive and 1K negative dialectal terms.

LivingArabic: We assigned an annotation task to three undergraduate Lebanese students to select the sentiment words from LivingArabic. The students selected 531, 672, and 1K sentiment words. We took 732 words (179 positive and 553 negative) where at least two students had agreed on the sentiment.

[5] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[6] https://mpqa.cs.pitt.edu/lexicons/subj_lexicon

[7] http://www.livingarabic.com

[8] https://bab.la

5.1.4 Combination: Arabic Lexicons

We took the union of the selected dialectal words from HL-MPQA-Ar and the selected sentiment words from LivingArabic to have a list of 607 positive and 1.4K negative Arabic words.

5.1.5 Transliteration

We asked the Lebanese graduate student to transliterate the resulting dialectal Arabic words to Arabizi. This marked the first version of SenZi, consisting of 2K Arabizi words (607 positive and 1.4K negative).

5.2 Lexicon Expansion

As mentioned in Section 2, Arabizi inherits the rich morphology of Arabic and introduces an inconsistent orthography causing a high degree of lexical sparsity. We addressed this challenge by expanding SenZi automatically to cover a range of inflectional and orthographic forms for each sentiment word.

We created a word vector space by training a word embeddings fastText skipgram model (Bojanowski et al., 2016) on the 1M Arabizi FB comments corpus (from Section 4.3). We retrieved a vector of nearest word neighbours for every SenZi word. Then, we matched the inflectional and orthographic forms from the retrieved vector with the SenZi word automatically. We learned heuristically that a retrieved Arabizi word is a form of a SenZi word if it contains the same sequence of consonant letters of that SenZi word. We maximised the inflectional and orthographic forms from the retrieved vector by normalising its words lightly to stay consistent with the words of SenZi. We ran this expansion twice, recursively.

5.2.1 First Expansion:

1. Retrieve a vector of 50 nearest word neighbors for each SenZi word.
2. Normalise the orthography of these words:
   - Replace *8, 5* and *ch* with *gh, kh* and *sh.*
   - Remove repeated letters (exaggeration).
3. Take the nearest word neighbors (in their original form, before normalisation) that contain the same consonant letter sequence of the SenZi word.

For example, the consonant letter sequence *tyb* from the word *tayab* / *cute* or *tasty* matched 30 inflectional and orthographic forms such as, *atyab, atyabak, atyabo, 2tyab, tayoub, taybe, tayoubi, taybeee, taybin,* and *tayoubin,*.

We note that applying this technique against all words in the corpus would match many irrelevant words because most Arabic words stem from triliteral words, but in this case we are limited with the nearest word neighbors, i.e., words that are semantically related.

This expanded SenZi to 12.3K words (5.1K positive and 7.2K negative).

5.2.2 Second Expansion:

1. Retrieve a vector of 50 nearest word neighbors for each *new* word.
2. Normalise the orthography of these words.
3. Take the nearest word neighbors that contain the same consonant letter sequence of the *original* SenZi word (not the new word).

For example, the word *atyabak* was retrieved in the first expansion of the word *tayab*. In the second expansion we match the consonant letter sequence *tyb* of the original word *tayab* with the word neighbors of the newly retrieved word *atyabak*. This further expanded *tayab* with two new inflectional forms of *atyabak* in two different orthographies *atyabek, atyabik, atyabkon,* and *atyabkoun* / *cute-singular-feminine* and *cute-plural* in the second person perspective.

We cleaned SenZi by deleting duplicates of words and words that occurred in both the positive and negative lists. This further expanded SenZi to 24.6K words (11.3K positive and 13.3K negative).

6 Evaluation

This section describes the sentiment analysis approach and results and presents a manual error analysis.

6.1 Evaluation Setup and Results

We followed the evaluation method of AraSenti, an Arabic sentiment lexicon proposed by (Al-Twairesh et al., 2016), to evaluate SenZi. we applied a 2-class sentiment classification using a simple lexicon-based approach. We evaluated SenZi before and after the expansion against the SA Dataset (from Section 4.1) which consists of 800 positive and 800 negative tweets.

We created a list of 10 negators and expanded it to 170 words[9] using the same lexicon expansion technique. We classified a sentiment word in its opposite sentiment class if it is preceded by a

[9] List of negators available at the project's webpage.

negator. However, the negator ما *ma* acts as an intensifier in some cases. For example, the *ma* in *ma ajmala! / How beautiful-she/it-is!* precedes an inflection of *jamil / pretty* that begins with a glottal stop أ *2* or *a*, we therefore exempted this negator from negation if followed by a glottal stop.

We matched the positive and negative words in the tweets with SenZi and classified the tweet *positive* if the positive matches were greater than the negative matches, *negative* if the negative matches were greater than the positive matches, and *no sentiment* otherwise. Since this is a 2-class classification and the dataset is balanced, we classifed tweets with *no sentiment* as *positive* or *negative* randomly. The average results are presented in Table 4.

Table 4: SenZi Evaluation
Lexicon-Based Classification

	R	P	F	A
SenZi Original	0.56	0.59	0.57	0.58
SenZi 1st Expansion	0.74	0.64	0.69	0.67
SenZi 2nd Expansion	0.79	0.66	0.72	0.69

Enriching SenZi with inflectional and orthographic forms pushed the F1-score by a solid 0.15 over the baseline. This implies that the word forms and variants play a significant role in sentiment analysis for Arabizi.

6.2 Error Analysis

We provide a detailed error analysis for the lexicon-based classification using the best version of SenZi (2nd expansion) in classifying 800 positive and 800 negative tweets (SA Dataset from Section 4.1). The confusion matrix of this classification is presented in Table 5.

Table 5: Confusion Matrix
Lexicon-Based Classification

Actual	Classified		
	Positive	Negative	No Sentiment
Positive	55%	3%	42%
Negative	13%	39%	48%

Most of the error (45%) lies in not determining a sentiment class for the tweets. As such, we extracted a sample of 100 positive and 100 negative tweets that were wrongly classified for a manual assessment. We checked all the words in the sample whether matched or missed by SenZi. The challenges that impact the performance of the lexicon-based approach and their percentages are presented in Table 6 followed by a small discussion for each challenge.

Sentiment form not in the lexicon: The main factor for not classifying sentiment tweets was due to lexical sparsity, the same challenge that we addressed in this paper. Although we expanded SenZi from 2K to 24.6K words with an increase of 0.23 in recall, it still did not match 38% of inflectional and orthographic forms of SenZi's sentiment words.

Sentiment word is in English: Although the Twitter dataset was annotated as mostly Arabizi, 12% of the unclassified sentiment words are written in English. Sentiment words in English appeared in the positive set slightly more than the negative set with expressions like *my love, miss you, happy birthday, best wishes,* and *good luck* over cursing and swearing in the negative set. Borrowing is also common in Arabizi e.g., *luvik* and *missik* for *love-you-feminine* and *miss-you-feminine* in the second person perspective.

Neutral word classified: The drawback of the automatic expansion of words is a decrease in precision with 14% wrong classification of neutral words in this case.

Multi-word expressions and sarcasm: Many common multi-word expressions that express sentiment or sarcasm lack sentiment words, hence bypass a simple lexicon-based approach. For example, *to2bor albe / burry my heart* expresses love or *ras kbeer / big head* means stubborn.

No sentiment words: 9% of the unclassified tweets lack sentiment words with a higher tendency in the negative class. For example, the translated negative tweet *mom woke me up 30 minutes ago saying common common you have to give your sister a ride, guess who is still waiting?* or the positive *lets listen to keaton henson and eat shawarma*. This is an open problem in the literature of sentiment analysis (Liu, 2012).

Sentiment word not in the lexicon: SenZi did not match 6% of the unclassified sentiment words.

Word Ambiguity: We identified 5% of the wrongly classified words as ambiguous. As mentioned previously (in Section 2.2), word ambiguity is one of the Arabizi NLP challenges generated by the transcription of Arabic in Latin script.

Wrong negation: Classifying negated sentiment words accurately requires more effort than

Table 6: Challenges of Arabizi Sentiment Analysis

	Sentiment form not in the lexicon	Sentiment word is in English	Neutral word classified	Multiword expressions or sarcasm	No sentiment words	Sentiment word not in the lexicon	Word ambiguity	Wrong negation
Positive	37.5%	15%	12.5%	11%	5%	8%	5%	3%
Negative	39%	10%	15.5%	10%	12%	4%	5%	4%

negating sentiment words that are preceded by a negator. A negator may precede or succeed a sentiment word by several words and it may only diminish the sentiment in some cases.

7 Discussion

In this work we focused on an area that has not been explored thoroughly in the literature of sentiment analysis. Arabizi has been proven to be a prominent way of texting on social media among the Arab youth yet there are no public resources to analyse sentiment from this script.

We provided a rigorous explanation of the linguistic challenges for analysing Arabizi text. We created SenZi, the first Lebanese Arabizi sentiment lexicon. We addressed the high degree of lexical sparsity by enriching SenZi with different inflectional and orthographic forms using word embeddings. We achieved an F1-score of 0.72 using a lexicon-based sentiment classification approach.

To the best of our knowledge, there are no other Levant dialect Arabizi datasets or sentiment lexicons to compare our work with. As such, we provided a detailed error analysis to point out the cases that bypassed SenZi.

The annotations carried out to create SenZi and the datasets took place at different times between 2016 and 2018. We tried our best to keep three annotators per task, but in a few cases we had one annotator at hand. However, we tested the annotators with test sets for credibility.

Word embeddings proved to be an excellent technique to expand SenZi, yet 38% of the unmatched sentiment words are forms of SenZi words. Next, we will explore cross-lingual word embeddings with Arabic for further expansion.

Arabizi is a code-switched language, with English appearing the most in Arabizi text from Lebanon. We plan to add English sentiment words to SenZi carefully to handle codeswitching.

Nevertheless, this work is one of the very first attempts to create and evaluate NLP resources for Arabizi sentiment analysis. We created a new sentiment lexicon consisting of 11.3K positive and 13.3K negative words, a sentiment-annotated dataset of 3.4K tweets, and a Facebook corpus of 1M comments. All resources and detailed description are made public and freely accessible on the project's webpage[10].

8 Conclusion

We presented SenZi, the first sentiment analysis lexicon for the Lebanese dialect Arabizi. We built it by translating, annotating, and transliterating other resources to have an initial set of 2K sentiment words. We expanded it to 24.6K sentiment words by importing inflectional and orthographic forms using word embeddings. We evaluated it using a lexicon-based sentiment analysis, achieving an F1-score of 0.72. We finally presented a detailed error analysis to pinpoint its limitations and the challenges that impact the lexicon-based approach for Arabizi sentiment analysis.

Acknowledgements

We thank the annotators and the web developers for their fundamental contributions.

References

Muhammad Abdul-Mageed and Mona T Diab. 2014. Sana: A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis. In *LREC*, pages 1162–1169.

Mariam Aboelezz. 2009. Latinised arabic and connections to bilingual ability. In *Papers from the Lancaster University Postgraduate Conference in Linguistics and Language Teaching*.

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of romanized dialectal arabic. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 30–38.

Mohammed Al-Kabi, Amal Gigieh, Izzat Alsmadi, Heider Wahsheh, and Mohamad Haidar. 2013. An opinion analysis tool for colloquial and standard arabic. In *The Fourth International Conference on Information and Communication Systems (ICICS 2013)*, pages 23–25.

[10] https://project-rbz.kmi.open.ac.uk

Mohammed N Al-Kabi, Amal H Gigieh, Izzat M Alsmadi, Heider A Wahsheh, and Mohamad M Haidar. 2014. Opinion mining and analysis for arabic language. *International Journal of Advanced Computer Science and Applications (IJACSA), SAI Publisher*, 5(5).

Nora Al-Twairesh, Hend Al-Khalifa, and Abdulmalik AlSalman. 2016. Arasenti: Large-scale twitter-specific arabic sentiment lexicons. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 697–705.

Wid H Allehaiby. 2013. Arabizi: An analysis of the romanization of the arabic script from a sociolinguistic perspective. *Arab World English Journal*, 4(3):52–62.

Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. 2014. A large scale arabic sentiment lexicon for arabic opinion mining. In *Proceedings of the EMNLP 2014 workshop on arabic natural language processing (ANLP)*, pages 165–173.

Basis-Technology. 2012. The burgeoning challenge of deciphering arabic chat.

Robert Michael BIANCHI. 2012. 3arabizi-when local arabic meets global english. *Acta Linguistica Asiatica*, 2(1):89–100.

Ann Bies, Zhiyi Song, Mohamed Maamouri, Stephen Grimes, Haejoong Lee, Jonathan Wright, Stephanie Strassel, Nizar Habash, Ramy Eskander, and Owen Rambow. 2014. Transliteration of arabizi into arabic orthography: Developing a parallel annotated arabizi-arabic script sms/chat corpus. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP)*, pages 93–103.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Kareem Darwish. 2014. Arabizi detection and conversion to arabic. *ANLP 2014*, page 217.

Rehab M Duwairi, Mosab Alfaqeh, Mohammad Wardat, and Areen Alrabadi. 2016. Sentiment analysis for arabizi text. In *2016 7th International Conference on Information and Communication Systems (ICICS)*, pages 127–132. IEEE.

Rehab M Duwairi and Islam Qarqaz. 2014. Arabic sentiment analysis using supervised classification. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 579–583. IEEE.

Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. 2014. Foreign words and the automatic processing of arabic social media text written in roman script. *EMNLP 2014*, page 1.

Ramy Eskander and Owen Rambow. 2015. Slsa: A sentiment lexicon for standard arabic. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2545–2550.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Imane GUELLIL, Ahsan Adeel, Faical AZOUAOU, Ala-eddine Hachani, Amir Hussain, et al. 2018. Arabizi sentiment analysis based on transliteration and automatic corpus annotation. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 335–341.

Imane Guellil, Faiçal Azouaou, Mourad Abbas, and Sadat Fatiha. 2017. Arabizi transliteration of algerian arabic dialect into modern standard arabic. In *Social MT 2017/First workshop on Social Media and User Generated Content Machine Translation*.

Yuen Chee Keong, Othman Rahsid Hameed, and Imad Amer Abdulbaqi. 2015. The use of arabizi in english texting by arab postgraduate students at UKM. *The English Literature Journal*, 2(2):281–288.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

M'hamed Mataoui, Omar Zelmati, and Madiha Boumechache. 2016. A proposed lexicon-based sentiment analysis approach for the vernacular algerian arabic. *Res. Comput. Sci*, 110:55–70.

Randa Muhammed, Mona Farrag, Nariman Elshamly, and Nady Abdel-Ghaffar. 2011. Summary of arabizi or romanization: The dilemma of writing arabic texts. In *Jīl Jadīd Conference, University of Texas at Austin*, pages 18–19.

Natalie Sullivan. 2017. *Writing Arabizi: Orthographic Variation in Romanized Lebanese Arabic on Twitter*. Ph.D. thesis.

Taha Tobaili. 2016. Arabizi identification in twitter data. *ACL 2016*, page 51.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Mohammad Ali Yaghan. 2008. "arabizi": A contemporary style of arabic slang. *Design Issues*, 24(2):39–52.

Mining the UK Web Archive for Semantic Change Detection

Adam Tsakalidis[1], **Marya Bazzi**[1,2,3], **Mihai Cucuringu**[1,3],
Pierpaolo Basile[5], **Barbara McGillivray**[1,4]

[1] The Alan Turing Institute, London, United Kingdom
[2] University of Warwick, Coventry, United Kingdom
[3] University of Oxford, Oxford, United Kingdom
[4] University of Cambridge, Cambridge, United Kingdom
[5] University of Bari, Bari, Italy
{atsakalidis, mbazzi, mcucuringu, bmcgillivray}@turing.ac.uk
pierpaolo.basile@uniba.it

Abstract

Semantic change detection (i.e., identifying words whose meaning has changed over time) started emerging as a growing area of research over the past decade, with important downstream applications in natural language processing, historical linguistics and computational social science. However, several obstacles make progress in the domain slow and difficult. These pertain primarily to the lack of well-established gold standard datasets, resources to study the problem at a fine-grained temporal resolution, and quantitative evaluation approaches. In this work, we aim to mitigate these issues by (a) releasing a new labelled dataset of more than 47K word vectors trained on the UK Web Archive over a short time-frame (2000-2013); (b) proposing a variant of Procrustes alignment to detect words that have undergone semantic shift; and (c) introducing a rank-based approach for evaluation purposes. Through extensive numerical experiments and validation, we illustrate the effectiveness of our approach against competitive baselines. Finally, we also make our resources publicly available to further enable research in the domain.

1 Introduction

Semantic change detection is the task of identifying words whose lexical meaning has changed over time. Detecting this temporal variation enables historical and social scientists to study cultural shifts over time (Michel et al., 2011), but it can also have important implications on the performance of models in various NLP tasks, such as sentiment analysis (Lukeš and Søgaard, 2018).

While early theoretical work on semantic change dates back to the previous century (Bloomfield, 1933), the recent availability of historical datasets has made the computational study of the task feasible (Sandhaus, 2008; Michel et al., 2011; Davies, 2012). Past work has demonstrated that semantic change can manifest over decades (Cook and Stevenson, 2010; Mihalcea and Nastase, 2012), years (Yao et al., 2018; Basile and McGillivray, 2018), or even months and weeks (Kulkarni et al., 2015; Tsakalidis et al., 2018).

However, important gaps make progress in the field slow. In particular, there is a relative lack of labelled datasets to study the task over a short time-frame, since most known instances of semantic change took place over centuries or decades. Furthermore, the evaluation of a semantic change detection model is typically performed by manually inspecting a few examples, which can result in unreliable or even non-measurable performance. Finally, on a methodological front, a common practice to measure the semantic shift of words between consecutive time periods is to calculate their displacement error that results from "aligning" word vector representations across these time periods (Hamilton et al., 2016). However, a subset of these words may have actually undergone semantic change and thus trying to align their representations across time is counter-intuitive for the task of semantic change detection, and – importantly – can result in drop in performance. To this end, our work makes the following contributions:

- We release a new dataset for semantic change detection, comprised of word vector representations trained on yearly time intervals of the UK Web Archive (>20TB), along with a list of words with known semantic change, as provided by the Oxford English Dictionary.
- We propose a variant of Procrustes alignment

Proceedings of Recent Advances in Natural Language Processing, pages 1212–1221,
Varna, Bulgaria, Sep 2–4 2019.

for semantic shift detection, trained on an extremely small number of "anchor words" whose meaning is "stable" across time.

- We illustrate the effectiveness of our approach through extensive experimentation, by also proposing the employment of rank-based metrics for evaluation purposes.

2 Related Work

Early work on semantic change detection relied primarily on the comparison of word frequency and co-occurrence patterns between words at different time intervals (Sagi et al., 2009; Cook and Stevenson, 2010; Gulordava and Baroni, 2011), most often representing a single word based on its context (Mihalcea and Nastase, 2012; Jatowt and Duh, 2014; Basile and McGillivray, 2018). Recently, word embeddings have become the common practice for constructing word representations in NLP (Mikolov et al., 2013). A typical process followed in the context of semantic change is to learn the representations of a word over different time intervals and then compute its shift, by employing some distance metric over the resulting representations (Kim et al., 2014; Hamilton et al., 2016; Del Tredici et al., 2018).

A key issue that results from this process is that the comparison of the same word across different time periods becomes impossible, due to the stochastic process of generating the word vectors (e.g., word2vec). To accommodate that, Kim et al. (2014) proposed the initialisation of the word embeddings at time $t+1$ based on the resulting word representations at time t. Kulkarni et al. (2015) learned a linear mapping between the word representations of the nearest neighbours of a word at different time periods. Hamilton et al. (2016) employed Orthogonal Procrustes (Schönemann, 1966) to map the resulting word representations of the whole vocabulary at time t to their corresponding ones at time $t+1$. Another strand of work focuses on generating diachronic word embeddings (Kutuzov et al., 2018), aiming to learn word representations across time (Bamler and Mandt, 2017; Rosenfeld and Erk, 2018; Yao et al., 2018; Rudolph and Blei, 2018). However, these are often hard and slow to train under a massive dataset, such as the UK Web Archive. Similarly, the approach by Kim et al. (2014) does not allow for parallel processing of massive historical collections, since the word vectors at time $t+1$ need to be initialised based on the resulting representations at t. Our work is more closely related to Hamilton et al. (2016). However, aligning the vectors of the whole vocabulary at different times can be noisy and is counter-intuitive for the task of semantic change detection. To mitigate this effect, we propose to learn the alignment based only on a few "stable" (from a semantic point of view) words and apply the same transformation to the full vocabulary, leading to more appropriate alignment and, therefore, to more effective detection of semantically shifted words.

Regardless of the methodological approach, an open issue is the evaluation method of such a model. Owed to the lack of large-scale ground-truth datasets, past work has performed the evaluation either on the basis of detecting only a few word cases of semantic change (Cook and Stevenson, 2010; Gulordava and Baroni, 2011; Del Tredici et al., 2018) or by creating an artificial task, such as word epoch disambiguation (Mihalcea and Nastase, 2012). In this work, we propose instead a rank-based approach that can be employed for the evaluation of a semantic change detection model, even with a few positive examples of words whose lexical semantics have changed.

For more information on semantic change detection, the reader is referred to Tang (2018).

3 Methodology

3.1 Task Definition

Let $[W^{(0)}, ..., W^{(|T|)}]$ be word representations of a common (intersected) vocabulary of $|V|$ terms across $|T|$ consecutive time intervals given by $\{[t, t+1], t \in \{0, \ldots, T-1\}\}$, where each t maps to a given year. Our goal is to *find the words whose meaning has changed the most over each of the consecutive time intervals* $[t, t+1]$ (e.g., between [2000, 2001], [2001, 2002], etc.).

Clear cases of words that have undergone semantic change are difficult to find in a short time period. Furthermore, it has been recently demonstrated that semantic shift is a gradual process and not a sudden and distinctive phenomenon (Rosenfeld and Erk, 2018). Therefore, here we treat our task as a *word ranking problem*, where our aim is to rank the words based on their semantic shift. Importantly, this also enables us to validate the performance of our models in a more robust way as compared to treating the task as a classification problem, since in the latter case the precision score

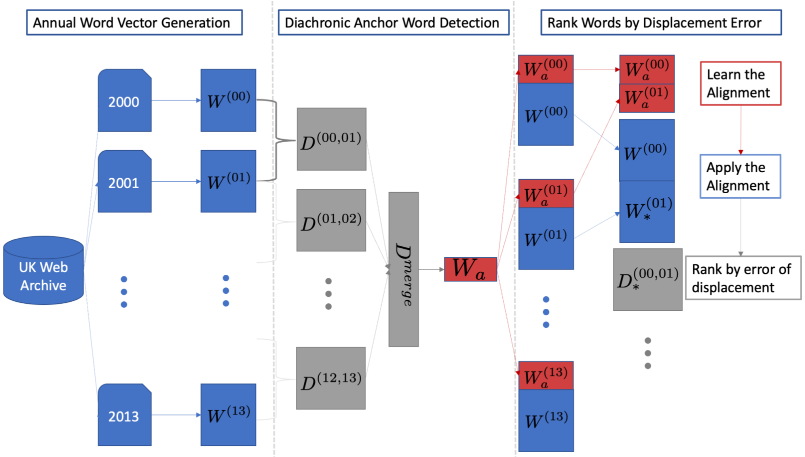


Figure 1: After constructing the word vectors on an annual basis, we learn their pairwise alignments of the resulting word vectors $\{W^{(t)}, W^{(t+1)}\}$. We rank the words based on their average displacement errors across all pairwise alignments in D^{merge} and select the k most stable words as our diachronic anchors W_a. Finally, we learn the alignment of $W_a^{(t+1)}$ based on $W_a^{(t)}$, and apply the same transformation to $W^{(t+1)}$ based on $W^{(t)}$. The words whose meaning has changed the most within $[t, t+1]$ are the ones with the largest displacement error in $D_*^{(t,t+1)}$.

of the positive (semantically shifted) class can be highly biased due to the small number of words belonging to it.

3.2 Our Approach

Figure 1 provides an overview of our approach for ranking the words based on their semantic shift levels. Given some word representations $\{W^{(t)}, W^{(t+1)}\}$ across two consecutive years (see section 4), our goal is to find an optimal way to align $W^{(t+1)}$ based on $W^{(t)}$, so that we can then compute the semantic shift level of a word by means of some distance metric. Typically, this alignment between $W^{(t+1)}$ and $W^{(t)}$ is performed on the complete vocabulary (Hamilton et al., 2016). This implies that the representation of words that have undergone semantic shift are still used as an input to the alignment algorithm, which can result into noisy pairwise alignments (Lubin et al., 2019).

To mitigate this issue, inspired by recent work in word translation (Conneau et al., 2017), we propose the use of a small number of *"anchor words"* to learn the optimal alignment between word representations at two consecutive time periods. Anchor words are defined as words whose lexical semantics remain static over two consecutive time periods. Similarly, *"diachronic anchor words"* correspond to those whose representations remain static across multiple and consecutive time intervals. The detection of these words can lead to more appropriate pairwise alignments of the word vectors, thus facilitating the task of finding semantically shifted words in a more robust fashion.

Anchor Words We formulate our approach on aligning the word vectors $\{W^{(t)}, W^{(t+1)}\}$ across consecutive time periods $[t, t+1]$ on the basis of the Orthogonal Procrustes problem (Schönemann, 1966). Besides past work on semantic change (Hamilton et al., 2016), this approach has been employed in related NLP tasks, such as word translation (Conneau et al., 2017; Ruder et al., 2018). In our case, it finds the optimal transformation of $W^{(t+1)}$ that best aligns it with $W^{(t)}$, by:

$$R = \underset{\Omega;\Omega^T\Omega=I}{\operatorname{argmin}} \left\| W^{(t)}\Omega - W^{(t+1)} \right\|_F. \quad (1)$$

The solution to Eq. 1 can be found via singular value decomposition: $R = UV^T$, where $U\Sigma V^T = SVD(W^{(t+1)}W^{(t)T})$. In our work, we ensure that $W^{(t+1)}$ and $W^{(t)}$ are centered at the origin and that $tr(W^{(t)}W^{(t)T}) = tr(W^{(t+1)}W^{(t+1)T}) = 1$. Finally, we transform $W^{(t+1)}$ as: $W_*^{(t+1)} = W^{(t+1)}R^Ts$, where $s = \sum\Sigma$. We measure the displacement error matrix $D^{(t,t+1)}$ using the cosine distance over the resulting representations $\{W^{(t)}, W_*^{(t+1)}\}$. The k anchor words across $[t, t+1]$ correspond to the k words of $D^{(t,t+1)}$ with the lowest cosine distance (where one can vary the "stability" threshold of the anchor words by varying k).

Diachronic Anchor Words The sets of the detected anchor words may vary between consecutive pairwise time intervals $\{[t, t+1], [t+1, t+2], ...\}$. This contrasts with our intuition of aligning the word vectors based on a few static (from a lexical semantic point of view) words. An intuitive way to accommodate this is to use words that are static throughout a longer period of time. Therefore, to detect "diachronic anchor words", we first perform all of the pairwise alignments and calculate the cosine distances of the words as before. We then concatenate these distances in a $|W|$-by-$|T|$ matrix D^{merge}. The diachronic anchor words correspond to the k words with the lowest average cosine distance in D^{merge}. In Figure 1, we denote their representations as W_a.

Semantic Change Detection We can now use a two-fold process to align the word vectors of two consecutive years $[t, t+1]$: first, we use Procrustes to learn the alignment of $W_a^{(t+1)}$ based on $W_a^{(t)}$, where $W_a^{(i)}$ corresponds to the vector representations of the diachronic anchor words at the time period i. Then, the learned transformation is applied to the representations of the complete vocabulary $W^{(t+1)}$, which are transformed into $W_*^{(t+1)}$. This way, we map the word representations at $t+1$ to the corresponding ones at t in a more robust way. Finally, we calculate the cosine distance matrix $D_*^{(t,t+1)}$ between the word representations in $W^{(t)}$ and $W_*^{(t+1)}$, where lower ranks indicate the index of a word with a higher level of semantic shift. The process is repeated for every pair of consecutive years, by keeping the same set of k diachronic anchor words for each alignment.

4 Data

We employ two datasets in our analysis: (a) the UK Web Domain Dataset 1996-2013 (JISC-UK) is used to learn word representations over different time periods (section 4.1); (b) the Oxford English Dictionary (OED) is used to refine our vocabulary and to build our ground truth – i.e., words that have changed their meaning over time (section 4.2).

4.1 JISC-UK Dataset

The UK Web Domain Dataset 1996-2013 (JISC-UK) contains textual information published in UK-based websites over the time period 1996-2013, thus facilitating the task of semantic change detection in a short-term and fine-grained temporal resolution (Basile and McGillivray, 2018).

Word Vectors Generation The dataset was processed based on previous work by Basile and McGillivray (2018), resulting in over 20TB of textual data. Instead of generating a single vector representation of a word across all years (e.g., by using Temporal Random Indexing (Basile and McGillivray, 2018)), we treated the concatenated content that was published within each year as a single (annual) document $D^{(t)}$, $t \in \{2000, ..., 2013\}$[1]. Following most of the past approaches on semantic change (Kim et al., 2014; Hamilton et al., 2016), we generated word representations by training $|T+1|$ `word2vec` models $m^{(t)}$ (one per year), using Skip-Gram with Negative Sampling and excluding all words appearing less than 1,000 times within a year. Each model was trained for five epochs, using a window size of five words. Finally, we represent every word in year t as a 100-dimensional vector $w_i^{(t)}$, and the resulting matrix of all words as $W^{(t)}$. The resulting vocabulary size per year is shown in Figure 2.

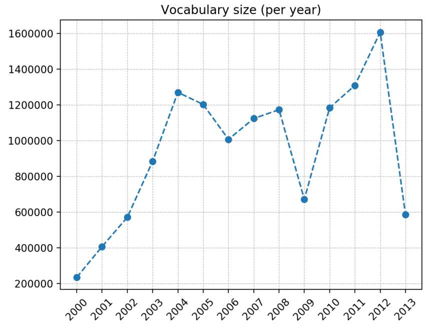


Figure 2: Vocabulary size per year (till May '13), excluding words appearing less than 1,000 times.

4.2 Oxford English Dictionary

The Oxford English Dictionary (OED) is one of the largest dictionaries and the most authoritative historical dictionary for the English language. It records over 250K lemmata along with their definitions, including the year in which each sense was first introduced in the language.

Ground Truth We consider the lemmata that are single words and with definitions whose first appearance in OED is recorded between 2001 and

[1] We excluded the years 1996-1999 owed to the data sparsity observed for these years.

2013 as our ground truth (218 words). Arguably, we expect there to be cases of words whose meaning has changed over time but are not recorded in the OED – i.e., the precision rate of our ground truth is not guaranteed to be 100%. However, this does not affect much our evaluation, since we are not treating our task as a classification problem, but are instead interested in ranking words with known semantic shift in an appropriate manner compared to more semantically stable words (i.e., we are interested in having high recall score of our ground truth, which is guaranteed by the OED).

4.3 Resulting Dataset

As opposed to early work studying the change in word frequency over time to detect semantic change (Michel et al., 2011), here we are interested in detecting words that have undergone semantic change based purely on their context. Therefore, to avoid any bias towards words that have appeared at a certain point in time (e.g., "facebook"), we focus strictly on the words that appear every year, yielding a vocabulary of 168,362 unique words. Finally, we filter out any word that does not appear in OED, due to the lack of ground truth for these words. The resulting dataset that is employed in our modelling is composed of **47,886** unique words that are present in OED and appear at least 1,000 times in every single year between 2000 and 2013, out of which **65** are marked by OED as words that have gained a new meaning after the year 2000[2].

4.4 Empirical Evidence of Semantic Change

Before presenting our experiments, it is important to get some insights on whether (a) semantic change actually occurs in such a limited time period and (b) that our ground-truth showcases this shift, in a qualitative manner.

We begin our analysis by leveraging Procrustes alignment in all possible year-to-year combinations and measure the sum of squared errors of each of the respective alignments. We try this approach on both (a) the intersected vocabulary (approximately 168K words) and (b) the resulting vocabulary by keeping only the words that are mapped to an OED entry (approximately 48K words, see section 4.3).

The results of this process are illustrated in Figure 3. We plot three heatmaps for each case, so that we see if there is an influence stemming from relatively rarely appearing words (when the threshold is set to 1,000). The results demonstrate that the further we move away from the diagonal, the higher the error becomes – note that this is picked up even though there is no notion of "time" in the alignments – indicating that there is a gradual/temporal shift in the meaning of the words, as captured by the context they appear in.

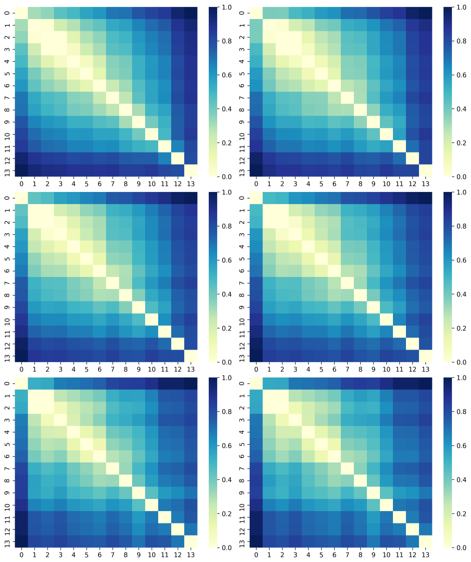


Figure 3: Normalised sum squared errors when aligning the word vectors across different years (2000-2013), using the complete vocabulary (left) and the its intersection with the OED dictionary (right), with different minimum frequency thresholds: 1K (top), 10K (middle) and 100K (bottom).

To further validate the notion of semantic change with respect to our ground truth, we take a closer look at the 65 semantically shifted words that are used in our experiments. The five closest neighbours (by means of cosine similarity) of eight of these words in the years 2000 and 2013 are shown in Figure 4, along with the shift level, measured for each word w and its neighbour n as $cos(w^{(13)}, n^{(13)}) - cos(w^{(00)}, n^{(00)})$. Figure 5 shows the temporal shift in the meaning of the same words from their top-100 neighbours in 2000, using a 3-year moving averaging filter. Both figures demonstrate that the semantic change of our ground-truth is captured through our representations. However, it is also demonstrated that semantic shift is a gradual process and its level may vary across different words. In what follows, we examine the extent to which we can capture this effect using our approach presented in section 3.2.

[2] The resulting dataset is available through: https://github.com/adtsakal/Semantic_Change

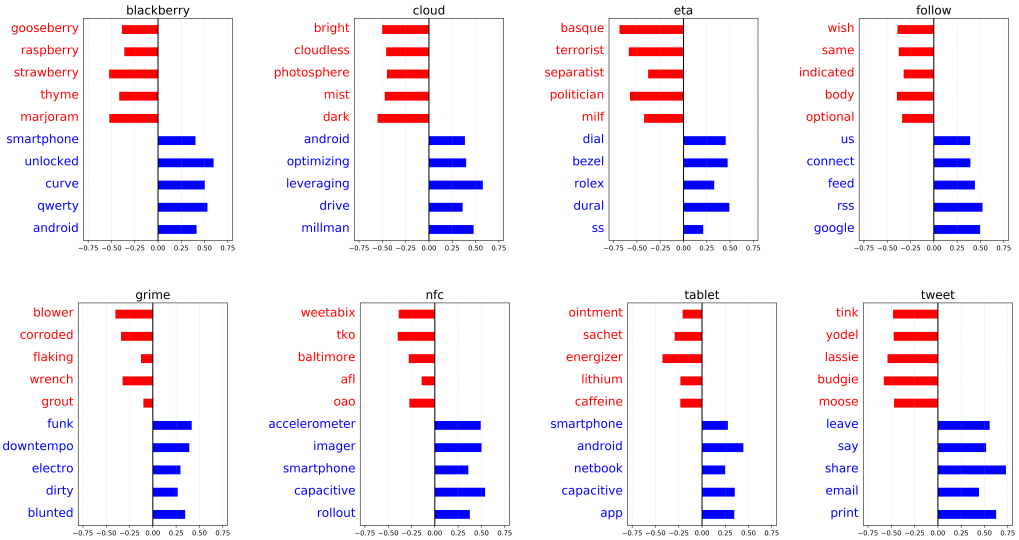


Figure 4: Closest neighbours of words that have undergone semantic shift, at two years (2000, 2013). The bar indicates the shift level of each word towards (away from) each of its neighbours in 2013 (2000).

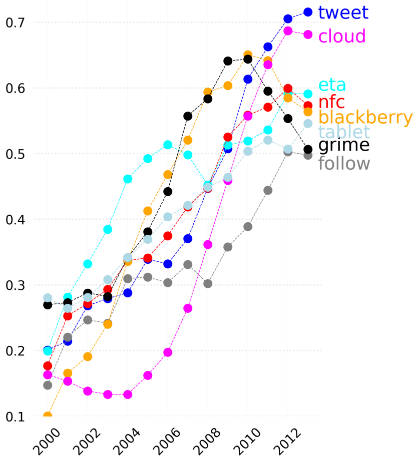


Figure 5: Cosine distance over time between four semantically shifted words (as marked by OED) and their top-100 neighbours in the year 2000.

5 Experiments

5.1 Task Formulation

Given the vector representations of all words across consecutive pairs of years (i.e., {[2000, 2001], ..., [2012, 2013]}), *our aim is to rank the words* based on their respective displacement errors that result after each pairwise alignment. Similarly to past work, we assume that the words corresponding to the highest displacement are those whose semantics has changed the most (Kim et al., 2014; Hamilton et al., 2016). The displacement of a word in a certain interval of a pair of years is calculated on the basis of the cosine distance between its resulting vectors on the first and the second year. Our task is performed on every pair of consecutive years separately.

5.2 Data Split

Experiment 1 We split our data into two sets: (a) in our training set we use most of our data to learn the alignment of the word representations across two different years, by ensuring that none of the 65 words denoted by the OED as words with altered meaning falls in this set (i.e., all of them are considered “static”); (b) we use the rest of our data for evaluation purposes (see next subsection), by ensuring that we include the 65 “changed” words in this set. We experiment with different percentage splits between training and evaluation sets (evaluation set size: [10%, ..., 50%]). This enables us to study the effect of the training set size and the number of diachronic anchor words (see 5.4 below) that are needed to detect semantic change effectively. Due to the small number of “changed” words in the evaluation set, for each percentage split, we perform 40 runs with random splits of the “static” words into the two sets.

Experiment 2 Here we use the complete set of word representations to learn the alignments across the different time intervals, disregarding the split into train/evaluation sets. This enables us to get clearer insights on the performance of the models under a complete setting and study the effect of diachronic anchor words in more detail.

5.3 Evaluation

We propose an alternative, rank-based metric, which can yield robust comparisons across different models, even with a relatively small number of labelled words. Given the final word rankings of an algorithm when applied on a certain pair of years, we denote *the average relative rank of a word whose meaning has changed* (as denoted by the OED) as μ-rank. The value of a single word for this metric lie within the $[0, 1]$ interval, with lower values indicating a better rank produced by the model. The μ-rank of a model is calculated for each of the 13 pairs of years independently and all the results are averaged across the 40 runs, yielding a vector of rank 13. Finally, we consider the average μ-rank score of this vector as our evaluation metric. For all of the models used in Experiment 1, the μ-rank is calculated based on the evaluation set.

5.4 Models

Baselines Our first vanilla approach ($\boldsymbol{PROCR_{100}}$) ranks the words by means of their respective displacement errors (i.e., cosine distance), by learning a single transformation across the whole dataset (Hamilton et al., 2016). For Experiment 1, we also include a second approach ($\boldsymbol{PROCR_{90}}$) which similarly learns the transformation based on the training set and then applies it to the evaluation set.

Our Models We employ two models based on the notion of anchor words: for a given pair of years, $\boldsymbol{PROCR_k}$ first learns an optimal alignment based on the full training set (similarly to $PROCR_{90}$) and then selects the k words with the lowest displacement error of this set to serve as "anchor" words, in order to learn a new alignment based strictly on them; this new transformation is then applied in the evaluation set to yield the final word rankings. This implies that the anchor words are not necessarily the same across all pairs of years. $\boldsymbol{PROCR_{kt}}$ operates in a similar fashion, albeit resolving this drawback through the use of diachronic anchor words: it first learns all of the alignments across the different pairs of years ({[2000, 2001], ..., [2012, 2013]}) and then it selects the k words with the lowest average displacement error across time; finally, it ranks the words in the evaluation set by learning a single transformation for every pair of years based strictly on these anchor words. For both of our models, we experiment with a varying value for k, measured as a % of the size of our training set in Experiment 1. For Experiment 2, we fix k to be the optimal number of words found in Experiment 1.

6 Results

Experiment 1 The results of our models and the baselines are presented in Figure 6. We provide one chart for each evaluation set percentage of the data that was used in our experiments, averaged over the 40 randomised splits we performed.

It becomes apparent that the anchor-based approaches perform clearly better (i.e., they have consistently lower average μ-rank) than those based on the alignment of all of the words – either of the training set, in $\boldsymbol{PROCR_{90}}$, or of both sets, in $\boldsymbol{PROCR_{100}}$. This is because the alignments of the former are based on the representations of words that are indeed stable over time. As we have empirically demonstrated in the previous section, semantic change is a gradual process; thus, aligning words whose representations are not stable across time results into noisy alignments that fail to capture the semantic change of words effectively.

The comparison between the anchor ($\boldsymbol{PROCR_k}$) and diachronic anchor ($\boldsymbol{PROCR_{kt}}$) approaches indicates that the latter performs consistently better. We find that using a very small number of anchor words (0.1% of the training set) yields much better results in almost all cases. Depending on the size of the evaluation set, this number of words ranges from 43 (in the case of 10%) down to 28 words (in the case of 40%). When we further increase the size of the evaluation set (thus decreasing the size of the training set) to 50% of our dataset, we find that using 1% (239) of the words in the training set as anchor words yields slightly better results than using 0.1%. This is because some of the anchor words are placed within the evaluation set, thus the alignment is learned based on weaker anchors, yielding poorer performance. Having a large training set to extract the (diachronic) anchor words from and learn the optimal alignments between their representations across different years is sufficient to overcome this issue.

Finally, while the proposed models outperform the standard practices found in related work, we observe that their performance is still relatively poor: a semantically shifted word is expected to be

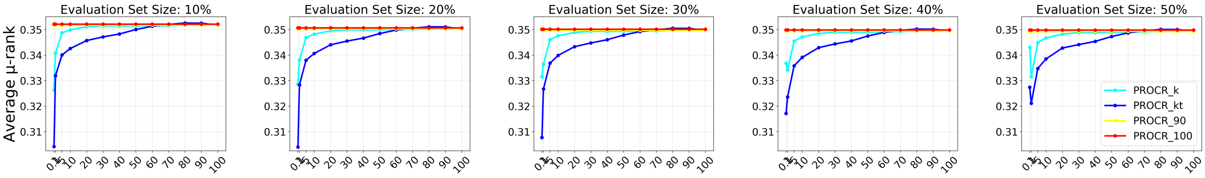


Figure 6: Average μ-rank in Experiment 1 across all runs, using different % of anchor words (x-axis).

ranked close to the top-30% of all of the competing words, with respect to its semantic shift level. This indicates that the task of semantic change detection is rather challenging. Incorporating the temporal dimension of the task is a promising direction for future research in this perspective.

Experiment 2 We present the results when we employ the full dataset to learn the alignments of the $\boldsymbol{PROCR_{100}}$, $\boldsymbol{PROCR_k}$ and $\boldsymbol{PROCR_{kt}}$ models. We fix the percentage of (diachronic) anchor words to be the 47 most stable words (i.e., the top-0.1%). The results are provided in Figure 7 in a per-year basis. $\boldsymbol{PROCR_{kt}}$ performs better on average μ-rank terms (29.48±3.67) against $\boldsymbol{PROCR_k}$ (32.68±4.93) and $\boldsymbol{PROCR_{100}}$ (35.08±4.71), demonstrating again the effectiveness of the alignment based on the diachronic anchor words.

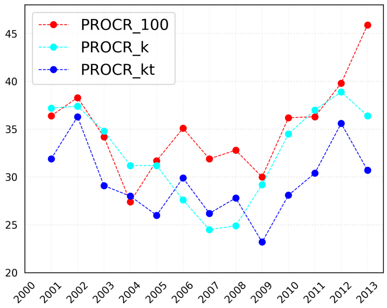


Figure 7: μ-rank of the three models on an annual basis in Experiment 2.

To shed light into the difference between the performance of models employing the anchor and the diachronic anchor words, we calculate the number of anchor words that belong to the set of the diachronic anchors, per year. On average, we find that only 16% (st.dev.: 5.9%) of the annually detected anchor words belong to the latter set. Throughout the pairwise alignments, there are overall 434 unique anchor words detected, from an overall possible of 611. This is owed to the "noisy" selection of anchor words. In Experiment 1, we have demonstrated that aligning the word vectors based on a very small number of anchors performs better. However, the accurate selection of such a small proportion of words can be rather challenging and can vary a lot over consecutive time intervals, due to the noisy nature of the word representations and the alignments themselves. By selecting diachronic anchor words, we are able to filter out this noise, thus yielding more accurate word alignments and tracking the semantic shift of words through time in a more robust way.

7 Conclusion and Future Work

We have introduced a new labelled dataset for semantic change detection. Approaching our task as a word ranking problem, we have proposed an approach to align word representations across different points in time on the basis of a few stable words across time. Through extensive experimentation, we have demonstrated that our approach yields better performance compared to current practices that are based on aligning word representations at different points in time.

An extension to our work is the incorporation of Generalised Procrustes Alignment (Gower, 1975). This will allow us to align the word representations across all years simultaneously and observe the trajectory of each word through time. Furthermore, in our exploratory analysis, we have qualitatively demonstrated that semantic change is a gradual process. Therefore, incorporating the temporal dimension of the task in our approach is a major direction for future work. In particular, we plan to incorporate temporal approaches that are well-suited for the task, such as temporal word clustering and change point detection. Finally, by making our resources publicly available, we hope to facilitate further research in the domain.

Acknowledgments

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and the seed funding grant SF099.

References

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings via skip-gram filtering. *stat* 1050:27.

Pierpaolo Basile and Barbara McGillivray. 2018. Exploiting the Web for Semantic Change Detection. In *International Conference on Discovery Science*. Springer, pages 194–208.

Leonard Bloomfield. 1933. Language.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087* .

Paul Cook and Suzanne Stevenson. 2010. Automatically Identifying Changes in the Semantic Orientation of Words. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Mark Davies. 2012. Expanding horizons in historical linguistics with the 400-million word Corpus of Historical American English. *Corpora* 7(2):121–157.

Marco Del Tredici, Raquel Fernández, and Gemma Boleda. 2018. Short-term meaning shift: an exploratory distributional analysis. *arXiv preprint arXiv:1809.03169* .

John C Gower. 1975. Generalized procrustes analysis. *Psychometrika* 40(1):33–51.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*. pages 67–71.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1489–1501.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*. IEEE Press, pages 229–238.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal Analysis of Language through Neural Language Models. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. pages 61–65.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 625–635.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 1384–1397.

Noa Yehezkel Lubin, Jacob Goldberger, and Yoav Goldberg. 2019. Aligning Vector-spaces with Noisy Supervised Lexicons. *arXiv preprint arXiv:1903.10238* .

Jan Lukeš and Anders Søgaard. 2018. Sentiment analysis under temporal shift. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. pages 65–71.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331(6014):176–182.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. pages 259–263.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 474–484.

Sebastian Ruder, Ryan Cotterell, Yova Kementchedjhieva, and Anders Søgaard. 2018. A Discriminative Latent-Variable Model for Bilingual Lexicon Induction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. pages 458–468.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 1003–1011.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. Association for Computational Linguistics, pages 104–111.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31(1):1–10.

Xuri Tang. 2018. A state-of-the-art of semantic change computation. *Natural Language Engineering* 24(5):649–676.

Adam Tsakalidis, Nikolaos Aletras, Alexandra I Cristea, and Maria Liakata. 2018. Nowcasting the Stance of Social Media Users in a Sudden Vote: The Case of the Greek Referendum. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, pages 367–376.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, pages 673–681.

Cross-Lingual Word Embeddings for Morphologically Rich Languages

Ahmet Üstün **Gosse Bouma** **Gertjan van Noord**

University of Groningen

{a.ustun, g.bouma, g.j.m.van.noord}@rug.nl

Abstract

Cross-lingual word embedding models learn a shared vector space for two or more languages so that words with similar meaning are represented by similar vectors regardless of their language. Although the existing models achieve high performance on pairs of morphologically simple languages, they perform very poorly on morphologically rich languages such as Turkish and Finnish. In this paper, we propose a morpheme-based model in order to increase the performance of cross-lingual word embeddings on morphologically rich languages. Our model includes a simple extension which enables us to exploit morphemes for cross-lingual mapping. We applied our model for the Turkish-Finnish language pair on the bilingual word translation task. Results show that our model outperforms the baseline models by 2% in the nearest neighbour ranking.

1 Introduction

Cross-lingual word embeddings (CLEs) have drawn a lot of attention in recent times. CLE models learn vectors of words in two or more languages and represent them in a shared cross-lingual word embedding space, where words with similar meaning have similar vectors, independent of their language. Most popular approaches for CLEs are mapping-based approaches which are also called offline approaches. These kinds of approaches require only pre-trained monolingual embeddings and a small seed dictionary so that the CLE model learns a mapping that minimizes the distance between word pairs in the seed dictionary to align the pre-trained embedding spaces.

CLE models enable multi-lingual modeling which has direct applications on cross-lingual tasks such as unsupervised machine translation (Lample et al., 2017), and cross-lingual transfer for downstream NLP tasks and low-resource languages. Document classification (Klementiev et al., 2012), information retrieval (Vulić and Moens, 2015), dependency parsing (Guo et al., 2015), and sequence labelling (Zhang et al., 2016) are examples of downstream NLP tasks in which CLEs serve as a source of cross-lingual knowledge.

Although the existing models achieve high performance, agglutinative languages, such as Turkish, Finnish and Estonian, pose a challenge to learn cross-lingual word embeddings due to three main reasons. First, with respect to the monolingual aspects, morphological complexity causes high sparsity which decreases the quality of monolingual embedding spaces (Cao and Rei, 2016; Üstün et al., 2018). Second, in the context of CLEs, the rich morphology causes inaccurate mappings especially for complex words because the existing CLE models cannot access the sub-word level information to align complex words with the correct morphological counterparts. Søgaard et al. (2018) shows that the existing CLE models underperform on rich morphological complexity. On the bilingual dictionary induction task, while the baseline method achieves 82.62% score on English-Spanish, it performs very poorly on English-Finnish (28.01%), English-Estonian (31.45%) and English-Turkish (39.22%). In the Estonian-Finnish dictionary induction experiment in which both languages are morphologically complex, the baseline model performs even worse (24.35%). Last, in addition to this limitation, word-based CLE models are also unable to map an inflected word in the morphologically complex language to a counterpart which corresponds to a phrase in a language with simple morphology.

In this study, we propose a morphologically-sensitive cross-lingual word embedding model[1] in order to overcome the second limitation. We build a cross-lingual model to learn the morpheme representations in the source languages so that a word can be represented through its morphemes in the target space. We design a supervised learning setting as in the baseline model that contains a small bilingual dictionary consisting of morphologically complex word pairs. We perform experiments on Turkish and Finnish as a pair of morphologically complex languages and compare our approach with the baseline models.

2 The Morpheme-Based Alignment Model

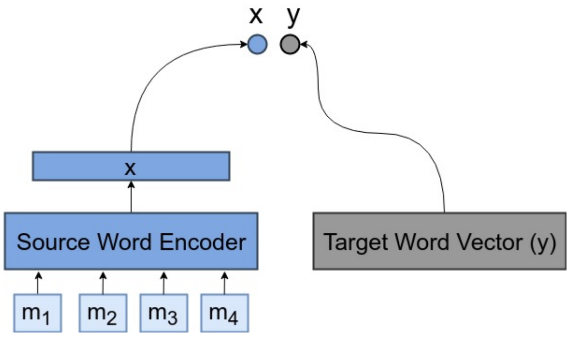


Figure 1: Morpheme-based cross-lingual alignment model that contains a source side word encoder for morphemes. The encoder is trained to learn morpheme representations in the target space

Baselines In this paper, we consider two baseline models. As the first baseline, we employ a simple projection-based CLE method which learns a mapping between embedding spaces by solving the Procrustes problem (Smith et al., 2017; Artetxe et al., 2016). This method first learns a linear transformation matrix to minimize the distance between vectors of word pairs in a seed dictionary by imposing the orthogonality constraint (Gower et al., 2004) and then it uses this matrix to transform the source language embedding space to represent both languages in a shared embedding space. The baseline method is denoted by Procrustes in this paper.

As the second baseline, we use relaxed cross-domain similarity local scaling (RCSLS) (Joulin et al., 2018). RCSLS optimizes the transformation matrix by maximizing the cross-domain similarity local scaling (CSLS) score, instead of minimizing the distance between word pairs in the training dictionary. CSLS is a modification of cosine similarity commonly used in information retrieval. In this way, RCSLS relaxes the orthogonality constraint used in Procrustes according to a retrieval criterion.

Note that for the both baseline models and our model, we use fastText (Bojanowski et al., 2017) to generate monolingual word embeddings. Fasttext represents words as sequence of character n-grams but in many cases this is suboptimal since not all character n-grams are morphemes (Üstün et al., 2018). Besides that the aim of this study is to incorporate morphology into cross-lingual training, whereas fastText is designed for monolingual training.

Morpheme-based Model In the morpheme-based model, we extend the projection-based baseline (Procrustes) in order to exploit sub-word (morpheme) level information for the cross-lingual mapping. Our model starts by splitting all words in the source language into morphemes by using a morphological analyzer. A vector is then computed for each generated morpheme, by using fastText (Bojanowski et al., 2017), as fastText is able to generate a vector for any sub-word since it is based on character n-gram representations.

After the resulting morpheme vectors are inserted into the source vector space, we apply a linear transformation based on the seed dictionary by using the Procrustes method to initialize source and target side vectors in a shared embedding space. Then, our model learns an encoder that encodes each word as a morpheme sequence and transforms them by aligning to their counterparts in the target language. In this way, the resulting encoder learns to represent source side morphemes in the target embedding space.

The model architecture is given in Figure 1. In the figure, x denotes the fixed length word representation generated by the word encoder through morphemes and y represents the target side word embedding. The source encoder is trained to mimic target word embeddings in the bilingual dictionary by minimizing the loss function:

$$L_{align} = dist(x, y) - \lambda(dist(x_c, y) + dist(x, y_c))$$

where (x, y) corresponds to the source and target word embeddings, (x_c, y_c) is a contrastive

[1] Code available at: https://bitbucket.org/ahmetustunn/morphology-sensitive-cle

term. λ[2] controls the effect of the negative samples in the alignment loss. We use the *cosine* similarity for the distance measure.

For the encoder model, following (Conneau et al., 2017a), we use bidirectional LSTMs with max pooling. It encodes the words in both the forward and the backward direction to capture unidirectional information, then it combines the resulting numbers to form a fixed-size vector by selecting the maximum value over each dimension of the hidden units. Figure 2 shows the encoder model. In the figure, each word vector u is computed from morphemes m_n through the bidirectional LSTM encoder.

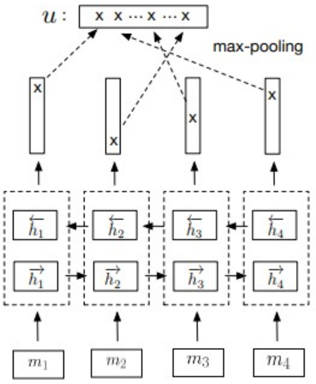


Figure 2: An overview of the word encoder which is used for the source language. It consists of bidirectional LSTMs and a max pooling layer. The inputs of the encoder are the morpheme sequences for each word

3 Morphologically Sensitive Bilingual Lexicon

In order to build a bilingual dictionary for the Turkish and Finnish word pairs, we use the MUSE dataset (Conneau et al., 2017b). Since the MUSE bilingual lexicon consists of translations to or from English for these two languages, we use the intersection of their translation to English. These bilingual lexicons are built with an automatic translation system and the dictionaries handle well the polysemy of words. However, the dictionaries mostly consist of morphologically simple word pairs since English is used as a pivot language.

Considering the morphological variations in these languages, we enriched the dictionary with morphologically complex word pairs. To this end, we first create a lookup table for the lemmas, their inflections and the corresponding morphological features for both languages by using the Universal Dependency Treebanks (Nivre et al., 2016) and the Universal Morphology (Sylak-Glassman, 2016) project.[3] Each word pair in the dictionary is then searched in these lookup tables to list their inflections. The inflected word forms which have the same morphological features for a pair are then added to the bilingual dictionary. Table 1 shows the inflected wordforms found in lookup table for the seed pair *gölge-varjo*.[4] The morphological features that occur in both languages are given in Table 2.

Turkish	Finnish	
gölgem	varjoni	`N;SG;PSS1S`
gölgenin	varjojen	`N;SG;GEN`
gölgelerin	varjoja	`N;PL;PSS2S`
gölgelerde	varjoissa	`N;ESS;PLS`

Table 1: The inflected wordforms with the same morphological features for the word pair *gölge-varjo* which mean *shadow*

Attribute	Morphological Classes
Number	Sing, Plu
Polarity	Neg, Pos
Person	{Pss1,Pss2,Pss3}+{Sg,Pl}
Case	{in,on,at}+{Ess,Abl}, Gen, Prt
Tense	Pst, Prs, Imp
Agreement	P1,P2,P3
Voice	Pass
Mood	Ind, Imp, Cond

Table 2: Morphological features which are common in both Turkish and Finnish

The training dictionary comprises the first 5000 Turkish words and their Finnish counterparts while the test set is composed of the following 1500 word pairs in the lexicon.

4 Experiments

We compare our morpheme-based model with Procrustes (Smith et al., 2017; Artetxe et al., 2016) and relaxed cross-domain similarity local scaling (RCSLS) (Joulin et al., 2018), as explained in Section 2.

[2] Following Conneau et al. (2018), we set λ to 0.25

[3] During the preprocessing step, the default morphological features which are language specific are removed from the datasets.

[4] Both words mean *shadow* in English

Model	NN	CSLS
Turkish-Finnish (TR-FI)		
Procrustes	16.54	17.89
RCSLS	18.26	**21.06**
Our model	**20.35**	20.40
TR-FI on English		
Procrustes	12.72	14.89
RCSLS	15.10	17.05

Table 3: Bilingual word translation performance of the models at P@1 (%). First three rows show the results after training with Turkish-Finnish morphologically sensitive seed dictionary. The last two rows present the results when English is used as a pivot language.

Evaluation Task In order to evaluate the models, we used the bilingual word translation task. Bilingual word translation has become the standard evaluation task for mapping-based CLE models. Given a shared embedding space which is learned by a CLE model, the task is to translate source language words to the target language by retrieving a word in the target language. As the retrieval criterion, either nearest neighbor search (NN) or cross-domain local scaling (CSLS) can be used.

Implementation Details Our evaluation comprises Turkish and Finnish which are both morphologically complex languages. We use the l_2-normalized fastText word and morpheme vectors (Bojanowski et al., 2017) trained on Wikipedia for these languages. We initialize source side embeddings with a linear transformation defined by a Procrustes operation based on the seed dictionary. In order to split words into morphemes, we use the Zemberek toolkit (Akın and Akın, 2007)[5] for Turkish and the Omorfi project (Pirinen, 2015)[6] for Finnish. Both morphological analyzers are rule-based and run with high accuracy. All models are trained with the same seed dictionary and evaluated on the same test set. We evaluated the model by the scores of precision at rank 1 (P@1) so that the results can be morphologically sensitive.

5 Results

Table 3 shows the results on the bilingual word translation performance of the models for the Turkish-Finnish language pair. According to the CSLS scores, RCSLS (Joulin et al., 2018) outperforms our models by a slight margin (0.66%). This is expected because the RCSLS model is explicitly designed to maximize the CSLS objective which causes better performance on the bilingual word translation task according to the CSLS score. However, according to nearest neighbor ranking, our model displays the strongest performance compared to Procrustes and RCSLS with a 2.09% score difference. Table 5 show examples of the nearest neighbour predictions of different models including our model.

We also run Procrustes and RCSLS on the Turkish-English and Finnish-English language pairs so that all three languages share the monolingual English embedding space. We use the MUSE (Conneau et al., 2017b) training dictionaries for both language pairs. In this setting, both Procrustes and RCSLS perform worse on Turkish-Finnish bilingual word translation suggesting that a third language (as a pivot language) does not provide benefit for word translation across morphologically rich language pairs even if it has high-quality word vectors. As our model requires the translations of inflected (morphologically complex) words in the target language, we can not run our model on Turkish-English or Finnish-English pairs because the translations mostly correspond to phrases instead of words.

Model	Spearman
Morph2Vec (Üstün et al., 2018)	**52.90**
Our model	42.05
Fasttext (Bojanowski et al., 2017)	20.80

Table 4: The comparison of the Spearman correlation between human judgments and the word similarities obtained by computing the cosine similarity between the learned word embeddings for Turkish.

Monolingual Impact Similar to the RCSLS, our model changes the cosine distance between word vectors in the same language, that is, it also has an impact on the monolingual embedding space. We evaluate this impact on the Turkish morphologically complex wordlist (Üstün et al., 2018). Results are given in the Table 4.

The Morph2Vec model (Üstün et al., 2018) learns a morpheme-based encoder which is monolingually trained on the large Turkish wordlist which consists of 100K unique words. Although our model is trained on 5K Turkish-Finnish word

[5] https://github.com/ahmetaa/zemberek-nlp

[6] https://github.com/flammie/omorfi

No	Source Word *(Turkish)*	Target Translations *(Finnish)*		
		Procrestus	RCSLS	Our Model
1	öptüm	suutelit *(you kissed)*	suutelen *(I kiss)*	**suitelin** ***(I kissed)***
2	aileler	perhe *(a family)*	**perheet** ***(families)***	**perheet** ***(families)***
3	zamanımız	aikani *(my time)*	aikani *(my time)*	**aikamme** ***(our time)***
4	acemilerden	aloittelijoilla *(in the beginners)*	aloittelijasta *(from the beginner)*	**aloittelijoilta** ***(from the beginners)***
5	makineler	**koneet** ***(machines)***	**koneet** ***(machines)***	koneissa *(in machines)*

Table 5: Examples comparing the translations of different models which also includes the glosses in English. Bolding indicates the correct translation. In Examples 1-4, our model predicts correct word considering the morphological structure but in the Example 5, our model gives wrong translation.

pairs, it improves the monolingual quality of Turkish word vectors for morphologically complex words. The reason behind this impact is that our model also changes the cosine similarity among Turkish word vectors according to morphologically sensitive cross-lingual signals, during the cross-lingual transformation. However, the Morph2Vec model still outperforms our model by a high margin. The results demonstrate that even if training a morpheme-based encoder on cross-lingual word pairs improves the monolingual embedding quality, the same training strategy still performs substantially better on a monolingual wordlist.

Error Analysis Here we study the errors produced by our model on Turkish-Finnish word pairs. Although our model is motivated by morphology, a small portion of the wrong translations is caused by the prediction of wrong inflections of a correct root word. The model translates the Turkish word *santralin* (*of the power plant*) as *voimalaa* (*in the power plant*) instead of *voimalan*. However, the majority of errors have incorrectly translated root words with correct inflections. These observations can suggest two shortcomings. Firstly, our model over-focuses on morphemes so that in some cases it lost the meaning of the content word. Secondly, especially for the distant language pair, some morphological features have different meanings which depend on the sentence syntax and contextual meaning, even if they have the same label. This issue could be alleviated by modeling and processing sentence-level context.

Limitations Similar to the baseline models, the main limitation of our model is that it can not generate multi-word expressions such as phrases on the target side, although our model is able to represent a sequence of strings in the source encoder. However, a morphologically complex word in the source language such as Turkish or Finnish, in most cases corresponds to a phrase, containing more than one word, in morphologically simple target languages such as English. For this reason, our model does not have any direct benefit for the morphologically simple languages and this issue will be the focus of follow-up studies.

Another limitation is that, our model requires a morphological segmenter to split words into morphemes. A simple solution for this could be to employ an unsupervised morphological segmenter which is commonly used in the literature such as Morfessor (Creutz and Lagus, 2005).

6 Conclusion

In this work, we extend the simple mapping-based cross-lingual embedding (CLE) model to learn a morphology-sensitive transformation between embedding spaces for morphologically rich language pairs. We start with the baseline transformation to initialize the source and target embedding spaces and then our model learns an encoder based on morphological segments in the source side and their counterparts in the target space. Thus, the transition matrix which is computed to produce a shared cross-lingual embedding space, is learned through morpheme representations and their composition in the source language.

We evaluated our model on the bilingual word translation task and compare our results with Procrustes and RCSLS (Joulin et al., 2018) scores. Results show that our morpheme-based cross-lingual embeddings model learns slightly better alignments for complex word pairs for languages having rich morphology compared to the baseline models. In this work, we have made the first step towards the comprehensive evaluation of CLE models according to the morphology of languages, however, our evaluation is limited to the bilingual word translation task. For further analysis, we are planning to evaluate our model on other language pairs which consists of both morphologically complex and simple languages and on downstream NLP tasks such as POS tagging and dependency parsing.

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for turkic languages. *Structure*, 10:1–5.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. *arXiv preprint arXiv:1606.02601*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017b. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology Helsinki.

John C Gower, Garmt B Dijksterhuis, et al. 2004. *Procrustes problems*, volume 30. Oxford University Press on Demand.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2984.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *Proceedings of COLING 2012*, pages 1459–1474.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Tommi A Pirinen. 2015. Omorfifree and open source morphological lexical database for finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 313–315.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (unimorph schema). Technical report, Technical report, Department of Computer Science, Johns Hopkins University.

Ahmet Üstün, Murathan Kurfalı, and Burcu Can. 2018. Characters or morphemes: How to represent words? In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 144–153.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), volume 2, pages 719–725.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag–multilingual pos tagging via coarse mapping between embeddings. In *Proceedings of NAACL-HLT*, pages 1307–1317.

It Takes Nine to Smell a Rat:
Neural Multi-Task Learning for Check-Worthiness Prediction

Slavena Vasileva
Sofia University
slav.vasileva@gmail.com

Pepa Atanasova
University of Copenhagen
pepa@di.ku.dk

Lluís Màrquez
Amazon Core ML
lluismv@amazon.com

Alberto Barrón-Cedeño
Università di Bologna
a.barron@unibo.it

Preslav Nakov
Qatar Computing Research Institute, HBKU
pnakov@hbku.edu.qa

Abstract

We propose a multi-task deep-learning approach for estimating the check-worthiness of claims in political debates. Given a political debate, such as the 2016 US Presidential and Vice-Presidential ones, the task is to predict which statements in the debate should be prioritized for fact-checking. While different fact-checking organizations would naturally make different choices when analyzing the same debate, we show that it pays to learn from multiple sources simultaneously (PolitiFact, FactCheck, ABC, CNN, NPR, NYT, Chicago Tribune, The Guardian, and Washington Post) in a multi-task learning setup, even when a particular source is chosen as a target to imitate. Our evaluation shows state-of-the-art results on a standard dataset for the task of check-worthiness prediction.

1 Introduction

Recent years have seen the explosion of fake news, rumors, false claims, distorted facts, half-true statements, and propaganda, which are spreading primarily in social media, but also via standard news broadcasters. This trend became particularly evident during the 2016 US Presidential campaign, which was the turning point that attracted wide public attention to the problem. By then, a number of organizations, e.g., FactCheck[1] and Snopes[2] among many others, launched fact-checking initiatives. Yet, this proved to be a very demanding manual effort, and only a relatively small number of claims could be fact-checked. Thus, it is important to prioritize what to check.

The task of detecting check-worthy claims has been recognized as an important stage in the process of fully automatic fact-checking. According to Vlachos and Riedel (2014) this is a multi-step process that (*i*) extracts statements to be fact-checked, (*ii*) constructs appropriate questions, (*iii*) obtains the answers from relevant sources, and (*iv*) reaches a verdict using these answers. Hassan et al. (2015a) presented a similar vision, and in a follow up work they made check-worthiness an integral part of an end-to-end fact-checking system Hassan et al. (2017).

Here, we approach the problem of mimicking the selection strategy of several renowned fact-checking organizations such as PolitiFact, FactCheck, ABC, CNN, NPR, NYT, Chicago Tribune, The Guardian, and The Washington Post. An important characteristic of this setup is that, perhaps due to editorial policies, fact-checking organizations often select different claims for the same text, with little overlap in their choices (see Tables 1 and 2). Yet, it has been previously shown that it might be beneficial to learn from the union of the selections by multiple fact-checking organizations (Gencheva et al., 2017). Thus, we propose a multi-task deep learning framework, in which we try to predict the choice of each and every fact-checking organization simultaneously. We show that, even when the goal is to mimic the choice of one particular fact-checking organization, it is beneficial to leverage on the choices by multiple such organizations. The evaluation results on a standard dataset show state-of-the-art results.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the used dataset. Section 4 describes our method and features. Section 5 presents the experiments and the evaluation results. Finally, Section 7 concludes and points to some possible directions for future work.

[1] http://www.factcheck.org/
[2] http://www.snopes.com/

Proceedings of Recent Advances in Natural Language Processing, pages 1229–1239,
Varna, Bulgaria, Sep 2–4 2019.

2 Related Work

The proliferation of false information has attracted a lot of research interest recently. This includes challenging the truthiness of news (Brill, 2001; Hardalov et al., 2016; Potthast et al., 2018), of news sources (Baly et al., 2018, 2019), and of social media posts (Canini et al., 2011; Castillo et al., 2011; Zubiaga et al., 2016), as well as studying credibility, influence, bias, and propaganda (Ba et al., 2016; Chen et al., 2013; Mihaylov et al., 2015; Kulkarni et al., 2018; Baly et al., 2018; Mihaylov et al., 2018; Barrón-Cedeño et al., 2019; Da San Martino et al., 2019; Zhang et al., 2019).

Research was facilitated by shared tasks such as the SemEval 2017 and 2019 tasks on Rumor Detection (Derczynski et al., 2017; Gorrell et al., 2019), the CLEF 2018 and 2019 CheckThat! labs (Nakov et al., 2018; Elsayed et al., 2019b,a), which featured tasks on automatic identification (Atanasova et al., 2018, 2019) and verification (Barrón-Cedeño et al., 2018; Hasanain et al., 2019) of claims in political debates, the FEVER 2018 and 2019 task on Fact Extraction and VERification (Thorne et al., 2018), and the SemEval 2019 task on Fact-Checking in Community Question Answering Forums (Mihaylova et al., 2019), among others.

The interested reader can learn more about "fake news" from the overview by Shu et al. (2017), which adopted a data mining perspective and focused on social media. Another recent survey (Thorne and Vlachos, 2018) took a fact-checking perspective on "fake news" and related problems. Yet another survey was performed by Li et al. (2016), and it covered truth discovery in general. Moreover, there were two recent articles in *Science*: Lazer et al. (2018) offered a general overview and discussion on the science of "fake news", while Vosoughi et al. (2018) focused on the proliferation of true and false news online.

The first work to target check-worthiness estimation, i.e., predicting which sentences in a given input text should be prioritized for fact-checking, was the ClaimBuster system (Hassan et al., 2015b). It is trained on data that was manually annotated by students, professors, and journalists, where each sentence was marked as *non-factual*, *unimportant factual*, or *check-worthy factual*. The system used an SVM classifier and features such as sentiment, TF.IDF representations, part-of-speech tags, and named entities.

In our previous work (Gencheva et al., 2017; Jaradat et al., 2018), we used debates from the 2016 US Presidential Campaign and fact-checking reports by professional journalists; we use this same dataset here. Beside most of the features borrowed from ClaimBuster, our model paid special attention to the context of each sentence. This includes whether it is part of a long intervention by one of the debate participants and its position within such an intervention. We predicted both (*i*) whether any of the fact-checking organizations would select the target sentence, and also (*ii*) whether a specific fact-checking organization would select it. There was also a lab on fact-checking at CLEF 2018 and 2019 (Atanasova et al., 2018, 2019), which was partially based on a variant of this data, but it focused on one fact-checking organization, unlike our multi-source setup here.

Patwari et al. (2017) also focused on the 2016 US Election campaign. Their setup asks to predict whether any of the fact-checking organizations would select the target sentence. They used a boosting-like model that takes SVMs focusing on different clusters of the dataset and the final outcome is considered as that coming from the most confident classifier. The features considered range from LDA topic-modeling to part-of-speech (POS) tuples and bag-of-words representations.

Other claim monitoring tools include FactWatcher (Hassan et al., 2014) and DisputeFinder (Ennals et al., 2010b). FactWatcher classifies claims as situational facts, one-of-the-few, or prominent streaks. It checks whether a new text triggers some of the three types of claims, treating the sentences in the text as sequential data. DisputeFinder mines the Web for already-verified claims. Both maintain a growing database of facts and known claims.

Beyond the document context, it has been proposed to mine check-worthy claims on the Web. For example, Ennals et al. (2010a) searched for linguistic cues of disagreement between the author of a statement and what is believed, e.g., "*falsely claimed that X*". The claims matching the patterns would then go through a classifier. This procedure can be used to acquire a dataset of disputed claims.

Given a set of disputed claims, Ennals et al. (2010b) looked for new claims on the Web that entail the ones that have already been collected. Thus, the task can be reduced to recognizing textual entailment (Dagan et al., 2009).

de Marneffe et al. (2008) also looked for contradictions in text. They tried to classify the contradictions that can be found in a piece of text in two categories —those occurring via antonymy, negation, and date/number mismatch, and those arising from different world knowledge and lexical contrasts. The features that are selected for the task of contradiction detection include polarity, numbers, dates and time, antonymy, factivity, modality, structural, and relational features.

Finally, Le et al. (2016) used deep learning. They argued that the top terms in claim vs. non-claim sentences are highly overlapping in content, which is a problem for bag-of-words approaches. Thus, they used a Convolutional Neural Network, where each word is represented by its embedding and each named entity is replaced by its tag, e.g., *person*, *organization*, *location*.

Unlike the above work, we mimic the selection strategy of *one* specific fact-checking organization by learning to jointly predict the selection choices by *multiple* such organizations.

3 Data

In our experiments, we used the CW-USPD-2016 dataset from our previous work (Gencheva et al., 2017), which can be found on GitHub.[3] It is derived from transcripts of the 2016 US Presidential campaign, and includes one Vice-Presidential and three Presidential debates, all of which were fact-checked by the following nine reputable fact-checking organizations: PolitiFact, FactCheck, ABC, CNN, NPR, NYT, Chicago Tribune, The Guardian, and The Washington Post.

Overall, there are four debates with a total of 5,415 sentences. A sentence is considered check-worthy with respect to a source if that source has chosen to fact-check it. Overall, a total of 880 sentences were fact-checked by at least one source, 191 were selected by two or more sources, 100 by three or more, and only one sentence was chosen by all nine sources, as Table 1 shows. Table 2 shows an example: interventions by Hillary Clinton and Donald Trump from the first US presidential debate. This reflects the disparities in check-worthiness selection criteria. More details about the dataset can be found in (Gencheva et al., 2017).

Selected by # Sources	Number of Sentences	Cumulative Sum
9	1	1
8	6	7
7	5	12
6	19	31
5	26	57
4	40	97
3	100	197
2	191	388
1	**492**	**880**

Table 1: Agreement between the fact-checkers: sentences selected by 1, 2, . . ., 9 of them.

4 Our Multi-Task Learning Model

We approach the task of check-worthiness prediction as a multi-source learning problem, using different sources of annotation over the same training dataset. Thus, we can learn to mimic the selection strategy of each of the individual sources.

Figure 1 shows the architecture of our neural multi-task learning model which, given an input sentence in the context of a political debate, predicts whether each of the nine individual sources (tasks) would have selected it, and whether at least one of them would, which is the special *task ANY*.

The input to our neural network consists of various domain-specific features that have been previously shown to work well for the task of check-worthiness prediction. In particular, from (Hassan et al., 2015b), we adopt TF.IDF-weighted bag of words, part-of-speech tags, the presence of named entities, sentiment scores, and sentence length (in number of tokens). Moreover, from (Gencheva et al., 2017), we further adopt *lexicon features*, e.g., for bias (Recasens et al., 2013), for sentiment (Liu et al., 2005), for assertiveness (Hooper, 1974), and for subjectivity; *structural features*, e.g., for location of the sentence within the debate/intervention; LDA topics (Blei et al., 2003); word embeddings, pretrained on Google News (Mikolov et al., 2013); and discourse relations with respect to the neighboring sentences (Joty et al., 2015). See (Hassan et al., 2015b; Gencheva et al., 2017) for more details about each of these feature types.

After the input layer, comes a hidden layer that is shared between all tasks. It is followed by ten parallel task-specific hidden layers. During training, in the process of backpropagation, each task modifies the weights of its own task-specific layer and also of the shared layer.

[3] http://github.com/pgencheva/claim-rank

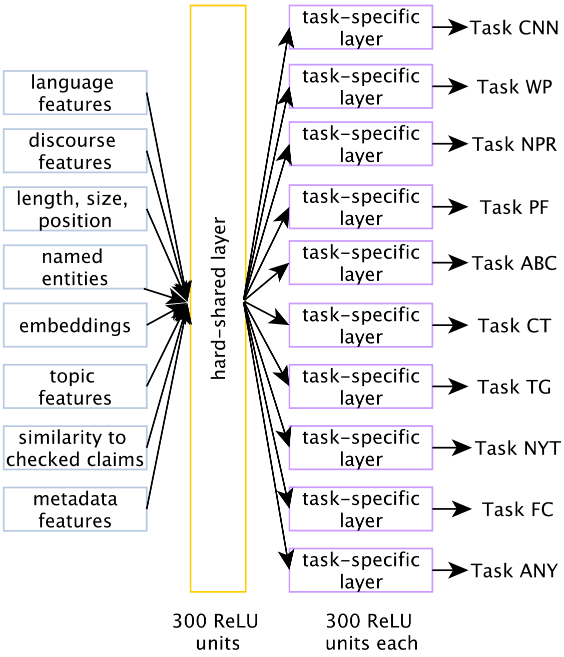


Figure 1: The architecture of our neural multi-task learning model, predicting whether each of the nine individual fact-checking organizations (tasks) would consider this sentence check-worthy and one cumulative source: *task ANY*.

Finally, each task-specific layer is followed by an output layer: a single sigmoid unit that provides the prediction of whether the utterance was fact-checked by the corresponding source. Eventually, we make use of the probability of the prediction to prioritize claims for fact-checking. This kind of neural network architecture for multi-task learning is known in the literature as *hard parameter sharing* (Caruana, 1993), and it can greatly reduce the risk of overfitting.

5 Experiments and Evaluation

As the CW-USPD-2016 corpus contains four debates, we perform 4-fold cross-validation, where each time we leave one debate out for testing, and we train on the remaining three debates. Moreover, in order to stabilize the results, we repeat each experiment three times with different random seeds and we report the average over these three reruns of the system.[4]

In our neural model, we used ReLU units and a shared layer of size 300. For training, we used Stochastic Gradient Descent with Nesterov momentum,[5] iterating for 100 epochs.

Recall that our main objective is to prioritize the claims that should be selected for manual fact-checking, which is best achieved by proposing a ranked list of claims. Thus, we have a ranking task, for which we use suitable information retrieval evaluation measures. In particular, we adopt Mean Average Precision (MAP) as our primary evaluation measure. We further report R-Precision, or R-Pr, and precision at k, or P@k,[6] for $k = \{5, 10, 20, 50\}$. Note that 50 is the approximate number of claims checked by most of the sources for each debate (the exception being PolitiFact, with up to 99 checked claims).

Table 3 presents the evaluation results comparing three models. The first one is a single-task model *singleton* where a separate neural network is trained for each source. The other two are multi-task learning models: *multi* predicts labels for each of the nine tasks, one for each fact-checker, and *multi+any* predicts labels for each of the nine tasks (one for each fact-checker), and also for *task ANY* (as shown in Figure 1). We further compare to the online version of ClaimBuster (Hassan et al., 2015b) and to the *singleton* results reported in (Gencheva et al., 2017) (on the same dataset, with the same cross-validation).[7]

We can see in Table 3 that our *singleton* is comparable and even slightly better than the *singleton* model in (Gencheva et al., 2017), and both outperform the online version of ClaimBuster (Hassan et al., 2015a). We further see that limiting our singleton system to ClaimBuster's features yields a sizable drop in performance. Moreover, for most sources, multi-task learning improves over the singleton models. The results of the multi-task variations that improve over the single baseline are boldfaced. The improvements are consistent across the evaluation measures, but they vary largely depending on the fact-checking source and the evaluation measure.

[4] Having multiple reruns is a standard procedure to stabilize an optimization algorithm that is sensitive to the random seed, e.g., this strategy has been argued for when using MERT for tuning hyper-parameters in Statistical Machine Translation (Foster and Kuhn, 2009).

[5] Using Adam optimizer was faster, converging after only 30 epochs, but it yielded slightly worse results.

[6] See (Buckley and Voorhees, 2000) for a discussion on these evaluation measures.

[7] Note that we could not compare to (Patwari et al., 2017) directly as they used a different dataset. However, they use a small set of basic features that overlap with those of ClaimBuster (Hassan et al., 2015b) to a large extent, and thus we expect that they would perform similarly to ClaimBuster.

Speaker	Total	CT	ABC	CNN	WP	NPR	PF	TG	NYT	FC	Text
Clinton	0	0	0	0	0	0	0	0	0	0	So we're now on the precipice of having a potentially much better economy, but the last thing we need to do is to go back to the policies that failed us in the first place.
Clinton	6	1	1	0	0	1	1	0	1	1	Independent experts have looked at what I've proposed and looked at what Donald's proposed, and basically they've said this, that if his tax plan, which would blow up the debt by over $5 trillion and would in some instances disadvantage middle-class families compared to the wealthy, were to go into effect, we would lose 3.5 million jobs and maybe have another recession.
Clinton	1	1	0	0	0	0	0	0	0	0	They've looked at my plans and they've said, OK, if we can do this, and I intend to get it done, we will have 10 million more new jobs, because we will be making investments where we can grow the economy.
Clinton	0	0	0	0	0	0	0	0	0	0	Take clean energy.
Clinton	0	0	0	0	0	0	0	0	0	0	Some country is going to be the clean- energy superpower of the 21st century.
Clinton	6	1	1	1	1	0	0	1	0	1	Donald thinks that climate change is a hoax perpetrated by the Chinese.
Clinton	0	0	0	0	0	0	0	0	0	0	I think it's real.
Trump	5	1	1	0	1	1	1	0	0	0	I did not.

Table 2: Excerpt from the transcript of the first US 2016 Presidential Debate, annotated by nine sources: Chicago Tribune, ABC News, CNN, Washington Post, NPR, PolitiFact, The Guardian, The New York Times and Factcheck.org. Whether the media fact-checked the claim or not is indicated by a 1 or 0, respectively. The blue sentences are considered as positive in the *any* setting.

One notable exception is NYT, for which the single-task learning shows the highest scores. We hypothesize that the network has found some distinctive features of NYT, which make it easy to predict. These relations are blurred when we try to optimize for multiple tasks at once. However, it is important to state that removing NYT from the learning targets worsens the results for the other sources, i.e. it carries some important relations that are worth modeling.

Table 4 presents the same results but averaged over the nine sources. The first section in Table 4 shows the results for the online version of ClaimBuster (Hassan et al., 2015b), and for the *singleton* and the *task ANY* results in (Gencheva et al., 2017). We can see that our *singleton* model is comparable to the *singleton* and *any* models in (Gencheva et al., 2017), and our multi-task learning models consistently improve over them for all evaluation measures in all but one case.

It is common in neural networks to try to implicitly learn the representations based on word embeddings. We include this as a baseline in the second section in Table 4. The performance of the model that only uses embeddings is in general poor, which suggests that complex feature modeling is necessary for this task; including features that go beyond the current-sentence level. Further feature analysis is included in Table 6.

The third section of Table 4 presents the results for the models of this paper. Again, we can see that multi-task learning yields sizeable improvement over the single-task learning baseline for all evaluation measures.

Another conclusion that can be drawn from this table is that including the task *task ANY* (i.e., whether any of the nine media would select a target) does not help to improve the multi-task model. This is probably due to the fact that this information is already contained in the multi-task model with nine sources.

The last section in Table 4 presents two additional variants of the model: the single-task learning *any* system —which trains on the union of the selected sentences by all nine fact-checkers to predict the target fact-checker only—, and the system *singleton+any* that predicts labels for two tasks: (*i*) for the target fact-checker, and (*ii*) for *task ANY*. We can see that *any* performs comparably to the *singleton* baseline, thus being clearly inferior than the multi-task learning variants. Finally, *singleton+any* is also better than the single-task learning variants, but it falls short compared to the other multi-task learning variants. Including output units for all nine individual media seems crucial for getting advantage of the multi-task learning, i.e., considering only an extra output prediction node for the *task ANY* problem is not enough.

Model	**MAP**	**R-Pr**	**P@5**	**P@10**	**P@20**	**P@50**
ABC						
singleton CB	.057	.061	.050	.038	.056	.050
CB online	.065	.066	.150	.125	.088	.080
singletonG	.059	.068	.050	.050	.100	.060
singleton	.097	.112	.250	.175	.162	.100
multi	**.119**	**.157**	**.333**	**.225**	**.217**	**.122**
multi+any	**.118**	**.160**	**.300**	**.233**	**.229**	**.132**
The Washington Post (WP)						
singleton CB	.051	.053	.050	.033	.046	.048
CB online	.048	.056	.050	.075	.050	.045
singletonG	.102	.098	.200	.175	.113	.080
singleton	.106	.110	.150	.100	.112	.110
multi	**.127**	**.127**	**.350**	**.233**	**.162**	**.123**
multi+any	**.130**	**.129**	**.350**	**.250**	**.171**	***.110***
CNN						
singleton CB	.055	.058	.063	.038	.050	.053
CB online	.082	.096	.150	.125	.088	.085
singletonG	.079	.076	.100	.100	.100	.090
singleton	.087	.091	.250	.150	.121	.090
multi	**.113**	**.132**	***.250***	**.208**	**.183**	**.140**
multi+any	**.109**	**.126**	.167	**.200**	**.167**	**.128**
FactCheck (FC)						
singleton CB	.068	.072	.108	.071	.077	.070
CB online	.081	.213	.150	.125	.100	.115
singletonG	.081	.098	.050	.125	.088	.085
singleton	.084	.114	.117	.125	.088	.100
multi	**.105**	**.136**	**.250**	**.175**	**.146**	**.118**
multi+any	**.117**	.110	**.333**	**.242**	**.196**	**.107**
PolitiFact						
singleton CB	.137	.143	.250	.200	.188	.185
CB online	.154	.213	.200	.300	.238	.210
singletonG	.218	.274	.450	.325	.300	.270
singleton	.201	.278	.250	.250	.262	.262
multi	**.209**	.258	**.400**	**.367**	**.317**	**.270**
multi+any	**.210**	.252	**.500**	**.350**	**.333**	**.272**

Model	**MAP**	**R-Pr**	**P@5**	**P@10**	**P@20**	**P@50**
NPR						
singleton CB	.079	.085	.136	.089	.096	.087
CB online	.144	.186	.200	.225	.225	.180
singletonG	.193	.216	.550	.475	.350	.255
singleton	.175	.195	.250	.250	.283	.228
multi	**.186**	**.210**	**.333**	**.342**	**.300**	**.245**
multi+any	**.180**	**.207**	**.333**	**.283**	.250	.227
The Guardian (TG)						
singleton CB	.066	.075	.110	.070	.070	.066
CB online	.084	.128	.100	.100	.125	.140
singletonG	.121	.156	.250	.225	.200	.155
singleton	.127	.174	.200	.150	.196	.178
multi	**.133**	**.199**	.183	**.175**	**.192**	**.193**
multi+any	**.130**	.159	**.217**	**.175**	**.200**	.167
Chicago Tribune (CT)						
singleton CB	.058	.063	.050	.050	.050	.065
CB online	.053	.032	.050	.050	.038	.065
singletonG	.087	.118	.150	.150	.175	.105
singleton	.079	.110	.100	.100	.125	.075
multi	**.081**	.090	***.100***	**.133**	.104	**.082**
multi+any	**.087**	.087	**.133**	***.100***	.108	**.093**
The New York Times (NYT)						
singleton CB	.080	.084	.138	.094	.100	.088
CB online	.103	.250	.250	.163	.135	.135
singletonG	.136	.178	.250	.225	.188	.135
singleton	.187	.221	.350	.325	.238	.192
multi	.150	.213	.233	.200	.196	.180
multi+any	.147	.197	.200	.167	.158	.162

singleton CB	Singleton only w/ClaimBuster features
CB online	Online version of ClaimBuster
singletonG	Singleton from (Gencheva et al., 2017)
singleton	Trained on the target medium only
multi	Multi-task for nine sources
multi+any	Multi-task for nine sources+any

Table 3: Evaluation results for each of the nine fact-checking sources as a target to mimic. Shown are the results for single-source baselines vs. for multi-task learning with nine and with ten classes. The improvements over the singleton baseline are marked in bold. We further compare to *singleton* that is limited to ClaimBuster's features, to the online version of ClaimBuster (Hassan et al., 2015b), and to *singletonG* results in (Gencheva et al., 2017). The improvements over the latter are underlined.

Model	MAP	R-Pr	P@5	P@10	P@20	P@50
CB online	.090	.138	.144	.143	.121	.117
singletonG	.120	.142	.228	.206	.179	.137
anyG	.128	.225	.194	.186	.178	.153
singleton (embed.)	.058	.065	.055	.055	.068	.072
singleton CB	.072	.077	.106	.076	.081	.079
singleton	.127	.156	.213	.181	.176	.148
multi	**.136**	**.169**	**.270**	**.229**	**.202**	**.164**
multi+any	**.136**	**.159**	**.281**	**.222**	**.201**	**.155**
any	.125	.153	.204	.197	.175	.153
singleton+any	**.130**	.153	**.237**	**.220**	**.184**	***.148***

Table 4: Evaluation results averaged over nine fact-checking organizations (see Table 3 for the unrolled results). We compare multi-task learning to three *singleton* baselines; the improvements are shown in bold. The first section compares to the online version of ClaimBuster (Hassan et al., 2015b), as well as to *singleton* and to *task ANY* results in (Gencheva et al., 2017). The improvements over the latter are underlined. The last section shows the results for two more baselines: *any* and *singleton+any* .

6 Discussion

In this section, we provide deeper insight into the peculiar characteristics of the multi-task model.

Error Analysis First, we perform comparative error analysis, showing both examples of improvement of the proposed *multi* model with respect to the *singleton* as well as some cases where the former fails. The results are shown in Table 5. The first four rows are true positive claims, which were misclassified by the *singleton* model, but were correctly classified by the *multi-task* one. As we can see, the claims were selected for fact-checking by many organizations: between six and eight. This reflects that these instances were certainly check-worthy and the multi-task model correctly spotted them. The observation holds for a prevailing number of all of the new true positives. This is a natural consequence of our neural architecture, where all sources share a hidden layer and tend to learn from the selection criteria of the other sources as well.

Two types of false positive errors occur in rows 5–8. Rows 5 and 6 are predicted by multiple sources that reinforce one another for the wrong guess. We can attribute this to the specifics of the multi-task architecture. On the one hand, the shared layer helps a medium to learn from the selection process of other media. On the other hand, it begins to make more mistakes on claims selected by more media.

N	Type	Tgt	#	Sentence
1	TP	CT	8	**Trump** ▸ It's gone, $6 billion.
2	TP	WP	8	**Trump** ▸ I was against – I was against the war in Iraq.
3	TP	TG	6	**Trump** ▸ You ran the State Department, $6 billion was either stolen.
4	TP	NYT	6	**Pence** ▸ Less than 10 cents on the dollar of the Clinton Foundation has gone to charitable causes.
5	FP	CT	4	**Trump** ▸ Wrong.
6	FP	CT	3	**Trump** ▸ In Chicago, they've had thousands of shootings, thousands since January 1st.
7	FP	CNN	0	**Clinton** ▸ Donald has said he's in favor of defending Planned Parenthood.
8	FP	WP	0	**Trump** ▸ I never met Putin.
9	FN	FC	6	**Clinton** ▸ Donald thinks that climate change is a hoax perpetrated by the Chinese.
10	FN	NYT	4	**Pence** ▸ And Iraq has been overrun by ISIS, because Hillary Clinton failed to renegotiate...
11	FN	NPR	1	**Trump** ▸ China should go into North Korea.
12	FN	NPR	1	**Trump** ▸ We have no growth in this country.

Table 5: Sentences with prediction type (for the *multi* model, with respect to the target medium), the target medium, and total number of media that selected this sentence (#).

On the contrary, rows 7 and 8 show claims that are not check-worthy for any source, but exhibit features such as named entities and negations that typically suggest that the claim might be check-worthy. Finally, rows 9–12 are false negative instances. We have two claims that were fact-checked by several media and two selected by one medium only. The first group indicates that some tasks might try to learn their own features, while the second group shows a possible down side of the multi-task model.

Feature Importance Next, we conduct feature ablation experiments to determine which of the feature groups are most important for the final multi-task model. For this purpose, we remove one feature group at a time from the *multi* model.

Table 6 shows that without the Embedding features the performance of the model drops significantly. They were also the best features in the *singletonG* model of Gencheva et al. (2017). Metadata features are the second most important for the model. An interesting observation is that some of the best-preforming features from *singletonG* are the least contributing to the multi-task model. Such features are *Sim. to prev.* (similarity to previously fact-checked claims), and the linguistic features.

Feature	MAP	R-Pr	P@5	P@10	P@20	P@50
Embeddings	.102	.133	.250	.231	.188	.129
Metadata	.120	.147	.278	.217	.175	.139
Sentiment	.122	.146	.233	.203	.164	.140
Topics	.123	.147	.244	.211	.172	.142
Discourse	.123	.140	.261	.217	.175	.141
NER	.125	.149	.244	.217	.178	.140
Segment size	.125	.149	.256	.211	.172	.139
Position	.125	.143	.261	.219	.193	.138
Linguistic	.126	.150	.250	.208	.190	.151
Contradiction	.126	.149	.250	.203	.174	.142
Lengths	.127	.144	.272	.233	.175	.147
Sim. to prev.	.127	.151	.222	.214	.178	.148

Table 6: Ablation experiments: removing a feature group from the *multi* model, using all nine tasks.

Source Ablation Figure 2 shows ablation results with the *multi* model. A cell at row r and column c shows the performance difference for target c when excluding the target r at training time. For example, in the first row we run the *multi* model neglecting CT in the set of targets. Negative values indicate that removing target r worsens the MAP of target c. Conversely, positive values indicate that removing target r improves MAP for target c. We can observe that the MAP of ABC has dropped by .008, meaning that ABC finds beneficial information from sharing a layer with the CT target. On the contrary, the target FC improves after removing CT, pointing out the presence of conflicts in the learning phase of the shared layer. The largest decrease in MAP is observed in PF after removing CNN, NYT, and NPR. On the other hand, the most significant increase in MAP is in WP after removing NPR and CNN.

7 Conclusion and Future Work

We have presented a multi-task learning approach for estimating the check-worthiness of claims in political debates, and we have further demonstrated its effectiveness experimentally, pushing the state of the art.

In future work, we plan to experiment with more debates. We further plan to go beyond debates, i.e., to general news articles. Moreover, we would like to apply our approach to other languages for which multiple check-worthiness annotations of the same dataset are available.

We plan to try information sources such as the Web (Popat et al., 2017), as well as tweets and temporal information (Ma et al., 2016). We also want to explore other multi-task learning options, e.g., as described in (Ruder, 2017).

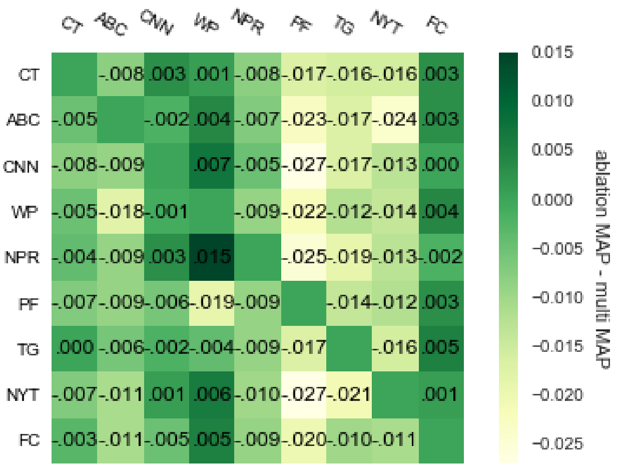


Figure 2: Ablation experiment with the *multi* model. Each row is an experiment removing one target. Each column is the MAP *difference* with respect to the *multi* model for the corresponding target.

It would be interesting to investigate the reasons why the NYT source does not benefit from the multi-task architecture. In order to adapt to this situation with a single model, we plan to experiment with a network with *soft parameter sharing*, e.g., as in (Duong et al., 2015). For example, we could create a chain of layers that back-propagate to the input using only single task targets and then add an auxiliary layer that is shared between the tasks on the side. In this way, the model would be able to turn off the multi-task learning completely for some of the sources. However, training such kind of model might require significantly more training data; semi-supervised training might be a possible solution.

Acknowledgments

We would like to thank the anonymous reviewer, whose constructive feedback has helped us improve the quality of this paper.

This work is part of the Tanbih project,[8] which aims to limit the effect of “fake news”, propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute (QCRI), HBKU and the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL).

[8] http://tanbih.qcri.org/

References

Pepa Atanasova, Lluís Màrquez, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims, Task 1: Check-worthiness. In *CLEF 2018 Working Notes*. CEUR-WS.org, Avignon, France.

Pepa Atanasova, Preslav Nakov, Georgi Karadzhov, Mitra Mohtarami, and Giovanni Da San Martino. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 1: Check-Worthiness. In *CLEF 2019 Working Notes*. CEUR-WS.org, Lugano, Switzerland.

Mouhamadou Lamine Ba, Laure Berti-Equille, Kushal Shah, and Hossam M. Hammady. 2016. VERA: A platform for veracity estimation over web data. In *Proceedings of the 25th International Conference Companion on World Wide Web*. Montréal, Québec, Canada, WWW '16, pages 159–162.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 3528–3539.

Ramy Baly, Georgi Karadzhov, Abdelrhman Saleh, James Glass, and Preslav Nakov. 2019. Multi-task ordinal regression for jointly predicting the trustworthiness and the leading political ideology of news media. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Minneapolis, MN, USA, NAACL-HLT '19, pages 2109–2116.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: Organizing the news based on their propagandistic content. *Information Processing & Management* 56(5):1849 – 1864.

Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Pepa Atanasova, Wajdi Zaghouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims, Task 2: Factuality. In *CLEF 2018 Working Notes*. CEUR-WS.org, Avignon, France.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3(Jan):993–1022.

Ann M Brill. 2001. Online journalists embrace new marketing function. *Newspaper Research Journal* 22(2):28.

Chris Buckley and Ellen M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Athens, Greece, SIGIR '00, pages 33–40.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and the IEEE International Conference on Social Computing*. Boston, MA, USA, SocialCom/PASSAT '11, pages 1–8.

Richard Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the International Conference on Machine Learning*. Amherst, MA, USA, ICML '13, pages 41–48.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the International Conference on World Wide Web*. Hyderabad, India, WWW '11, pages 675–684.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the Internet Water Army: detection of hidden paid posters. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. Niagara, Ontario, Canada, ASONAM '13, pages 116–120.

Giovanni Da San Martino, Seunghak Yu, Alberto Barron-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering* 15(4):i–xvii.

Marie-Catherine de Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Columbus, OH, USA, ACL '08, pages 1039–1047.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Vancouver, Canada, SemEval '17, pages 60–67.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China, ACL-IJCNLP '15, pages 845–850.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Pepa Atanasova, and Giovanni Da San Martino. 2019a. CheckThat! at CLEF 2019: Automatic identification and verification of claims. In *Proceedings of the 41st European Conference on Information Retrieval*. Cologne, Germany, ECIR '19, pages 309–315.

Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019b. Overview of the CLEF-2019 CheckThat!: Automatic identification and verification of claims. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*. Springer, Lugano, Switzerland.

Rob Ennals, Dan Byler, John Mark Agosta, and Barbara Rosario. 2010a. What is disputed on the web? In *Proceedings of the 4th Workshop on Information Credibility*. New York, NY, USA, WICOW '10, pages 67–74.

Rob Ennals, Beth Trushkowsky, and John Mark Agosta. 2010b. Highlighting disputed claims on the web. In *Proceedings of the International Conference on World Wide Web*. New York, NY, USA, WWW '10, pages 341–350.

George Foster and Roland Kuhn. 2009. Stabilizing minimum error rate training. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Athens, Greece, StatMT '09, pages 242–249.

Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Varna, Bulgaria, RANLP '17, pages 267–276.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, MN, USA, SemEval '19, pages 845–854.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Varna, Bulgaria, AIMSA '16, pages 172–180.

Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 2: Evidence and Factuality. In *CLEF 2019 Working Notes*. CEUR-WS.org, Lugano, Switzerland.

Naeemul Hassan, Bill Adair, James T. Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015a. The quest to automate fact-checking. In *Proceedings of the Computation+Journalism Symposium*.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015b. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. Melbourne, Australia, CIKM '15, pages 1835–1838.

Naeemul Hassan, Afroza Sultana, You Wu, Gensheng Zhang, Chengkai Li, Jun Yang, and Cong Yu. 2014. Data in, fact out: Automated monitoring of facts by FactWatcher. *PVLDB* 7:1557–1560.

Naeemul Hassan, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, Aaditya Kulkarni, Anil Kumar Nayak, Vikas Sable, Chengkai Li, and Mark Tremayne. 2017. ClaimBuster: The first-ever end-to-end fact-checking system. *Proc. VLDB Endow.* 10(12):1945–1948.

Joan B. Hooper. 1974. *On Assertive Predicates*. Indiana University Linguistics Club.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 26–30.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. CODRA: A novel discriminative framework for rhetorical analysis. *Comput. Linguist.* 41(3):385–435.

Vivek Kulkarni, Junting Ye, Steve Skiena, and William Yang Wang. 2018. Multi-view models for political ideology detection of news articles. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium, EMNLP '18, pages 3518–3527.

David M.J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science* 359(6380):1094–1096.

Dieu-Thu Le, Ngoc Thang Vu, and Andre Blessing. 2016. Towards a text analysis system for political debates. *LaTeCH 2016* page 134.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.* 17(2):1–16.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion Observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*. New York, NY, USA, WWW '05, pages 342–351.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. New York, New York, USA, IJCAI '16, pages 3818–3824.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Conference on Computational Natural Language Learning*. Beijing, China, CoNLL '15, pages 310–314.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research* 28(5):1292–1312.

Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, MN, USA, SemEval '19, pages 860–869.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. Atlanta, GA, USA, NAACL-HLT '13, pages 746–751.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims. In *Proceedings of CLEF*. Avignon, France, pages 372–387.

Ayush Patwari, Dan Goldwasser, and Saurabh Bagchi. 2017. TATHYA: a multi-classifier system for detecting check-worthy statements in political debates. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. Singapore, CIKM '17, pages 2259–2262.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*. Perth, Australia, WWW '17 Companion, pages 1003–1012.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Melbourne, Australia, ACL '18, pages 231–240.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, ACL '13, pages 1650–1659.

Sebastian Ruder. 2017. An overview of multitask learning in deep neural networks. *CoRR* abs/1706.05098.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.* 19(1):22–36.

James Thorne and Andreas Vlachos. 2018. Automated fact checking: Task formulations, methods and future directions. In *Proceedings of the International Conference on Computational Linguistics*. Santa Fe, NM, USA, COLING '18, pages 3346–3359.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*. New Orleans, LA, USA, NAACL-HLT '18, pages 809–819.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Baltimore, MD, USA, pages 18–22.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359(6380):1146–1151.

Yifan Zhang, Giovanni Da San Martino, Alberto Barrn-Cedeo, Salvatore Romeo, Jisun An, Haewoon Kwak, Todor Staykovski, Israa Jaradat, Georgi Karadzhov, Ramy Baly, Kareem Darwish, and Preslav Nakov James Glass. 2019. Tanbih: Get to know what you are reading. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Hong Kong, China, EMNLP '19.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE* 11(3):1–29.

Deep Learning Contextual Models for Prediction of Sport Events Outcome from Sportsmen Interviews

Boris Velichkov and **Ivan Koychev**
Faculty of Mathematics and Informatics,
Sofia University ”St. Kliment Ohridski”,
Sofia, Bulgaria
bobby.velichkov@gmail.com
koychev@fmi.uni-sofia.bg

Svetla Boytcheva
Institute of Information and
Communication Technologies,
Bulgarian Academy of Sciences,
Sofia, Bulgaria
svetla.boytcheva@gmail.com

Abstract

This paper presents an approach for prediction of results for sport events. Usually the sport forecasting approaches are based on structured data. We test the hypothesis that the sports results can be predicted by using natural language processing and machine learning techniques applied over interviews with the players shortly before the sport events. The proposed method uses deep learning contextual models, applied over unstructured textual documents. Several experiments were performed for interviews with players in individual sports like boxing, martial arts, and tennis. The results from the conducted experiment confirmed our initial assumption that an interview from a sportsman before a match contains information that can be used for prediction the outcome from it. Furthermore, the results provide strong evidence in support of our research hypothesis, that is, we can predict the outcome from a sport match analyzing an interview, given before it.

1 Introduction

1.1 Motivation

The problem of predicting sports results is very challenging and is widely explored in artificial intelligence (AI) (McCabe and Trevathan, 2008). This task requires application of complex algorithms over a huge variety of heterogeneous types of features. Classical decisions are based on statistical and probability models. The AI techniques used to solve this task are based on machine learning (Keshtkar Langaroudi and Yamaghani, 2019) and data mining (Haghighat et al., 2013). This is due to the lack of large datasets with previous results for players and games. Sport is a very dynamic area and players are active for a relatively short period of time. There are also limitations to the predictability of sports outcomes data over a long period of time.

Team sports are more difficult to predict because different team members are selected to play games and even during the game several changes are made to the team, several penalties and injuries to the players that can have a huge impact on the end result. Such predictions in sports rely on many features and time models over a long period of time. In this way, we are able to tackle the task of predicting sports results only in individual sports, such as tennis, boxing, mixed martial arts (MMA) and etc.

1.2 Related Works and Methods

The task of prediction of the sport results can be solved as a classification problem. Naïve Bayes (NB) (McCallum et al., 1998) is the simplest method of classification. It can show good results even for small sets of training data, as is the case with the sports prediction task. The main drawback of the approach is the assumption of attribute independence. Joseph et al (Joseph et al., 2006) presents an application of NB to predict football scores from a database of 76 matches with 30 attributes. The overall average percentage of correct NB learner estimates is 47.86% for the entire database. For a subset of season 1, both with train and test settings from the same season, accuracy increases to 81.58%.

Support Vector Machines (SVM) (Cortes and Vapnik, 1995) are linear classifiers that, like NB classifiers, do not require a large set of training data, making them an appropriate method for predicting sports outcomes. Igiri (Igiri, 2015) presents experimental football prediction results

Proceedings of Recent Advances in Natural Language Processing, pages 1240–1246,
Varna, Bulgaria, Sep 2–4 2019.

with an accuracy of 53.3%. The dataset is based on the English Premier League, with 38 attributes, data for 16 matches in a training set and 15 matches in a test set.

Logistic Regression (Dreiseitl and Ohno-Machado, 2002) can handle the latter problem when the size of the feature space is larger than the size of the training set.

K-nearest neighbour (kNN) (Cunningham and Delany, 2007) A classifier is a proximity classifier that uses distance-based measures for the classification task. Joseph et al (Joseph et al., 2006) presents a kNN application for predicting football scores with an overall average accuracy of 50.58%. They report the highest accuracy of 97.37% for a subset where both training and test sets contain data from the same season.

Other classic classification methods are Random forest, Decision trees, and Rule-based classification. Lock and Nettleton (Lock and Nettleton, 2014) also propose a Random Forest-based approach to predicting winners in the National Football League. Joseph et al.(Joseph et al., 2006) report experiment results for classification for predicting football scores with an overall average accuracy of 45.77% and a maximum accuracy of 78.95%.

Artificial neural networks (ANNs), deep learning, and transfer training are the current preferred approaches to the classification task as they show very high accuracy for large training datasets. An example of applying ANN in predicting football results is presented in (Arabzad et al., 2014) for a set of 2,068 match results records.

The successful application of the classification techniques in tweets for football prediction was presented by (Kampakis and Adamides, 2014) and (Sinha et al., 2013). The model has been learned from about 2 million posts by Tweeter. The maximum accuracy reported for classification is 74.7% (Kampakis and Adamides, 2014).

1.3 Research Hypotheses

We assume that an interview by a sportsman shortly before the match contains information that can be used to predict the outcome of it. In order to extract this information, we first need to understand what the interviewee specifically says about the outcome of the match. Furthermore, this information can be shaded. In addition, we need to capture information that is relevant to the match, but is expressed in a semi-explicit or implicit way, such as health conditions, confidence, psyche, etc. Therefore, we formulate the following research hypothesis: We can predict the outcome of a sport match by analyzing a given pre-match interview using modern NLP and ML methods. To test this hypothesis, we developed the following experiments. First, we learned a model for predicting the outcome of a match without thinking about the interview, using only the available player data such as rank, score in the previous match and ages. We then learned a model for predicting math score solely based on an NLP interview analysis.

2 The Dataset

2.1 Data Collection

For the purpose of our study we collected 50 articles with interviews, in Bulgarian language, conducted with sportsmen shortly before their matches. Interviews are collected online manually and include only individual sports - Boxing, Mixed martial arts (MMA) and Tennis. The idea is to determine if information from them could serve to guess the outcome of the upcoming match - win or lose. For these interviews, we also collected some additional structured data from the official sports rankings, as follows: **Sport** (Boxing:MMA:Tennis – 21:5:24), **Sex** (M:F – 47:3), **IntRank** (Rank of the interviewee), **OppRank** (Rank of the opponent), **IntAge** (Age of the interviewee), **OppAge** (Age of the opponent), **PrevMatch** (The result in the previous match with the same opponent: *W* (The interviewee wins), *L* (The interviewee loses), *N* (There isn't a previous match)) and **Result** (Whether the interviewee *Wins* or *Loses* the match - 56%:44%).

There are no missing values. All structured data and interviews are publicly available[1].

2.2 Data Preprocessing

2.2.1 Structured Data Preprocessing

There is a significant difference in the presentation of player rank and calculations for different sports. For example, tennis rank is a singular number, unlike boxing rank and MMA players are usually presented as a triple ”win – lose – draw”. So, some sort of rank data format was merged. In addition, two derived attributes were added to represent the difference in age and rank of players:

[1] https://github.com/BorisVelichkov/paper

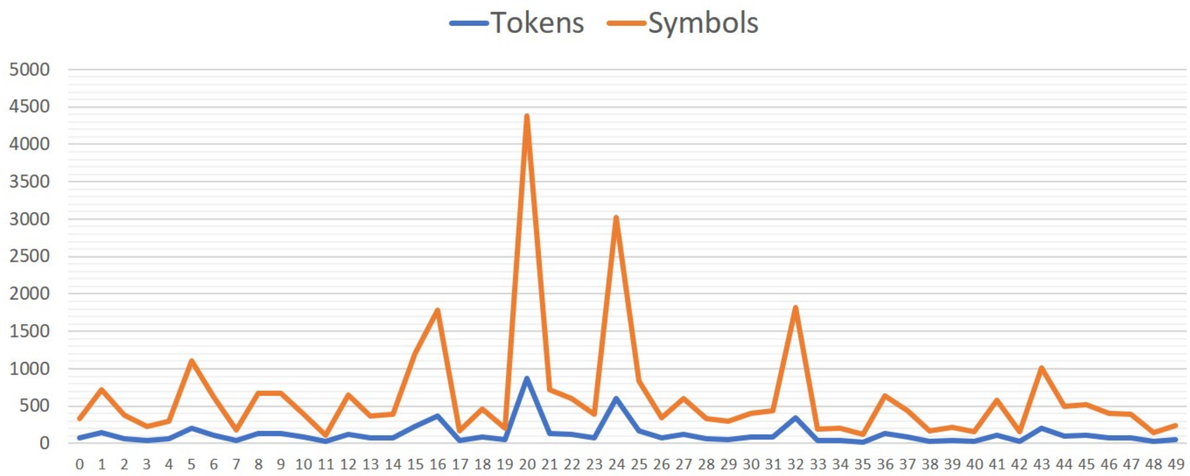


Figure 1: The number of tokens and symbols in the interviews

DiffRank and **DiffAge**. Finally, all data were normalized using a min-max normalization approach.

2.2.2 Unstructured Data Preprocessing

Initial data cleaning and preprocessing was performed. From interviews were removed reporters' comments, leaving only sportsman's quotes/replies. All data are labeled in two categories ”win” and ”lose”, depending on whether the interviewee wins or loses the match discussed in the interview.

Some additional text transformations are applied to the texts for text vectorization. The basic transformations consist of the following steps:

- tokenization - the collection contains 7,799 tokens from 1,469 types;
- all words are converted to lower case;
- all non-Cyrillic words and symbols are removed;
- all punctuation marks are removed;
- all numbers are removed;
- a stemmer is applied - we used the stemmer for Bulgarian language - Bulstem (Nakov, 2003), that provides 3 types of context stemming rules. Based on this, three different datasets are formed, for which we will refer as ”Stem 1”, ”Stem 2” and ”Stem 3”.
- and finally text vectorization based on TFIDF is applied.

The number of words and characters for each interview is shown in Figure 1. The average number of words and characters for interview is respectively 124.52 and 623.8.

3 Experiments

The main purpose of the conducted experiments is to test the assumption that an interview by a sportsman before a match contains information that can be used to predict the outcome of it. Furthermore we would like to explore whether modern pre-trained contextualization models such us a Word2vec (Mikolov et al., 2013) and BERT (Devlin et al., 2018) could help in this task. We also explored how feature selection can affect the accuracy of model building prediction using machine learning (ML) algorithms. Feature selection seems to be an important preprocessing step for many ML algorithms, especially when the attribute space is large, but the examples shown are scarce.

In the experiment we use the following supervised ML algorithms: k-Nearest Neighbors, Support Vector Machines (v-SVM, RBF kernel), Stochastic gradient descent (Squared loss regression, Squared & insensitive classification, Elastic Net regularization, Inverse scaling learning rate), Random Forest (5 trees, 4 attributes per split), Neural Networks (ReLu, 20 hidden layers, Adam solver), Naïve Classifier and Logistic Regression (Regularization type – Ridge L2). Most of the algorithms' parameters are on its default value. The initial setup of some of them was made for structural data so that the algorithms would show their best performance on it. It then remains unchanged throughout the remaining experiments.

For all experiment we used 10-fold cross-validation for models' prediction evaluation.

3.1 Experiments with Structured Data

In the first experiments, we used a structured data set just to learn models that could predict the outcome of the matches. In our general setup of experiments, the prediction accuracy of these models will serve as a baseline for the performance of models learned on an unstructured dataset. The very baseline of the dataset is the prediction of the majority class - 56%.

Model	Accuracy
kNN	0.60
SVM	0.64
SGD	0.60
Random Forest	0.62
Neural Network	**0.66**
Naïve Bayes	0.56
Logistic Regression	0.58

Table 1: Performance of employed ML algorithm using structured data only.

Table 1 presents the experiment results with structured data. The average forecast accuracy is 61%, which is slightly higher than the baseline. Our main goal is not to compare the accuracy achieved with different ML algorithms, but we can mention that algorithms that build more sophisticated models, such as ANN, SVM, and Random Forest, achieve slightly higher accuracy.

3.2 Experiments with Unstructured Data

In the second group of experiments the employed ML algorithms are used on unstructured datasets.

3.2.1 Topic Models

This experiment is based on topic models that have been chosen as a more advanced method than the BOW and TFIDF, as an attempt to capture the basic semantics of the interviews. We experimented with several Topic Modeling Techniques for pre-selected limit 20 for topics sets: Hierarchical Dirichlet Process (HDP) (Teh et al., 2005), Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and Latent Semantic Indexing (LDI) (Hofmann, 2017). The result from experiments are presented in Table 4. We can see that model performance is about the same as that learned on structural data, and HDP slightly outperforms the other two algorithms in this task.

Model	HDP	LDA	LSI
SVM	0.62	0.36	0.58
Random Forest	0.60	**0.60**	0.48
Neural Network	**0.64**	0.56	0.50
NaïveBayes	0.62	0.62	**0.54**

Table 2: Average accuracy of used machine learning algorithms on unstructured data using Topic models HDA, LDA and LSI

3.2.2 Features Selection

To reduce the space dimensionality after text vectorization are applied several features selection techniques. The approach combines the results from the following 6 features selection techniques (Yang and Pedersen, 1997), (Shardlow, 2016), (Saeys et al., 2007):

- Filter methods: (1) χ^2 and (2) Pearson Correlation;
- Wrapper methods: (3) Recursive feature elimination with Logistic Regression;
- Embedded methods: (4) Logistics Regression L1; (5) Random Forest, and (6) LightGBM (Gradient Boosting Machines).

The features, selected from all 6 methods as appropriate, form the first set of features called ”Top 1 features”. Features that are selected through 5 of the 6 appropriate methods form the second feature set, called the ”Top 2 Features”. These two feature categories help to create datasets with filtered features. As a result, there are 3 versions for each dataset: ”All features”, ”Top 1 & Top 2 features” and ”Top 1 features”.

We experimented with the 3 stemmers available. We found no significant effect on the prediction accuracy of the selected stemmer for this task.

Most of the Top 1 features include words that describe in some way the player's condition (“форма” - form, “способен” - ability, “специал” - specialty, “силен” - strong, “здрав” - solid), player expectations and attitudes (“чувств” - feel, “участва” - involved, “нокаутира” - knocked out, “постижени” - achieved, “получи” - received, “оценява” - evaluated, “вярвам” - believe, “вълнува” - excite, “край” - end), information about the training process (“треньор” - trainer, “тренировъч” - training) and many others that are difficult to summarize as a specific category (“деца” - children, “взето” - taken, “софия”

- sofia, and etc.). Interesting is the presence of the words “бокс” - box and “боксов” - boxing in these features because they are describing one exact sport - boxing. 42% of interviews are about boxing matches. Top 2 features include words that are related to pre-match preparation (“подготвя” - prepares, “план” - plan, “процес” - process) and its outcome (“видим” - visible, “обрат” - turning point, “доказва” - proves).

Using feature selection we can reduce features to average 4.80% features with Top 1 features and average 7.49% features with Top 1 & Top 2 features, see Table 3.

Features	Stem 1	Stem 2	Stem 3
All Features	1281	1350	1453
Top 1 Features	65	64	67
Top 2 Features	36	40	34

Table 3: Number of features for Top 1, Top 2 and all features on unstructured data

Model	Stem 1	Stem 2	Stem 3
kNN	**0.60**	**0.60**	**0.62**
SVM	0.48	0.50	0.40
SGD	**0.60**	0.52	0.52
Random Forest	0.50	0.38	0.42
Neural Network	0.58	0.48	0.46
Naïve Bayes	0.52	0.54	0.52
Logistic Regression	0.48	0.52	0.50

Table 4: Accuracy of prediction for the employed ML algorithm using unstructured data and all features.

The experiments with all features (Table 4) show comparable result with those obtained for topic model and unstructured data.

Model	Stem 1	Stem 2	Stem 3
kNN	0.62	0.62	0.62
SVM	0.92	**0.90**	**0.88**
SGD	0.90	0.88	**0.88**
Random Forest	0.78	0.70	0.74
Neural Network	**0.94**	0.78	0.82
Naïve Bayes	0.78	0.88	0.82
Logistic Regression	0.84	**0.90**	0.86

Table 5: Accuracy of prediction for the employed ML algorithm on unstructured data with Top 1 & Top 2 features

Experiments with datasets with feature selection in above described setup (Top 1 & Top 2 features) shows surprisingly good accuracy of prediction, see Table 5. Further reducing the size of feature space (Top 1 features) results in even better forecasting accuracy, see Table 6. Given the large number of features and the relatively small training data set, such an improvement after the selection of features is not unexpected. All experiments were performed with the same parameter settings for ML algorithms as those for the structured dataset.

Model	Stem 1	Stem 2	Stem 3
kNN	0.62	0.62	0.62
SVM	**0.96**	**0.94**	**0.92**
SGD	0.94	0.82	**0.92**
Random Forest	0.84	0.70	0.72
Neural Network	0.82	0.88	0.86
Naïve Bayes	0.84	0.86	0.88
Logistic Regression	0.86	0.86	**0.92**

Table 6: Accuracy of prediction for the employed ML algorithm on unstructured data using Top 1 features

3.2.3 Employing BERT Pre-trained Models

For the text/unstructured dataset, we also used the Google’s pre-trained model BERT (deep bidirectional transformers for language understanding) (Devlin et al., 2018). It has been trained on English Wikipedia and the BookCorpus. For this study we used two of the models: the first one is the default one - *“bert_uncased_L-12_H-768_A-12”*: the second model we used is the Multilingual one - *“bert_multi_cased_L-12_H-768_A-12”*. For our experiments the raw text format is used as an input.

Table 7 presents the results of the experiments performed. We can see that the BERT default model performs significantly better than the multilingual BERT model - over 20 %. At the moment we do not have explanation to such difference. An interesting observation is that the accuracy varies very much across the folds from 0% to 100%. Therefor we run 10 times 10-fold cross validation on random selected folds.

In the context of our main research hypothesis, the two BERT models achieved greater accuracy than the models learned from structured data. This is further evidence that strongly supports our main hypothesis that sportsmen interviews contains information (mostly implicitly presented) that modern NLP techniques and pre-trained models can capture and use it to predict the outcome of a match with very high altitude accuracy.

3.3 Discussion

The results of the experiments performed on structured data alone show that we can build a model

BERT model	Average Accuracy
Default BERT	**0.92**
Multilingual BERT	0.70

Table 7: Performance of BERT models on interviews in Bulgarian language

that achieves a prediction accuracy of 66%. This is significantly above the accuracy of the majority class prediction baseline , which is 56%.

Model based on interviews' content only, achieves an maximum accuracy of 64% for the topic models and 62% for all features. This confirms our initial assumption that the content of the sportsman's interview given before the match contains information that can be used to predict the outcome of the match. In addition, it provides evidence to support our research hypothesis that using modern NLP and ML methods, we can build a classifier that "understands" the text, even possibly caching implicit signals in the text related to the outcome of the match. The interviews show the sportsman's current attitude towards the match and his/her current physical and mental form for the next match. The text contains many moods and shows the sportsman's willingness and readiness to win.

In comparison with our basic model, based on structure data only, we can see that the model build on interviews only, provides approximately the same accuracy. Finally, using feature selection that allows to be captured more significant words for the interviews context, we achieve accuracy 96% for SVM model and Top 1 features, which is an increase in comparison to the previous results. This provides evidence to support the hypothesis that the interview text contains some implicit signals that current NLP methods are able to extract, and that cannot be extracted from structured data.

4 Conclusions

The results of the experiment confirmed our initial assumption that the pre-match sportsman's interview contain information that could be used to predict the outcome of the match. In addition, the results provide strong evidence to support our research hypothesis, that is, we can predict the outcome of a sport match by analyzing an interview given before it using modern NLP and ML methods. More generally, the result of the experiment provides some evidence that current NLP methods are quite cable to "understand" the meaning of text at an almost human level. For feature work we plan to collect a bigger corpora of interviews and conduct further experiments to provide more solid evidences about our research hypotheses and to explore the problem in more details. We also plan to make experiments for collective sports and to combine information from several player interviews, because for such sports is not clear how individual player performance can contribute to the overall match result.

Acknowledgements

The research presented in this paper is partially funded by the Bulgarian National Science Fund, grant DN-02/4-2016 'Specialized Data Mining Methods Based on Semantic Attributes' (IZIDA). We are grateful to anonymous reviewers for useful comments and suggestions.

References

S Mohammad Arabzad, ME Tayebi Araghi, S Sadi-Nezhad, and Nooshin Ghofrani. 2014. Football match results prediction using artificial neural networks; the case of iran pro league. *Journal of Applied Research on Industrial Engineering* 1(3):159–179.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Padraig Cunningham and Sarah Jane Delany. 2007. k-nearest neighbour classifiers. *Multiple Classifier Systems* 34(8):1–17.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Stephan Dreiseitl and Lucila Ohno-Machado. 2002. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics* 35(5-6):352–359.

Maral Haghighat, Hamid Rastegari, and Nasim Nourafza. 2013. A review of data mining techniques for result prediction in sports. *Advances in Computer Science: an International Journal* 2(5):7–12.

Thomas Hofmann. 2017. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*. ACM, volume 51, pages 211–218.

Chinwe Peace Igiri. 2015. Support vector machine—based prediction system for a football match result. *IOSR Journal of Computer Engineering (IOSR-JCE)* 17(3):21–26.

Anito Joseph, Norman E Fenton, and Martin Neil. 2006. Predicting football results using bayesian nets and other machine learning techniques. *Knowledge-Based Systems* 19(7):544–553.

Stylianos Kampakis and Andreas Adamides. 2014. Using twitter to predict football outcomes. *arXiv preprint arXiv:1411.1243* .

Milad Keshtkar Langaroudi and Mohammadreza Yamaghani. 2019. Sports result prediction based on machine learning and computational intelligence approaches: A survey. *Journal of Advances in Computer Engineering and Technology* 5(1):27–36.

Dennis Lock and Dan Nettleton. 2014. Using random forests to estimate win probability before each play of an nfl game. *Journal of Quantitative Analysis in Sports* 10(2):197–205.

Alan McCabe and Jarrod Trevathan. 2008. Artificial intelligence in sports prediction. In *Fifth International Conference on Information Technology: New Generations (itng 2008)*. IEEE, pages 1194–1197.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*. Citeseer, volume 752 (1), pages 41–48.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Preslav Nakov. 2003. Bulstem: Design and evaluation of inflectional stemmer for bulgarian. In *Workshop on Balkan Language Resources and Tools (Balkan Conference in Informatics)*.

Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. 2007. A review of feature selection techniques in bioinformatics. *bioinformatics* 23(19):2507–2517.

Matthew Shardlow. 2016. An analysis of feature selection techniques. *The University of Manchester* pages 1–7.

Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A Smith. 2013. Predicting the nfl using twitter. *arXiv preprint arXiv:1310.6998* .

Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*. pages 1385–1392.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *Icml*. volume 97, page 35.

Exploiting Frame Semantics and Frame-Semantic Parsing for Automatic Extraction of Typological Information from Descriptive Grammars of Natural Languages

Shafqat Mumtaz Virk[1], Azam Sheikh Muhammad[2], Lars Borin[3], Muhammad Irfan Aslam[4], Saania Iqbal[5], and Nazia Khurram[6]

[1,3]Språkbanken, University of Gothenburg, Sweden *{shafqat.virk, lars.borin}@svenska.gu.se*
[2]Department of CS & Eng., Chalmers University of Technology, Sweden, *azams@chalmers.se*
[4]University of Skövde, Sweden, *irfan.aslam@hotmail.com*
[5,6]GIFT University, Pakistan, *sania__iqbal@hotmail.com, nazia.yousaf@gmail.com*

Abstract

We describe a novel system for automatic extraction of typological linguistic information from descriptive grammars of natural languages, applying the theory of frame semantics in the form of frame-semantic parsing. The current proof-of-concept system covers a few selected linguistic features, but the methodology is general and can be extended not only to other typological features but also to descriptive grammars written in languages other than English. Such a system is expected to be a useful assistance for automatic curation of typological databases which otherwise are built manually, a very labor and time consuming as well as cognitively taxing enterprise.

1 Introduction

There are more than 7,000 living languages in the world and grammatical descriptions[1] are available for some 4,000 of these (Seifart et al., 2017). A central concern of the academic discipline of linguistics is to classify languages along different dimensions. The subbranch of linguistics which deals with classification and comparison of languages based on their structural and functional characteristics is known as *linguistic typology* (or *typological linguistics*. In addition to comparing the world's languages, practitioners of linguistic typology aim to explore the distribution of various structural and functional patterns among languages and to explain them in historical and/or universal terms. (Song, 2010)

To achieve its goals, typological linguistics has relied largely on manual reading of available descriptive material about languages for the extraction of pertinent feature values for comparison of these across languages. For example, in their sentence structure, different languages favor different word orders (e.g. subject-object-verb, verb-object-subject, or object-verb-subject, etc). If word order is to be used as one structural feature for language comparison, the word order of all the languages to be compared has to be found out manually by reading available material on those languages. This is doable, if the scope of comparison is to be limited to a few features spanning across a few languages. If one aims to extend the scope from a few to a few hundred features spanning across thousands of languages, the manual reading and comparison strategy seems simply unfeasible. With the availability of large amounts of digital data, and the recent advances in natural language processing, automatic extraction of typological features from grammatical descriptions seems a plausible task.

There have already been a few attempts to automatically extract typological and other linguistic information from descriptive grammars (Virk et al., 2017; Borin et al., 2018). In these studies, the authors have reported simple pattern matching and syntactic parsing based approaches, and have shown that their strategy is useful, yielding reasonable precision and recall values. Even though simple pattern matching based systems are easy to comprehend, implement, and maintain, they require a deep understanding of the rules/patterns which may not be very obvious in certain cases. Further, such systems are heuristics based and also require a large manual effort (Chiticariu et al., 2013). For these reasons the pattern-matching based systems are becoming a less and

[1] Grammatical descriptions are plain text descriptions of various phonological, morphological, and syntactic characteristics of languages.

Proceedings of Recent Advances in Natural Language Processing, pages 1247–1256,
Varna, Bulgaria, Sep 2–4 2019.

less attractive choice and machine learning and big-data based approaches are taking their place.

In this paper, we report a novel methodology and a system for automatic extraction of typological information from descriptive grammars. The system is based on the theory of frame semantics and frame-semantic parsing, and employs a machine learning based approach. Using a set of domain-specific semantic frames, a handful of descriptive grammars are manually annotated with linguistic frames and their associated frame elements. Using these annotations as training data, machine learning models are trained and tested, which are then used to automatically annotate new descriptive grammars. The annotations are subsequently converted into typological feature values using a small rule based module, hence resulting in automatic extraction of typological feature values from descriptive grammars.

Section 2 describes frame semantics, FrameNet, and frame-semantic parsing as a theoretical background, followed by a brief introduction to the the linguistic domain FrameNet (Section 3). The development of a parser for the linguistic domain is outlined in Section 4, leading to a description of the system for automatic extraction of typological features (Section 5).

2 Frame Semantics, FrameNet, and Frame-Semantic Parsing

2.1 Frame Semantics

Frame semantics is a theory of meaning in language introduced by Charles Filmore (Fillmore, 1976, 1977, 1982). The theory is based on the notion that meanings of words can be best understood when studied in connection with the situations to which they belong, and/or in which they may occur.

The backbone of the theory is a conceptual structure called a *semantic frame*, which is a script-like description of a prototypical situation, an event, an object, or a relation. As an example, consider a real life scenario of robbery – a situation in which someone (a perpetrator) wrongs a victim by taking something (goods) from him/her. A structured representation of such a situation is called a *semantic frame*. The participants of the situation (i.e. the perpetrator, the victim, the goods, time, place, manner) are called *frame elements*. Some of the them (the perpetrator, the victim, and the goods) are necessary for the situation to make sense and are called *core frame elements*. Others like the place where the robbery took place, the manner in which it took place are called *non-core frame elements* (see Ruppenhofer et al., 2016 for details). Now, with the availability of a structured representation of the robbery situation, words like *hold up, mug, ransack, rifle, rob, stick up* can be better understood and analyzed.

2.2 FrameNet

The development of a lexico-semantic resource – FrameNet (Baker et al., 1998) – based on the theory of frame semantics was initiated in 1998 for English. In this lexical resource, generally referred to as simply FrameNet or Berkeley FrameNet (BFN), each of the semantic frames has a set of associated words (or *triggers*) which can evoke that particular semantic frame. The linguistic expressions for participants, props, and other characteristic elements of the situations (called *frame elements*) are also identified for each frame. In addition, each semantic frame is accompanied by example sentences taken from naturally occurring natural language text, annotated with triggers, frame elements and other linguistic information.

In the context of deploying FrameNets in NLP applications, BFN and other FrameNets have often been criticized for their limited coverage. A solution to this problem is to develop domain-specific (sublanguage) FrameNets to complement the corresponding general-language FrameNets for particular NLP tasks. In the literature we find such initiatives covering various domains, e.g.: (1) a FrameNet to cover medical terminology (Borin et al., 2007); (2) *Kicktionary*,[2] a soccer language FrameNet; (3) the *Copa 2014* project, covering the domains of soccer, tourism and the World Cup in Brazilian Portuguese, English and Spanish (Torrent et al., 2014).

Because of their perceived usefulness for a variety of purposes, general-language FrameNets have also been developed for a number of other languages including Chinese, French, German, Hebrew, Korean, Italian, Japanese, Portuguese, Spanish, and Swedish.

2.3 Frame-Semantic Parsing

In addition to the annotated examples, FrameNets are also often accompanied by varying amounts of frame-annotated natural running text intended

[2] http://www.kicktionary.de/

both to illustrate particular semantic-frame usages and to demonstrate the utility of frame semantics as a model of meaning in language. One of the uses of such annotated text is to develop automatic frame-semantic parsers, which in turn have proved useful in a number of natural language processing tasks including question answering (Shen and Lapata, 2007), coreference resolution (Ponzetto and Strube, 2006), paraphrase extraction (Hasegawa et al., 2011), machine translation (Wu and Fung, 2009), and information extraction (Surdeanu et al., 2003).

Frame-semantic parsing necessarily involves three basic steps. These are frame identification, frame element identification, and frame-element classification. Consider the annotated sentence shown in Figure 1 to better understand those basic steps of the frame semantic parsing. If the annotation shown is to be done automatically, the first step would be to consider each word of the sentence and check if it evokes a particular frame or not, and disambiguate in case if the candidate word evokes more than one frame. As a result, the word *agrees* (shown in bold) will be recognized as a lexical unit triggering the AGREEMENT frame. This is the frame identification task. Having identified the frame-triggering lexical units and the triggered frame, the next steps are to identify the text segments filling various semantic roles (i.e. frame elements) of the triggered frame. For this task, each word (or combination of words) in the sentence needs to checked for whether it expresses a frame element or not. This is the frame-element identification task. When a particular word or word-combination has been recognized as a frame element, it should be labeled next i.e. frame-element classification. So the frame-element identification and classification tasks will label the text segment *The genitive* as 'Participant_1', *sometimes* as 'Frequency', *noun* as 'Participant_2', *in gender* as 'Grammatical_Category' and *Gondi* as the 'Reference_Language'.

All of these three steps can be formulated as supervised machine learning classification tasks. Gildea and Jurafsky (2002) were the first to report their experiments with an automatic frame-semantic parsing system, and since then there have been a number of studies (Johansson and Nugues, 2008; Swayamdipta et al., 2017; Kabbach et al., 2018) and a shared task (Surdeanu et al., 2008) devoted to exploring and improving the task of frame-semantic parsing.

3 LingFN – a FrameNet for the Linguistic Domain

Linguistics has established a rich set of domain-specific terms and concepts such as *verbs*, *nouns*, *determiners*, *inflection*, *agreement*, *affixation*, etc. Inspired by other domain-specific FrameNets (mentioned in Section 2.2), the development of a FrameNet for the linguistic domain (LingFN) has been previously reported (Malm et al., 2018). LingFN contains two types of frames: the *filler frames* and the *eventful frames*. The former are to cover simple linguistic terms such as *noun*, *verb*, etc. mostly referring to the morpho-syntactic linguistic categories, and the later type covers linguistic processes such as *inflection*, *agreement*, *affixation* etc. Based on the empirical investigations of the usage of those terms and concepts in a large collection of domain-specific data, both types of frames were constructed. Consider again Figure 1 which also shows the structure of the AGREEMENT frame – an eventful of frame. In the linguistic domain, agreement is a phenomenon in which words of a particular morphological category (e.g. nouns) agree with another morphological category for a particular grammatical category (gender, number, etc.). The structure shown in Figure 1 was developed based on the investigations of the usage of the word *agree* within the linguistic corpora. See Malm et al. (2018) for a detailed description of the procedure followed to design frames together with annotated example sentences given to show the realization of those frames in the linguistic domain data.

The current version of LingFN contains a total of 100 frames, 32 frame elements, 360 lexical units, and around 2,800 annotated examples.

For the study reported in this paper, we have restricted ourselves to the frames outlined in Table 1 in addition to the AGREEMENT frame above. These frames will prove useful while automatically extracting values of certain typological feature as will be elaborated in Section 5.

4 The LingFN Parser

This section describes the development of an automatic parser based on LingFN. Treating it as a

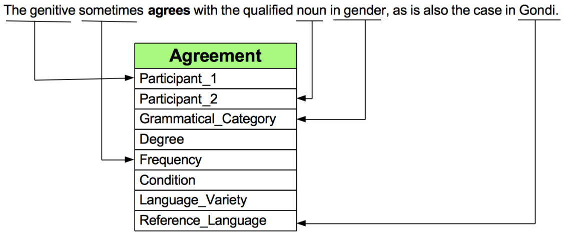


Figure 1: The structure of the AGREEMENT frame

supervised machine learning task, we describe the production of training data, feature selection and training data generation, and the training and testing of machine learning models in the following subsections.

4.1 Data Annotation

A set of 66 grammatical descriptions from the classical *Linguistic Survey of India* (LSI; Grierson, 1903–1927)[3] were annotated with the frames from LingFN described in Section 3.[4]

An online annotation tool from the Brazilian FrameNet Brasil project[5] was used. The annotation process was a collective effort and a number of data annotators were involved in this step. Each data annotator was responsible for 6 documents, and the length of each document is between 90 and 255 sentences. The task for each annotator was to go through the sentences of each document, identify each lexical unit and record the frame triggered by it. Once the frame has been identified, the next task was to identify and label the text segments (if present) of the sentence indicating the frame elements of the frame. Figure 2 shows a screenshot from the web tool used for annotation. As can be seen, the tool provides a layered view with the sentence to be annotated appearing in the top layer. This is followed by two layers, one for the frame and the other for frame elements annotation, for each of the triggered frames. The screenshot shows the annotation of the sentence *the verb agrees in number and gender with the subject* with two frames i.e. the frame VERBAL triggered by the lexical units *verb* (shown in black) with an empty FE layer (since there is no FE to be annotated in the sentence for this frame) and the frame AGREEMENT triggered by the lexical unit *agrees*. The FE layer for the AGREEMENT frame contains the annotations 'Participant_1' (in red) referring to text-segment *the verb*, 'Participant_2' (in blue) referring to the text segment *the subject*, and 'Grammatical_Category' (in green) referring to the text segment *in number and gender*.

1	**Frame Name:** AFFIXATION
	Definition: A frame to capture the phenomena of affixation in linguistics, which is the process of adding a morpheme — or affix— to a word to create either a different form of that word or a new word with a different meaning
	Frame Elements: Stem, Affix, Language_Variety, Reference_Language, Location Frequency, Manner, Purpose, Condition, Position, Degree
	Example Annotation: [An n]$_{\text{Morpheme\_one}}$ is [often]$_{\text{Frequency}}$ [infixed]$_{\text{LU}}$ [after the first vowel of a word]$_{\text{Morphosyntactic\_position}}$, the vowel being also repeated after n .
2	**Frame Name:** SEQUENCE
	Definition: A frame to capture the ordering information of various morphological or syntactical categories
	Frame Elements: Entity_1, Entity_2, Order, Language_Variety, Entities, Condition, Frequency, Reference_Language, Certainty, Data, Data_Translation
	Example Annotation: [Adjectives]$_{\text{Entity\_1}}$ in [Garo]$_{\text{Language\_Variety}}$, [as in Kacha ri]$_{\text{Reference\_Language}}$, [generally]$_{\text{Frequency}}$ [[follow]$_{\text{Order}}$]$_{\text{LU}}$ [the noun they qualify]$_{\text{Entity\_2}}$
3	**Frame Name:** CREATION
	Definition: A frame to capture the phenomena of creation of a morphological or syntactic category from another morphological or syntactic category
	Frame Elements: Created_Entity, Created_From, Process, Degree, Certainty , Language_Variety, Reference_Language, Data, Data_Translation, Condition
	Example Annotation: [Adverbs]$_{\text{formed\_Entity}}$ are often separate words , but are also [frequently]$_{\text{Degree}}$ [formed]$_{\text{LU}}$ from [the corresponding adjective]$_{\text{formed\_From}}$ [by adding hui or ui]$_{\text{Process}}$.

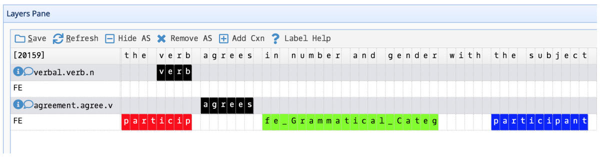
Table 1: Targeted frames

Table 2 shows some statistics of the produced annotated data. Further details, such as inter-annotator agreement, etc., are beyond the scope of

[3] The LSI presents a comprehensive survey of the languages spoken in South Asia conducted in the late nineteenth and the early twentieth century by the British government. It has descriptions of various phonological, morphological, and grammatical features of about 723 linguistic varieties spoken in the nineteenth-century British-controlled India (modern Pakistan, India, Bangladesh, and parts of Burma).

[4] There, we also mentioned that we have restricted ourselves to a few frames for the study reported in this paper, but it is worth mentioning that the data was annotated with the full set of LingFN frames.

[5] http://www.ufjf.br/framenetbr-eng/



[20159] the verb agrees in number and gender with the subject

verbal.verb.n: verb

FE

agreement.agree.v: agrees

FE: particip | fe_Grammatical_Categ | participant

Figure 2: Frame annotations

# Documents	# Sentences	# Frames	# Frame Elements
66	3,926	7,080	4,599

Table 2: Annotated data statistics

this paper, and will be reported on in a separate publication.

4.2 Feature Selection and Training Data Generation

To train machine learning based classification models, the first task is to define a useful set of features, and then compute the feature values from the training data. The area of frame-semantic parsing is well researched meaning that a set of suitable features both for the frame-element identification and frame-element classification tasks have previously been explored (Johansson and Nugues, 2008; Das et al., 2014). Since our objective in this work is not to improve frame-semantic parsing, but rather to show how frame-semantic parsing can be exploited to extract linguistic features from descriptive grammars, we have opted to use the same feature set as described by Johansson and Nugues (2008).

While a detailed explanation of the features can be found in Johansson and Nugues (2008), Table 5 lists *15* features used for training both the frame-element identification and frame-element classification models. The procedure for generating the training instances and computing the features values is as follows: Each sentence of the training data set was parsed using the Stanford constituency parser (Manning et al., 2014) resulting into parse trees as shown in Figure 3. Each node of the tree is then taken as one training instance and the required feature values are computed. The features values given in the last column of Table 5 were computed for the NP node referring to *the qualified nouns* (the one enclosed within the dotted area) as the argument node (i.e. the frame-element node) and with *agree* as the target word (i.e. frame triggering word). When computing for the whole tree, if a given argument node has been annotated as a frame element in the annotation the computed feature vector will get 'Y' as its class label, and 'N' otherwise resulting into the type of training instances shown in Table 3, and making it a binary classification task.

For the frame-element classification task, the objective is not to learn whether an argument node is a frame element or not, but rather to learn the frame-element label for all the annotated nodes (a multi-class classification task). The training data for the frame-element classification task was generated by going through all the nodes in the parse tree (as above), but this time only keeping those nodes which have been annotated as frame elements together with their label. Table 4 below shows a few instances from the generated frame-element classification data set.

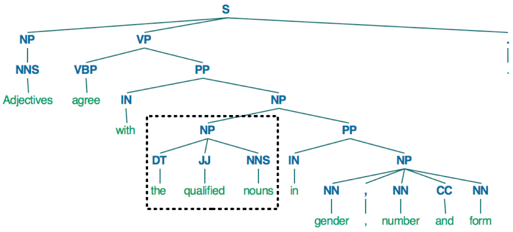


Figure 3: Example parse tree

The described procedure resulted into a set of 197,313 training instances for the frame-element identification task, and 11,904 training instances for the frame-element classification task. After removal of duplicates, 81,878 instances were left. Out of these, 76,036 (92.86%) were labeled 'Y', and the remaining 5,842 (7.14%) were labeled 'N'.

After removing duplicates, 5,855 cases were available for the frame-element classification task, covering 49 different classes of frame elements.

For the frame identification task, a simple dictionary lookup based approach was preferred at this stage simply because there are not many frames in the LingFN, indicating that frame disambiguation is rarely required. In future, we intend to train model for this task as well.

4.3 Data Representation

All of the variables in both of the datasets are the same, except that they differ on the possible set of values for the target variable, *label*. However, in either case, together with the target, all of the variables are categorical.

target_lemma	target_pos	arg_word	arg_word_pos	right_word	right_word_pos	left_word	left_word_pos	parent_word	parent_word_pos	c_subcat	phrase_type	position	fes_list	gov_cat	label
verb	NNS	also	RB	Default	Default	about	RB	walked	VBD	NP->JJNNS	ADVP	R	fe_language_variety#and#fe_data	VP	N
plural	NN	is	VBZ	Default	Default	was	VBD	past	NN	NP->NNNN	SBAR	L	fe_subclass#and#fe_data#and#fe_data_translation	ROOT	Y
plural	NN	past	NN	.	.	The	DT	ROOT	ROOT	NP->NNNN	ROOT	O	fe_subclass#and#fe_data#and#fe_data_translation	ROOT	N
oblique	JJ	ag	NN	Default	Default	twai-na	NN	twai	NN	ADJP->JJJJ	NP	R	fe_sublass#and#fe_data#and#fe_data_translation	VP	Y
decline	VBD	like	IN	Default	Default	Default	Default	like	IN	VP->VBDPP	IN	R	fe_inflectional_scheme#and#fe_form	VP	N

Table 3: A sample from the frame-element identification dataset

target_lemma	target_pos	arg_word	arg_word_pos	right_word	right_word_pos	left_word	left_word_pos	parent_word	parent_word_pos	c_subcat	phrase_type	position	fes_list	gov_cat	label
verb	VB	tong	NN	Default	Default	Default	Default	tong	NN	VP->VBSBAR	NN	R	fe_data#and#fe_data_translation#and#fe_subclass	VP	data
pronoun	NNS	Relative	JJ	Default	Default	Default	Default	pronouns	NNS	NP->DTJJNNS	JJ	L	fe_language_variety#and#fe_data	VP	sublass
prefix	NNS	towards	IN	Default	Default	signifying	VBG	Hon	NNP	NP->DTVBGNNS	UCP	R	fe_subclass#and#fe_data#and#fe_language_variety	VP	fe_Data_Translation
suffix	NN	yo	NN	Default	Default	Default	Default	ya	NN	NP->DTNN	NP	L	fe_subclass#and#fe_language_variety	VP	data
pronoun	NN	who	WP	Default	Default	Default	Default	who	WP	NP->JJNNNN	WP	R	fe_language_variety#and#fe_data#and#fe_data_tr	ROOT	data_translation

Table 4: A sample from the frame-element classification dataset

#	Feature	Explanation	Example Feature Value
1	target_lemma	Lemmatized form of the target word	agree
2	target_pos	Part of speech (POS) tag of the target_lemma	VBP
3	arg_word	The head word of the argument node	nouns
4	arg_word_pos	POS tag of the arg_word	NNS
5	right_word	The right most dependent word of the argument node	the
6	right_word_pos	POS tag of the right_word	DT
7	left_word	The left most dependent word of the argument node	NA
8	left_word_pos	POS tag of the left_word	NA
9	parent_word	Head word of the parent node of the target	agree
10	parent_word_pos	POS tag of the parent_word	VBP
11	c_subcat	Subcategorization frame corresponding to the phrase structure rule used to expand the phrase around the target	VP- >VBP PP
12	phrase_type	Phrase type of the argument node	NP
13	position	Position of the argument w.r.t target word	
14	fes_list	List of frame elements of the triggered frame	(Participant_1, Participant_2, Grammatical_Category, Degree, Frequency, Language_Variety, Reference_Language, Condition)
15	gov_cat	The governing category either S or VP	VP

Table 5: Feature set

In order to achieve best performance while performing machine learning modeling, the right choice of data representation technique for categorical data is very important. The main reason is that there are a limited number of machine learning algorithms that can be directly applied to categorical data. On the other hand, if we can turn them into numerical variables, starting from basic Decision Trees, Naïve Bayes, Support Vector Machines, Logistic Regression, Random Forest, to Multi-layer Perceptron (Deep Learning), almost all of the machine learning algorithms can be applied. There are plenty of techniques to transform categorical values to numerical data. One such technique is one-hot encoding. The basic strategy is to convert each category level (value) of the categorical variable into a new variable, and assign the value *1* to this new variable wherever the corresponding categorical variable equals this level, and *0* otherwise. This is done for all category levels of the variable being encoded except one, which will be redundant (applies when all other associated variables equal zero) and can be any category level. The key is to always create one fewer binary variables than the number of categories. The new binary variables together replace the original categorical variable. The new variables are sometimes termed *dummy variables*, and the approach is also called *Dummy Variables Encoding*. This encoding has the benefit of not weighting a value improperly, but does have the downside of adding more variables to the dataset.

4.4 Model Training

Successful encoding makes the dataset ready to be used for applying machine learning algorithms. We experimented with different machine learning models. A comparison of the machine learning models chosen for binary classification (frame-element identification) and multiclass classification (frame-element classification) tasks for our datasets has been performed. Tables 6 and 7 provide a comparison of various evaluation metrics using average scores of 5-fold cross validation) respectively for the frame-element identification and classification tasks.

Model	Accuracy	Precision	Recall	F_score
Decision Tree	0.926	0.712	0.664	0.921
Logistic Regression	0.936	0.797	0.609	0.922
Naïve Bayes	0.658	0.552	0.686	0.741
Support Vector Machine	0.929	0.465	0.5	0.895

Table 6: Model comparison for the frame-element identification dataset using one hot encoding

Model	Accuracy	Precision	Recall	F_score
Decision Tree	0.789	0.56	0.542	0.786
Logistic Regression	0.817	0.619	0.545	0.808
Naïve Bayes	0.528	0.481	0.489	0.537
Support Vector Machine	0.465	0.019	0.04	0.295

Table 7: Model comparison for the frame-element classification dataset using one hot encoding

The best performing logistic regression models were selected and used in the typological feature extraction system described in the next section.

5 Topological Feature Extraction System

Figure 4 shows the complete architecture of the typological feature extraction system. As shown (the middle part within dotted area), the system takes a descriptive grammar in raw form and annotate it with LingFN frames using the pre-trained models both for the frame-element identification and frame-element classification tasks (i.e. the part above the dotted area). The annotated data is further processed with a simple rule based module to convert those annotations to typological feature values (i.e. the part below the dotted area). Lets take an example to explain this part in particular, and the overall purpose of such a system in general.

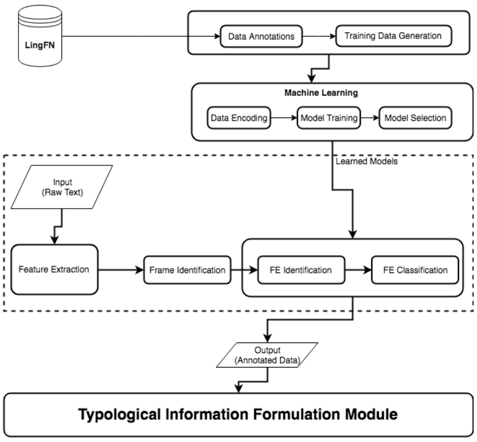
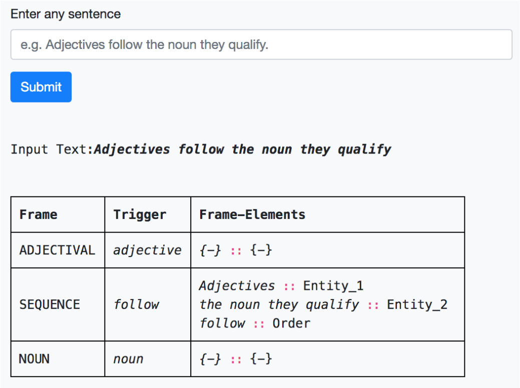


Figure 4: System architecture

Suppose we are interested in finding an answer to the question "What is the order of adjective and noun in the noun phrase" for the Siyin[6] language. The LSI data set contains a grammatical description of this language, and one of the sentences in that description is *The adjectives follow the noun they qualify*. Automatic parsing of this sentence using the developed LingFN parser will result into the annotations shown in Figure 5 (a screenshot from the web demo of the parser).



Input Text:***Adjectives follow the noun they qualify***

Frame	Trigger	Frame-Elements
ADJECTIVAL	*adjective*	*{-}* :: {-}
SEQUENCE	*follow*	*Adjectives* :: Entity_1 *the noun they qualify* :: Entity_2 *follow* :: Order
NOUN	*noun*	*{-}* :: {-}

Figure 5: Automatic frame annotation

This parse contains the answer to the above asked question. However, the typological databases often record answers in a specific format. For example, the answer to the above question could be required to be of one of these values 'NA', 'AN', or 'Both' meaning that the order is 'Adjective-Noun', 'Noun-Adjective', or 'Both' respectively. If required, the above given parse information can be converted into specific feature values using a simple rule-based module such as given below (only a part of the full module is shown). The module simply checks the contents of different frame elements to formulate the feature value.

Using the same sort of procedure and the frames mentioned in Section 3, we have targeted to extract and formulate values for some of the typological features given in the Grambank[7] and other typological databases. A few of these features are given below.

- Can an adnominal property word agree with the noun in gender/noun class?
- Can an article agree with the noun in gender/noun class?
- Can an article agree with the noun in number?
- Can the relative clause precede the noun?

[6] A Tibeto-Burman language spoken in southern Tedim township, Chin State, Burma. Also known as Siyin Chin and Sizang Chin, ISO 639-3: csy

[7] A typological database: https://github.com/clld/grambank.

Algorithm 1 Extract adjective noun order

```
procedure ExtractAdjective-NounOrder(parse)
    for <every frame in parse> do
        if $frame = SEQUENCE$ then
            $NA \leftarrow False$
            $AN \leftarrow False$
            $Both \leftarrow False$
            if $'adjective' \in Entity\_1 \wedge' noun' \in Entity\_2$ then
                if $Frequency \in [sometimes, usually, mostly, often]$ then
                    $Both \leftarrow True$
                else if $order = follow$ then
                    $AN \leftarrow True$
                else if $order = precede$ then
                    $NA \leftarrow True$
                end if
            end if
        end if
    end for
end procedure
```

- Can the relative clause follow the noun?
- Order of Adjective and Noun.
- Order of Subject, Object and Verb.
- Order of Numeral and Noun.
- Order of Relative Clause and Noun.

It is worth mentioning that the same methodology can be used to extract values for various other typological features from the descriptive grammars. This will require designing suitable frames, annotating the data and re-training models. Further, the methodology can be extended to descriptive grammars written in languages other than English.

6 Conclusions and Future Work

We have presented a novel system for automatic extraction of typological features from descriptive grammars based on the theory of frame semantics and frame-semantic parsing. We have presented the methodology, set up the machinery and architecture, and shown the working of this machinery for extraction of feature values of an example typological feature. The methodology is scalable and can easily be extended not only to other features but also to the descriptive grammars written in other natural languages. This is required because there are many grammatical descriptions written in languages other than English (German, French, Spanish, and Russian are among them).

The system we report is expected to be a useful assistance for the development of typological databases, which otherwise are built manually. Manual curation of typological databases is very time and labor consuming, as well as cognitively taxing, thus making the scope of studies based on such databases very limited. We hope with the automatic extraction of typological databases, the scope of studies in typological and other related areas can be broaden further.

The current version of LingFN provides a very limited number of eventful frames restricting us to target only a few typological features. There are 195 typological features listed in Grambank. In the future, we would like to build more frames, annotate more grammars, and automatically extract values for as many as possible features of the Grambank.

In conclusion, the current study can be considered as a proof of concept. In the future, we plan to extend the system and evaluate it against existing manually curated typological databases to compute measures such as precision and recall. Further, the extraction of typological features is just a case study, the automatically annotated grammars are envisioned to be equally useful in other linguistic subdisciplines, in particular the related areas of genetic and areal linguistics. In the future, we also have plans to show the usefulness of the annotated descriptions in these and other related areas.

Acknowledgments

The work presented here was funded partially by the Swedish Research Council as part of the project *South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics* (2015–2019, contract no. 421-2014-969), and partially by the *Dictionary/Grammar Reading Machine: Computational Tools for Accessing the World's Linguistic Heritage* (DReaM) Project awarded 2018–2020 by the Joint Programming Initiative in Cultural Heritage and Global Change, Digital Heritage and Riksantikvarieämbetet, Sweden.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of ACL/COLING 1998*. ACL, Montreal, pages 86–90. https://doi.org/10.3115/980845.980860.

Lars Borin, Maria Toporowska Gronostaj, and Dimitrios Kokkinakis. 2007. Medical frames as target and tool. In *FRAME 2007: Building Frame Semantics Resources for Scandinavian and Baltic Languages. (Nodalida 2007 Workshop Proceedings)*. NEALT, Tartu, pages 11–18.

Lars Borin, Shafqat Mumtaz Virk, and Anju Saxena. 2018. Language technology for digital linguistics: Turning the Linguistic Survey of India into a rich source of linguistic information. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*. Springer, Cham, pages 550–563.

Laura Chiticariu, Yunyao Li, and Frederick Reiss. 2013. Rule-based information extraction is dead! Long live rule-based information extraction systems! In *Proceedings of EMNLP 2013*. ACL, Seattle, pages 827–832.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics* 40(1):9–56.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences* 280(1):20–32. https://doi.org/10.1111/j.1749-6632.1976.tb25467.x.

Charles J. Fillmore. 1977. Scenes-and-frames semantics. In Antonio Zampolli, editor, *Linguistic Structures Processing*, North Holland, Amsterdam, pages 55–81.

Charles J. Fillmore. 1982. Frame semantics. In Linguistic Society of Korea, editor, *Linguistics in the Morning Calm*, Hanshin Publishing Co., Seoul, pages 111–137.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics* 28(3):245–288. https://doi.org/10.1162/089120102760275983.

George A. Grierson. 1903–1927. *A Linguistic Survey of India*, volume I–XI. Government of India, Central Publication Branch, Calcutta.

Yoko Hasegawa, Russell Lee-Goldman, Albert Kong, and Kimi Akita. 2011. FrameNet as a resource for paraphrase research. *Constructions and Frames* 3(1):104–127.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING 2008*. ACL, Manchester, pages 393–400.

Alexandre Kabbach, Corentin Ribeyre, and Aurélie Herbelot. 2018. Butterfly effects in frame semantic parsing: Impact of data processing on model ranking. In *Proceedings of COLING 2018*. ACL, Santa Fe, pages 3158–3169.

Per Malm, Shafqat Mumtaz Virk, Lars Borin, and Anju Saxena. 2018. LingFN : Towards a framenet for the linguistics domain. In *Proceedings of the IFNW 2018 Workshop on Multilingual FrameNets and Constructicons at LREC 2018*. ELRA, Miyazaki, pages 37–43.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL 2014*. ACL, Baltimore, pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of HLT 2006*. ACL, New York, pages 192–199. http://www.aclweb.org/anthology/N/N06/N06-1025.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. *FrameNet II: Extended Theory and Practice*. ICSI, Berkeley.

Frank Seifart, Nicholas Evans, Harald Hammarström, and Stephen C. Levinson. 2017. Language documentation twenty-five years on. *Language* 94(4):e324–e345.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL 2007*. ACL, Prague, pages 12–21. http://www.aclweb.org/anthology/D/D07/D07-1002.

Jae Jung Song, editor. 2010. *The Oxford Handbook of Linguistic Typology*. Oxford University Press, Oxford.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*. ACL, Sapporo, pages 8–15. http://www.aclweb.org/anthology/P03-1002.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL 2008*. ACL, Manchester, pages 159–177.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *CoRR* abs/1706.09528.

Tiago Timponi Torrent, Maria Margarida Martins Salomão, Ely Edison da Silva Matos, Maucha Andrade Gamonal, Júlia Gonçalves, Bruno Pereira de Souza, Daniela Simões Gomes, and Simone Rodrigues Peron-Corrêa. 2014. Multilingual lexicographic annotation for domain-specific electronic dictionaries: The Copa 2014 FrameNet Brasil project. *Constructions and Frames* 6(1):73–91.

Shafqat Virk, Lars Borin, Anju Saxena, and Harald Hammarström. 2017. Automatic extraction of typological linguistic features from descriptive grammars. In *Proceedings of TSD 2017*. Springer, Cham, pages 111–119.

Dekai Wu and Pascale Fung. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Proceedings of HLT-NAACL 2009*. ACL, Boulder, pages 13–16. http://dl.acm.org/citation.cfm?id=1620853.1620858.

Exploiting Open IE for Deriving Multiple Premises Entailment Corpus

Martin Víta
NLP Centre, Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno
Czech Republic
info@martinvita.eu

Jakub Klímek
Department of Software Engineering
Faculty of Mathematics and Physics
Charles University
Malostranské nám. 2/25, 118 00 Prague 1
Czech Republic
klimek@ksi.mff.cuni.cz

Abstract

Natural language inference (NLI) is a key part of natural language understanding. The NLI task is defined as a decision problem whether a given sentence – hypothesis – can be inferred from a given text. Typically, we deal with a text consisting of just a single premise/single sentence, which is called a single premise entailment (SPE) task. Recently, a derived task of NLI from multiple premises (MPE) was introduced together with the first annotated corpus and corresponding several strong baselines. Nevertheless, the further development in MPE field requires accessibility of huge amounts of annotated data. In this paper we introduce a novel method for rapid deriving of MPE corpora from an existing NLI (SPE) annotated data that does not require any additional annotation work. This proposed approach is based on using an open information extraction system. We demonstrate the application of the method on a well known SNLI corpus. Over the obtained corpus, we provide the first evaluations as well as we state a strong baseline.

1 Introduction

Natural language inference (NLI), formerly known as recognizing textual entailment (RTE) task – as a part of natural language understanding (NLU) – belongs to one of the most prominent problems in NLP for more than ten years. Generally, the NLI task is to classify the relationship between a given text and a given hypothesis: whether the hypothesis can be inferred from the text. The task is typically formulated in a "sentence-pair setting", i. e., the text is just a single sentence. According to (Lai et al., 2017), we refer this setting as a *single premise entailment* (SPE for short). Current state-of-the-art approaches are based on deep learning and/or ensemble methods. Over the years, solid resources for supervised learning for SPE were developed. In contrast, problems related to NLI and/or problems derived from NLI, like relation inference (Levy and Dagan, 2016), question entailment (Abacha and Demner-Fushman, 2016), partial/facet entailment (Levy et al., 2013) and (Nielsen et al., 2009), and others, are strongly under-resourced. For further development in these fields, this fact may be limiting.

Recently, a task of NLI from multiple premises was proposed in (Lai et al., 2017) – the idea is based on relaxing the common assumption that the premise is just a single sentence. Again, according to this paper, we will call this derived of NLI task *multiple premises entailment* (MPE for short). Similarly to other mentioned entailment tasks, MPE is also under-resourced: to best of our knowledge, there exists only one annotated corpus (introduced in the original paper) for MPE.

The main aim of this work is to describe a novel method of preparing MPE annotated corpora from existing NLI SPE ones. It is based on using open information extraction systems and on several plausible assumptions. Then we apply the proposed method on a concrete corpus and provide the first evaluation and we state a strong baseline for MPE task on this obtained corpus.

2 Preliminaries and Related Work

In this section we are going to put MPE task in context, describe briefly the notion of open IE and recall two entailment tasks where textual tuples play a certain role.

2.1 NLI Task and Notable Corpora for NLI

There exist several definitions of NLI or, formerly, RTE task. Indeed, the differences among them are rather subtle and have no real consequences for NLP. For completeness, we provide the original definition from (Dagan et al., 2005): "*We say that T entails H if humans reading T would typically infer that H is most likely true.*" The deep insight into the nature of NLI from the logical and philosophical point of view is provided in (Korman et al., 2018).

Originally, RTE was proposed as a binary decision task (entailment/non-entailment). Later, the 3-way task (entailment/neutral/contradiction) became more frequent.

Nowadays, there exists a number of annotated corpora for the NLI task. At the beginning of RTE investigations, it was a collection of RTE corpora created for Pascal/NIST/SemEval challenges. A comprehensive overview of older RTE corpora is provided in (Bentivogli et al., 2017).

A massive development in the field of NLI using deep learning approaches was started after the release of the Stanford NLI corpus (Bowman et al., 2015), probably the most widely used annotated corpus for NLI, containing approx. 570K of annotated sentence text-hypothesis pairs. This corpus was later followed by MultiGenre NLI corpus – MultiNLI (Williams et al., 2018) – of a comparable size, but with a wider range of genres, including spoken language, newspapers, 9/11 etc. Both of these corpora were constructed in a similar way: given a sentence/premise, the annotators were asked to write a sentence that is entailed by the premise, a sentence that is contradictory to the premise, and a sentence neutral w. r. t. the premise, i. e., such that its truth value is independent to the truth value of the premise. According to the classification presented in the paper (Poliak et al., 2018), these two corpora belong to the *human elicited* category. The paper also provides a comprehensive analyses of SNLI, MultiNLI as well as an overview of more recent and specific NLI corpora.

2.2 Natural Language Inference from Multiple Premises

As already mentioned, the novel NLI task that is based on inference over multiple premises was recently introduced in (Lai et al., 2017). Given four premise sentences and one hypothesis sentence, the task is to label this premises-hypothesis pair in a standard 3-way manner – entailment, neutral, or contradiction.

This work was inspired by the *Approximate entailment task* (Young et al., 2014), that arises from processing the image captions – the task is to decide whether a brief caption h (the hypothesis) can describe the same image as a set of captions $P = \{P_1, \ldots, P_N\}$ known to describe the same image (the premises).

The (only one) MPE corpus[1] introduced in the paper (Lai et al., 2017) was created upon the FLICKR30K dataset (Plummer et al., 2015). Hypotheses were generated in by simplifying either a fifth caption describing the same image or a caption corresponding to a different image and given the standard 3-way tags (Poliak et al., 2018). The simplification process relies on the denotation graph (Young et al., 2014) – it is based on normalization and reduction rules (e. g. lemmatization, dropping modifiers and prepositional phrases, replacing nouns with their hypernyms, extracting noun phrases), see (Lai et al., 2017). Each hypothesis has at most 50% overlap with the words in its corresponding premises. The MPE corpus contains 8000 items in the training set, 1000 items in the development set and 1000 in the test set.

To provide a better idea about the corpus, here is an example of positive (entailment) item taken again from (Lai et al., 2017):

Premises:

1. *Two girls sitting down and looking at a book.*
2. *A couple laughs together as they read a book on a train.*
3. *Two travelers on a train or bus reading a book together.*
4. *A woman wearing glasses and a brown beanie next to a girl with long brown hair holding a book.*

Hypothesis: *Women smiling.*
Label: ⇒ ENTAILMENT

In the paper, the authors also investigate the relation between MPE and the standard (SPE) entailment. In this particular MPE task/corpus, each premise consists of four independently written sentences and, using crowdsourcing, single-premise entailment labels for each individual

[1] https://github.com/aylai/MultiPremiseEntailment/tree/master/data/MPE

single-premise-hypothesis pair in the DEV dataset were obtained. Based on these individual labels, it has been shown that majority voting strategies as well as more sophisticated rule based approaches over single labels to obtain the final MPE label do not lead to sufficient results, hence MPE cannot be trivially reduced to multiple SPE tasks.

2.3 Information Extraction (IE) and Open Information Extraction (Open IE)

Information extraction is generally a process of transforming an unstructured textual information into a structured representation in the form of relational phrase and its arguments, usually `(arg1 ; rel-phrase; arg2)`, see (Niklaus et al., 2018). Information extraction deals with a predefined relation vocabulary.

In contrast, in *open information extraction* introduced by (Banko et al., 2007), this assumption is relaxed, i. e., we do not require a fixed vocabulary of relations. Open information extraction systems extract textual n-tuples that represent basic propositions asserted by a sentence (Stanovsky et al., 2018). An example of a result of open IE process taken again from (Stanovsky et al., 2018):

INPUT: *Mercury filling, particularly prevalent in the USA, was banned in the EU, partly because it causes antibiotic resistance.*

OUTPUT:

- `(mercury filling; particularly prevalent; in the USA)`
- `(mercury filling; causes; antibiotic resistance)`
- `(mercury filling; was banned; in the EU; partly because it causes antibiotic resistance)`

2.4 Relational Entailment/Relation Inference

Open information extraction over particular parts of an NLI corpus (hypotheses) was already exploited in (Víta, 2018) in order to obtain "sentence-textual tuple" entailment pairs when introducing a task of *relational entailment*. This task can be employed for checking facts in open knowledge bases, i. e., sets of extracted tuples, see (Mausam, 2016).

Another entailment task based on relational tuples, was introduced by Levy et al. in (Levy and Dagan, 2016) together with a new annotation method for collecting data (on relation inference) in context: the inference task was transformed into simple factoid question answering. The resulting annotated corpus has a form of "textual triple-textual triple plus entailment label".

Indeed, in both cases, textual tuples are a "subject of entailment" – in the final corpus, the tuples are inputs for the entailment decision. In this paper we are going exploit open IE in a different way – to create certain sentences.

3 Methods

In this section we are going to describe a general method of constructing an MPE corpus from a given single premise entailment corpus using an open IE system and also its evaluation. Concrete implementation details are provided in the next section.

3.1 Creating MPE Corpus from SPE One

The main idea of this proposed approach is based on the following observations:

- From longer sentences, it is usually possible to extract multiple textual n-tuples.
- Results of open IE systems is naturally interpretable when reading from left to right (Stanovsky et al., 2018), hence they correspond with sentences.
- The entailment label in an SPE task can be used even for tasks where premise is represented in a "semantically equivalent form".

In order to provide a compact notation, we introduce the following convention: let us denote a set of word types contained in a sentence or a textual n-tuple t by a symbol $||t||$. Let $e(s)$ be a set of textual n-tuples extracted by an open IE system from a sentence s and, finally, let $s(t)$ be a string obtained from a textual n-tuple t by removing auxiliary symbols (brackets and semicolons) – this refers to the second observation: we assume that $s(t)$ can be treated as a sentence – it is a subject of further investigations.

Given a premise P and a hypothesis H and a corresponding entailment label L, we transform this P-H pair into a set of multiple premises $M(P) = \{s(t) \mid t \in e(P)\}$ accompanied with the unchanged hypothesis H a s well as unchanged label L, iff the following conditions hold:

1. $|e(P)| > 1$, i. e., we are interested only in such cases, when more than one tuple is extracted from a given premise,

2. $(\cup_{t \in e(P)} ||t||) \setminus \{$"and"$\} = ||P||$, i. e., each word of the premise is contained in at least one extracted tuple (except "and"),

3. $\forall t \in e(P), ||H|| \not\subseteq ||t||$, i. e., we do not allow situations, where the hypothesis is a shortening of some of the considered tuples,

4. $\forall t, u \in e(P), t \neq u, ||t|| \not\subseteq ||u||$ and $||u|| \not\subseteq ||t||$.

The first condition is obvious. The second one ensures that the information contained in the set of extracted tuples is the same as in the original premise under the natural assumption that a content of a sentence can be fully represented by a set of textual n-tuples for sufficiently high n. By the third condition we want to avoid cases of trivial entailment from one of the multiple premises. The last condition excludes situations when the result $M(P)$ set contains "inclusion sentences" like: `Three men standing on grass` and `Three men standing on grass by the water looking at something on a table.`

The procedure of generating the MPE corpus from an existing NLI one is now straightforward: for each $P - H - L$ triple of NLI corpus check the conditions 3.1 for P, $e(P)$ and H – if satisfied, the obtained $M(P)$-H-L item is added into MPE corpus being created. These items can be further filtered according to different intentions, e. g., we want to deal with items having at least k premises or at most m premises, for instance. Obviously, the quality of the annotation of the original SPE corpus hardly influences the quality of the obtained MPE corpus.

Unlike the MPE corpus from (Lai et al., 2017) where all entailment items contain a set of a *fixed number* of premises (4), we will generally obtain sets of *variable number* of premises.

Reducing the number of trivial inferences by limiting lexical overlap In order to reduce the number of items where the inferences can be done trivially, we may set up a threshold for lexical overlap and consider only such items that the fraction of the number of hypothesis tokens that appear in at least one premise and the number of hypothesis tokens after stopwords removal is lower or equal to a given threshold.

Remark The process can also produce positive (entailment) items where not all premises in the $M(P)$ set take part in the entailment of the hypothesis, i. e., one or more of the premises involved together with the hypothesis form a neutral pair/pairs. This phenomenon will be a subject of further investigations. Nevertheless, in real situations, it is natural to deal with it, as well as with premises such that one extends information provided by another, thus not fulfilling condition 4.

According to (Poliak et al., 2018) again, our proposed corpus can be classified as *automatically recast*, i. e., a corpus that was automatically generated from an existing dataset constructed for a different NLP task and the labeling was done with no or a little manual work. In such cases, some properties of the source corpora may be also transferred into the recasts.

3.2 Quality Evaluation of Our Corpus

NLI corpora are prone to contain several types of *annotation artifacts* (Gururangan et al., 2018). For example, a negation can indicate a contradiction label, mainly in human elicited corpora. Using "generic" words such as "animal" or "instrument" is often typical for the entailment label thanks to common annotators' strategies.

In order to obtain a better insight into the characteristics of the obtained corpus, we are going to investigate the role of occurrence of certain single words in hypotheses when predicting labels and the role of the hypothesis-context relation as well as appearance of different (annotators') patterns.

3.3 Lexical Biases

At first, we are going to focus on words that are strongly indicative of each inference class. In (Gururangan et al., 2018), the authors use pointwise mutual information (PMI) for all words and classes in the training set:

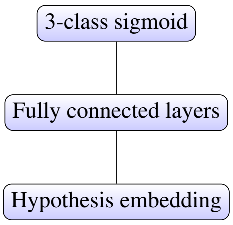
$$PMI(word, class) = \log \frac{p(word, class)}{p(word, .)p(., class)}.$$

The authors use add-100 smoothing to emphasize word-class correlation and select Top 5 words in each class.

In (Poliak et al., 2018), a conditional probability was used:

$$p(l|w) = \frac{count(w, l)}{count(w)}.$$

Figure 1: Overall architecture of the simple hypothesis-only classifier



Then they analogously select Top words for each class (label) l. If $p(l|w)$ is highly skewed across labels, there exists the potential for a predictive bias (Poliak et al., 2018). In this paper, we are going to use this second approach.

3.4 Hypothesis-Only Approach

Annotation artifacts in NLI corpora are common, since annotators, mainly in human elicited corpora development, have different strategies and patterns for generating hypotheses. There are also artifacts that arise from the "hypothesis-context" relation. To model the degree of annotation artifact existence, a classifier that uses only hypotheses for predicting the entailment classes can be trained. In other words, the classifier completely ignoring the information contained in the premise(s) is used.

In (Poliak et al., 2018), the authors call this approach "hypothesis-only" and they used a modified `InferSent` model (Conneau et al., 2017).

In (Gururangan et al., 2018), `fastText` (Joulin et al., 2017), bag-of-words and bigram based model was used, there it is called "premise-oblivious text classifier".

A general architecture of a hypothesis-only classifier is depicted in Figure 1.

Since the outstanding results of BERT model in SPE NLI task[2], we use BERT (Devlin et al., 2018) embeddings.

4 Results and Discussion

The proposed approach for MPE corpus development can be generally used on any single premise NLI annotated corpus in a language where a suitable open IE system is available. To demonstrate it on a concrete dataset, we have chosen the already mentioned SNLI corpus.

4.1 MPE(SNLI) Corpus

As an input for our approach, the training dataset of SNLI was used. From 150.736 *unique* texts/premises, 229.428 n-tuples were extracted, i. e., 1.522 n-tuples per sentence in average. The extraction process was performed by Open IE 5.0[3] (Mausam, 2016). Additional labels provided by the system (like `L:` for location) were removed.

For premise-hypothesis-GoldLabel items of SNLI train, we follow the list of requirements in subsection 3.1. The SNLI corpus contains appox. 2% of items without GoldLabel (marked "-") (Bowman et al., 2015). Even if P-H-L item with L ="-" meets the requirements, it is not added to the corpus being created. Hence the corpus contains only three common labels (entailment, neutral, contradiction) – the number of such items was 96.

Moreover, we set-up a threshold for a lexical overlap according to (Lai et al., 2017) to be equal to 0.5.

The obtained dataset contains 45.622 items. It was then randomly split into train/dev/test datasets containing 32.000, 7.000 and 6.622 items, respectively. Although the SNLI corpus is roughly balanced between the three classes, our corpus contains slightly higher fraction of contradiction labels, mainly because of application of the lexical overlap threshold – high lexical overlap typically indicates the entailment class.

To provide a better idea about the proposed corpus, we provide an example of each label – the premises are syntactically transformed (from t to $s(t)$ in our notation): from textual tuples into sentences, i. e. brackets and semicolons are removed, the first letter is capitalized.

Example 1
Premises:

1. *A white dog with his tongue out is in the snow.*
2. *A brown dog with his tongue out is in the snow.*
3. *A black dog with his tongue out is in the snow.*

Hypothesis: *There are animals outdoors.*
Label: ⇒ ENTAILMENT

Example 2
Premises:

[2] See state-of-the-art results on SciTail corpus: `https://leaderboard.allenai.org/scitail/submissions/public`

[3] `https://github.com/dair-iitd/OpenIE-standalone`

1. *3 women posing for a picture.*
2. *3 women are sitting down.*

Hypothesis: *The women are smiling.*
Label: ⇒ NEUTRAL

Example 3
Premises:

1. *A baby has food on his face.*
2. *A baby eats.*

Hypothesis: *Baby playing with a dog.*
Label: ⇒ CONTRADICTION

The distribution of labels in the corpus is provided in Table 1.

Table 1: Distribution of labels in MPE(SNLI)

	Train	**Dev**	**Test**
entailment	0.231	0.235	0.231
neutral	0.361	0.362	0.372
contradiction	0.407	0.403	0.396

As already mentioned, we do not require the same number of premises in each corpus item, the distribution of number of premises in the corpus is also provided, see Table 2.

Table 2: Number of premises in MPE(SNLI)

# prem.	2	3	4	5	≥ 6
# items	37765	5013	2294	253	297

4.2 MPE(SNLI) Lexical Biases

In order to select the most characteristic words for each entailment label, we used the conditional probability of a word w w. r. t. the label l. Since there are extremely discriminative words having a very low frequency, we focus only on words that appear at least five times in the training dataset. Table 3, Table 4, and Table 5 present Top 5 words for each class in MPE(SNLI) corpus (training): entailment, neutral, and contradiction respectively together with corresponding values of conditional probability.

These results correspond with our intuition, if we consider the fact that the source of SNLI sentences are mainly photo captions and the fact that the original SNLI corpus was human elicited. Using lexical items that refer to "general" words like *object, similar, multicolored* matches common strategies and patterns when creating entailment pairs. We can also note that the first item in the "contradiction list" is linked to negation. Investigations of annotation strategies and patterns can be a part of the future work.

Table 3: Cond. prob.: l = "entailment"

w	$p(l\|w)$
similar	0.875
facial	0.857
multicolored	0.857
object	0.848
least	0.840

Table 4: Cond. prob.: l = "neutral"

w	$p(l\|w)$
favorite	0.976
tired	0.964
first	0.940
tips	0.9385
tour	0.933

Table 5: Cond. prob.: l = "contradiction"

w	$p(l\|w)$
nobody	0.994
naked	0.971
quietly	0.968
cats	0.938
napping	0.931

4.3 Hypothesis-Only Classifier

Sentence (hypothesis) BERT embeddings were computed using `BERT-as-a-service` application (Xiao, 2018). We have used a pretrained BERT model[4] (12-layer, 768-hidden, 12-heads, 110M parameters). Each hypothesis was encoded as a 768-dimensional vector. The optimal dimension ($d = 100$) of the single hidden layer was obtained by a grid search.

The achieved accuracy reached **0.671** on the test dataset, highly above the majority baseline that equals **0.396**. This result indicates a notable

[4] https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

presence of annotation artifacts in MPE(SNLI) corpus. Nevertheless, approximately the same accuracy was obtained on the test set of the original SNLI single premise corpus using premise-oblivious `fastText` classifier. We may conclude that proposed corpus achieves a comparable level of annotation artifacts occurrence.

4.4 Conclusion

We have proposed a method of exploiting an open information extraction system for transforming a NLI single premise corpus into a multiple premises entailment (MPE) setting. The method was then applied on SNLI training data and an annotated MPE(SNLI) corpus was obtained. The final corpus will be available at: https://github.com/martinvita/openIE-MPE

4.5 Future Work

This work presents the first steps in creating corpora for MPE task using open IE with a particular application on SNLI data. The further work on the proposed MPE(SNLI) corpus will include extending investigations outlined in this paper, e. g. obtaining deeper insight into lexical bias or investigations about individual entailment classification between hypotheses and individual premises – "elements of M(P)" etc. as a natural continuation of the work.

The keystone of the further work are the investigations of relations among the members of the premise set $M(P)$ together with the development of the entire MPE task . According to our intuition, the MPE should be significantly more difficult than SPE task in the sense that the entailment judgement should be based on a fusion of information contained in the premises. This is also a natural step after the initial work (Lai et al., 2017) – approving the importance of the MPE task.

Both the quality and quantity of extracted tuples – hence also the number of premises and items in the MPE corpus – are strongly influenced by the functionality and quality of the open information extraction system used. Comparing results/outputs of different open IE systems is another prospective field of study.

As we have seen from examples provided in the text, the proposed process led to the MPE corpus with a variable number of premises whereas in the first MPE corpus (Lai et al., 2017), each item contains preciously four premises. In order to prepare a formally compatible corpus, we are interested also in "normalizing" the number of premises using various techniques, e. g. a paraphrase generation for cases when we have less premises than needed, and concatenation in cases when the number of premises exceeds the required number.

Obviously, the proposed process does not rely on certain properties of SNLI, hence it can be straightforwardly applied to other corpora, (e. g. MultiNLI, SciTail etc.), even to NLI single-premise corpora in different languages where open IE tools are available. SNLI is prone to several biases (that are transferred to MPE corpus), thus we can expect the result obtained by applying our procedure on other corpora can lead to more valuable and inspiring results.

A general task when having MPE corpora of suitable quality and volume, is the development of classifiers for MPE task based on different architectures, i. e., general development in MPE field as well as further study on the mutual relationship between the SPE NLI and the MPE NLI task.

Remark Finally, it should be noticed that MPE task is related to NLI with external/background knowledge, which seems to be a promising direction in the field of NLU (Jiang et al., 2018). Having a premise-hypothesis pair and an external/background knowledge that can be formalized in the form of sentences, we can generally add these sentences as additional premises. The key question is obviously the process of selection/recommendation of relevant sentences to become these new additional premises. Clearly, the number of premises needs to be limited. This observation illustrates the importance of the MPE task in the entire NLU field.

References

Asma Ben Abacha and Dina Demner-Fushman. 2016. Recognizing question entailment for medical question answering. In *AMIA Annual Symposium Proceedings*. American Medical Informatics Association, volume 2016, page 310.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*. volume 7, pages 2670–2676.

Luisa Bentivogli, Ido Dagan, and Bernardo Magnini. 2017. The recognizing textual entailment challenges: Datasets and methodologies. In *Handbook of Linguistic Annotation*, Springer, pages 1119–1147.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* .

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*. Springer, pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324* .

Shan Jiang, Bohan Li, Chunhua Liu, and Dong Yu. 2018. Knowledge augmented inference network for natural language inference. In *China Conference on Knowledge Graph and Semantic Computing*. Springer, pages 129–135.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. pages 427–431.

Daniel Z Korman, Eric Mack, Jacob Jett, and Allen H Renear. 2018. Defining textual entailment. *Journal of the Association for Information Science and Technology* 69(6):763–772.

Alice Lai, Yonatan Bisk, and Julia Hockenmaier. 2017. Natural language inference from multiple premises. *arXiv preprint arXiv:1710.02925* .

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 249–255.

Omer Levy, Torsten Zesch, Ido Dagan, and Iryna Gurevych. 2013. Recognizing partial textual entailment. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. volume 2, pages 451–455.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pages 4074–4077.

Rodney D Nielsen, Wayne Ward, and James H Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering* 15(4):479–501.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. *arXiv preprint arXiv:1806.05599* .

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*. pages 2641–2649.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. *arXiv preprint arXiv:1805.01042* .

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. pages 885–895.

Martin Víta. 2018. From building corpora for recognizing faceted entailment to recognizing relational entailment. In *Position Papers of the 2018 Federated Conference on Computer Science and Information Systems, FedCSIS 2018, Poznań, Poland, September 9-12, 2018.*. pages 33–38. https://doi.org/10.15439/2018F381.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 1112–1122. http://aclweb.org/anthology/N18-1101.

Han Xiao. 2018. bert-as-service. `https://github.com/hanxiao/bert-as-service`.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2:67–78.

Towards Adaptive Text Summarization:
How Does Compression Rate Affect Summary Readability of L2 Texts?

Tatiana Vodolazova, Elena Lloret
Dept. of Software and Computing Systems
University of Alicante
Apdo. de Correos 99
E-03080, Alicante, Spain
{tvodolazova,elloret}@dlsi.ua.es

Abstract

This paper addresses the problem of readability of automatically generated summaries in the context of second language learning. For this we experimented with a new corpus of level-annotated simplified English texts. The texts were summarized using a total of 7 extractive and abstractive summarization systems with compression rates of 20%, 40%, 60% and 80%. We analyzed the generated summaries in terms of lexical, syntactic and length-based features of readability, and concluded that summary complexity depends on the compression rate, summarization technique and the nature of the summarized corpus. Our experiments demonstrate the importance of choosing appropriate summarization techniques that align with users needs and language proficiency.

1 Introduction

It is often the case that people, at some point in their lives, are incapable of benefiting from available information due to various aspects of text complexity resulting from domain-specific terminology and grammatical structure complexity. The literature has identified specific instances of this, such as: diabetes patients with no medical training who try to make sense of recent scientific advances in the treatment of the disease (Ong et al., 2008); elderly people forced to embrace the technical terminology of the digital age when using computers and mobile phones (Li and Perkins, 2007); parents trying to decipher the latest slang words used by their teenage children (Vizgirdaite, 2009); people with different degrees of learning difficulties such as aphasia (Carroll et al., 1998), dyslexia (Rello et al., 2012) or autism (Štajner et al., 2012); and, second languag (L2) learners trying to infer the meaning of idioms from the literal meaning of their constituents (Charteris-Black, 2002).

Extensive research has been carried out in the field of automatic text simplification and text enrichment (Rello et al., 2014; Aranzabe et al., 2012; Woodsend and Lapata, 2011; Thomas and Anderson, 2012; Barbu et al., 2015). But only a few studies integrate them into other applications of natural language processing (NLP), such as, for example, in text summarization for the purpose of improving readability. Text summarization is not considered to be primarily a simplification task. However, it becomes useful when long documents are involved as it aims to reduce text processing time and thus to access quicker the main concepts of the document. In this context most of the studies apply text simplification to reduce redundant or less important information and to increase the informativeness of extractive summaries, but not their readability (Jing, 2000). Lloret et al. (2019) point out that summaries are rarely evaluated for readability. But text summarization can clearly benefit from readability assessment to better serve its purpose of saving reading time and to avoid the generation of incomprehensible summaries. We take this idea further and suggest that to maintain an optimal level of summary complexity and to adapt it in a personalized way according to user needs and language proficiency, the summarization aproach needs to identify and integrate the necessary degree of simplification.

This paper presents the initial study of ongoing research on the development of an abstractive text summarization approach that can adapt generated summaries to user language proficiency and cognitive abilities. In this study, we rely on the fact that texts for L2 learners are written in a well-structured manner with clear style. Unlike first language texts, these texts, in addition to their semantic readability features, include a broader range of syntactic features, thereby providing an

Proceedings of Recent Advances in Natural Language Processing, pages 1265–1274,
Varna, Bulgaria, Sep 2–4 2019.

overall richer set of readability metrics for examination (Heilman et al., 2007). Once identified these features can be used to measure variation in readability of automatically generated summaries.

For this purpose we harvested a corpus of texts for L2 learners of English that are classified into 7 levels according to language proficiency . This dataset provides an appropriate setup to explore the distribution of different readability characteristics across the levels and to study how these characteristics change when summarization is applied.

The main contributions of this paper are:

- we experiment with a new dataset of graded L2 learner texts that can be used both for text simplification and text summarization tasks;
- we test a number of summarization approaches on this corpus and demonstrate that compression rate always affects the complexity of generated summaries;
- we show that the domain of the corpus affects the results;
- we prove that the complexity of generated summaries varies depending on the summarization technique and the readability metric.

2 Related work

The present study spans two fields of NLP: text summarization and readability assessment as a part of the text simplification process.

Only a few studies address the problem of how text summarization affects summary readability. Petersen and Ostendorf (2007) are among the first to point out that text summarization techniques alone do not control the degree of readability of generated summaries because sentences with challenging vocabulary and complex grammatical structures may be chosen. Lloret et al. (2019) further explore this idea, showing that text summarization does not maintain the same degree of text complexity as the original document. However, based on their experiments with a corpus of unsimplified newswire documents for native speakers DUC 2002[1] and contrary to the observation of Petersen and Ostendorf (2007), they conclude that on some readability metrics the summaries generated with the compression rate of 20% score better and are easier to comprehend than the original texts. For their experiments Lloret et al. (2019) used a total of 9 lexical and length-based readability features and a modular extractive text summarization approach that allowed the testing of how anaphora resolution, word sense disambiguation and textual entailment affect the readability of summaries.

Only a handful of text summarization methods so far have integrated readability assessment to select not only the most informative, but also the most comprehensible sentences. Nandhini and Balasundaram (2014) designed one of such approaches. They represent each document as a set of 4 informative features (sentence position, title similarity, etc.) and 5 readability features (word length, sentence length, etc.) and treat summarization as an optimization problem to maximize the average informative score of the summary and to improve its readability. However, their set of readability features is small and they do not study the relative importance of each feature with respect to the corpus or the target language proficiency level.

In recent years, several studies appeared that address readability assessment and text simplification for L2 learners. Vajjala and Lučić (2018) compiled the OneStopEnglish corpus of simplified level-annotated news articles for L2 learners with comparable texts across all levels. Xia et al. (2016) trained a machine learning algorithm for readability assessment on past Cambridge English Exam papers. Their set of readability features includes data from English Vocabulary Profile, an online vocabulary resource with integrated grading scale based on the Common European Framework of Reference for Languages (CEFR) (Council of Europe, 2001).

Our study has been inspired by the aforementioned research and further expands on the idea of Lloret et al. (2019) by experimenting with a graded simplified corpus of texts for L2 learners of English, adding vocabulary-based and syntactic families of readability features, including an abstractive summarization system and testing these metrics with 4 different compression rates.

3 Data

The goal of this paper is to provide an initial analysis on how automatic text summarization affects readability for L2 learners with the long term goal of integrating these findings into an abstractive text summarization approach capable of adapting generated summaries to user language proficiency, knowledge and cognitive abilities. This requires a

[1] http://duc.nist.gov/

Level	0	1	2	3	4	5	6
# of docs	251	251	251	251	250	250	250
CEFR	A2	A2	A2	A2-B1	B1-B2	B2	C1-C2
% of docs	64.94%	45.02%	23.11%	61.35%	78.40%	54.40%	40.80%
Difficulty	low	low-med	med	pre-int	pre-int/int	int	upper-int

Table 1: Statistics for the BNE corpus

corpus of simplified texts that are adapted to different reading proficiency levels and also include the original source documents. To the best of our knowledge there is no such freely available level-annotated corpus of learner's materials that can also be used for text summarization.

Motivated by this fact we harvested a corpus of texts for English learners from Breaking News English website[2]. Breaking News English (BNE) is a website with different resources for English language teachers created and maintained since November 2004 by an experienced ESL/EFL teacher Sean Banville[3]. This website was nominated for a British Council ELTons award in the category "Innovation in Learner Resources" in 2014. We obtained permission from the author to use it for research purposes. The site contains materials classified into seven levels from 0 to 6 that roughly correspond to CEFR grading scale, comprising levels from A2 to C2 (see Table 3). A new lesson, containing quizzes, reading and listening activities, appears every two days. The main component of each lesson is a piece of news from 120 to 250 words long, depending on the level. Texts cover a broad range of topics, but try to avoid more emotive and sensitive ones (Banville, 2005). As the vocabulary difficulty depends on the topic, each news article topic is assigned to one of two groups, representing linguistic levels 0 to 3 or 4 to 6, respectively. The grading process is the following: first, the author manually creates a text for level 3 or 6 and then makes the easier levels by reducing sentence length, simplifying grammar, introducing easier vocabulary and avoiding idioms. We will revisit this grading policy in Section 4.2, as it affects readability statistics of the corpus.

The distinguishing features of this resource are:

- each text is available at different levels of complexity.
- each text can be considered as a summary of a set of news articles that can be extracted from the provided URLs.

As mentioned in Section 2 the OneStopEnglish corpus (Vajjala and Lučić, 2018) is a similar resource for readability assessment that contains the same texts rewritten for elementary, intermediate or advanced reading proficiency levels. However, given the 7 levels from the BNE dataset we expect to be able to identify the more subtle readability differences between them. Going forward, with access to the original news articles via URLs, we will be able to develop and test an adaptive multi-document summarization approach on this corpus.

The BNE website has been evolving since its creation in 2004, thereby level annotation is available only starting from July 2013. We harvested all the suitable data, but for this initial research we used a subset from March 2016 to January 2019, resulting in 1,754 news articles in total and 250/251 articles for each of the 7 levels. Table 1 shows further statistics of the corpus where *CEFR* and *Difficulty* rows reflect grading information provided on the resource's website.

To contrast BNE's website CEFR level annotation with the CEFR annotation of other state-of-the-art resources, we evaluated our corpus with the readability assessment method developed by Xia et al. (2016) and trained on past Cambridge English Exam papers and on the set of 100 additionally annotated news articles. The row *% of docs* illustrates how many texts were identified by this method as belonging to the indicated CEFR level.

4 Readability

4.1 Features

The most recent research on readability assessment uses machine learning based approaches in combination with a broad set of linguistic features. Such sets of features are usually organized into families that share similar linguistic properties. They typically include length-based, syntactic, lexico-semantic and discourse-based features, among others. Experiments show that different families, or even individual features, affect the accuracy of the classifier in a different manner (Xia et al., 2016).

[2] https://breakingnewsenglish.com/

[3] https://www.linkedin.com/in/seanbanville/

Following previous research on assessing readability of summaries by Lloret et al. (2019), we selected the same set of 9 readability features. However, after detailed analysis of the set and the revision of other research in readability assessment of L2 texts, we added further lexical and syntactic features (Xia et al., 2016; Heilman et al., 2007). The rationale for this was twofold. First, experiments show that these families of features significantly improve performance of classifiers and therefore help to correctly identify the grade of text complexity (Pitler and Nenkova, 2008). Second, the initial set of readability metrics would not be capable of grasping grammatical and vocabulary changes that are an integral part of second language acquisition. To cover these aspects of text complexity we added 12 more features[4] including the revised Dale-Chall formula (Chall and Dale, 1995). This formula calculates the proportion of words that do not belong to the list of 3,000 familiar words. Some studies view this formula as a simplified version of a language model (Collins-Thompson and Callan, 2004). The complete set of 21 features, by families, is described below.

Traditional Features This family includes superficial length-based features and traditional readability formulas that are easy to compute, but provide a competitive baseline.

- Flesch Reading Ease (FRE) (Flesch, 1948)
- Avg. Word Length (AWL)
- Mean Length of a Sentence (MLS)
- Avg. Number of Sentences (ANS)
- Avg. Text Length in Tokens (ANT)

Lexical Features Our final feature set does not include discourse features, although 4 already integrated features from the original feature set of Lloret et al. (2019) based on noun and proper noun ratios may be related to the entity-density features, which could cast some light on the discourse properties of the corpus (Feng et al., 2009).

- Word variation index (OVIX) is a variety of type-token ratio measure (Hultman and Westman, 1977)
- Revised Dale-Chall formula (DC)
- Proper Noun Ratio (PNR)
- Avg. Number of Unique Proper Nouns (uPNR)
- Noun Ratio (NR)
- Pronoun Ratio (PR)

Syntactic Features Heilman et al. (2007) emphasize that grammatical features may play a more important role in readability assessment for the L2 learners than for the native speakers. Following their example, we calculate the last 4 features in this family on a per word basis.

- Parse Tree Depth (PTD)
- Noun Phrase Ratio (NPR)
- Verb Phrase Ratio (VPR)
- Adjective Phrase Ratio (ADJPR)
- Adverbial Phrase Ratio (ADVPR)
- Avg. number of SBARs per sentence (SBAR)
- Ratio of Passive Voice constructions (PV)
- Avg. number of Relative Clauses (RC)
- Past Participles (VBN)
- Modal verbs (MD)

4.2 BNE Readability Statistics

In Section 3 we explained how Sean Banville manually creates the news articles for each complexity level. His grading scheme does not correspond one-to-one to other established classifications, such as for example CEFR level annotation. To analyze how readability varies across the 7 levels, we extracted statistics from the BNE corpus for each of the 21 features. Table 2 contains average values by level for each of the selected readability measures and Table 3 Pearson correlation coefficient.

Length-based readability features reveal irregularities in the size of articles between levels 3 and 4 when the more complex topic replaces the easier one. One can observe that the average number of tokens and sentences per article, as well as the mean sentence length (ANS, ANT, MLS) gradually increase from level 0 to level 3, but then in level 4 decrease almost to the values of level 1, again increasing and surpassing previous values up until level 6. It affects FRE formula and introduces the same irregularity in its values. These size differences are intended by the author. Indeed, he states on his website that levels 4 and 5 texts are shorter than levels 2 and 3 respectively. Average word length is the only length-based feature that grows linearly and slightly smooths out the values of FRE.

This tendency in length-based features also affects syntactic features, since more complex syntactic constructions tend to contain more words. Considering these findings, none of the discussed features in isolation, except for AWL, could be

[4] We indicate them with an * in Table 2

Level	**0**	**1**	**2**	**3**	**4**	**5**	**6**
FRE	**69.123**	**64.060**	**59.235**	**53.951**	**55.470**	**50.073**	**42.011**
AWL	4.621	4.681	4.754	4.839	4.860	4.937	5.063
MLS	**9.995**	**12.186**	**14.292**	**16.846**	**14.027**	**16.623**	**20.432**
ANS*	10.912	12.498	13.677	14.040	10.716	11.476	11.160
ANT	**106.000**	**147.701**	**188.829**	**229.016**	**144.572**	**183.588**	**220.160**
OVIX	45.091	45.855	46.258	47.009	50.507	51.076	52.702
DC*	**7.106**	**7.438**	**7.720**	**8.000**	**8.216**	**8.540**	**8.989**
PNR	0.053	0.052	0.052	0.055	0.048	0.049	0.052
uPNR	0.033	0.033	0.033	0.036	0.033	0.034	0.038
NR	0.311	0.304	0.301	0.299	0.305	0.302	0.300
PR	0.072	0.069	0.067	0.063	0.066	0.062	0.058
PTD*	**8.542**	**9.320**	**9.997**	**10.821**	**9.766**	**10.434**	**11.469**
NPR*	2.650	2.893	3.060	3.258	2.966	3.141	3.421
VPR*	1.067	1.091	1.114	1.130	1.106	1.136	1.160
ADJPR*	0.223	0.250	0.262	0.284	0.242	0.247	0.288
ADVPR*	0.200	0.241	0.285	0.354	0.281	0.323	0.401
SBAR*	0.337	0.401	0.460	0.527	0.420	0.476	0.542
PV*	0.005	0.006	0.007	0.008	0.009	0.009	0.011
RC*	0.006	0.008	0.009	0.010	0.008	0.009	0.010
VBN*	**0.117**	**0.171**	**0.226**	**0.312**	**0.285**	**0.354**	**0.491**
MD*	0.257	0.256	0.261	0.265	0.240	0.235	0.233

Table 2: Readability statistics for the BNE corpus

used to correctly identify the level of a BNE document. Among lexical features, we want to point out the OVIX and the DC metrics that together with the AWL suggest that the best approach to automatically assess complexity of this corpus may involve a statistical language model.

Based on the values of Pearson correlation coefficient we reduced our readability set to 3 features including FRE, DC and PTD - one feature per family - as the most reliable readability indicators for the given corpus (see Table 3). Since average sentence length is one of the components of FRE, we included FRE and not MSL in this set, even though it has a higher correlation coefficient. We will use this reduced set in the next experiments.

5 Summarization Methods

For this research we considered 7 state-of-the-art methods with different summarization techniques that include graph-based and frequency-based methods, methods that implement language models, incorporate such heuristics as word sense disambiguation and anaphora resolution and involve abstractive text summarization. Implementations of 2 of them were obtained from the authors (ExL19, AbL15), while the remaining 5 were provided by the sumy framework[5]. Each

Feature	**Pearson correlation**	
	r	**p-value**
FRE	**-0.6197**	**0**
AWL	0.4133	0
MLS	**0.6459**	**0**
ANS	-0.1167	0
ANT	0.6154	0
OVIX	0.4080	0
DC	**0.5429**	**0**
PNR	-0.0197	-0.410
uPNR	0.0527	-0.0272
NR	-0.0597	-0.0125
PR	-0.1393	0
PTD	**0.5454**	**0**
NPR	0.4114	0
VPR	0.2279	0
ADJPR	0.0753	-0.0016
ADVPR	0.3218	0
SBAR	0.2555	0
PV	0.2430	0
RC	0.1252	0
VBN	0.5199	0
MD	-0.0493	-0.0388

Table 3: Correlation coefficient for the BNE corpus.

[5] https://github.com/miso-belica/sumy

method is described in more detail below.

Luhn's classical technique was one of the first summarization algorithms ranking sentences on word and phrase frequencies (Luhn, 1958). It weighs each sentence according to the number of significant words it contains ignoring high frequency common words from a stop word list.

SumBasic (Nenkova and Vanderwende, 2005) is another frequency-based summarizer that incorporates context information. It assumes that distribution of words in a human summary is similar to that of the original text. The authors reported that it outperformed many of the DUC 2004 systems, so it is frequently used in the literature as a baseline summarizer.

KLSum (Haghighi and Vanderwende, 2009) uses Kullback-Leibler divergence to measure the similarity between a sentence and the language model of the document and selects a set of sentences such that the distribution of words in the selected sentences is as similar as possible to the overall distribution of words in the document.

ExL19[6] Lloret et al. (2019) designed a modular extractive text summarization approach based on frequencies. For this experiment we selected the combination that includes anaphora resolution, word sense disambiguation and textual entailment, scoring sentences on concept frequencies.

LexRank (Erkan and Radev, 2004) is a graph-based approach that uses cosine similarity of TF–IDF vectors to calculate pairwise similarity between two sentences. The final score of each sentence is the sum of the weights of all the edges connected to it. The sentences are ranked by applying PageRank to the resulting graph.

TextRank (Mihalcea and Tarau, 2004) is another graph-based approach that uses PageRank to rank the sentences. In the case of TextRank, the similarity between two sentences is calculated as the number of words they have in common normalized by sentence length. In contrast to LexRank, TextRank recursively changes the weights of the sentences incorporating in this manner how all the sentences in the graph relate to each other.

AbL15 (Lloret et al., 2015) is an abstractive text summarization approach that incorporates the stages of text interpretation, transformation and summary generation. Each text passes through the process of syntactic simplification that splits complex sentences into shorter ones. Subsequently, the system extracts subject-verb-object triplets, identifies named entities and head nouns in nouns phrases, and supplies all this information to the summarizer. The summarizer scores each sentence representation based on the extracted information. In the final step the system translates each sentence into its surface representation and selects the highest rated sentences with respect to the maximum allowed summary size.

6 Results

The final setup of our experiment includes: 7 summarization methods (Luhn, SumBasic, ExL19, LexRank, TextRank, KLSum, AbL15); 4 compression rates (20%, 40%, 60% & 80%); and, 3 readability metrics (FRE, DC and PTD). To evaluate how much readability of generated summaries ($sread$) differs from that of the original documents ($oread$) we calculated for each document *i* the percent deviation (*PD*)

$$PD_i = \frac{(sread_i - oread_i)}{oread_i} * 100 \qquad (1)$$

and then averaged it across the entire set. This value can be both negative and positive and indicates whether the summary is more or less complex than the original text.

Another way to assess the summarizer's performance on readability is to calculate the average absolute deviation from the original text's complexity, in other words from 0. With this information we can determine the degree to which the summaries, independently from being more complex or simple, differ from the original documents across all the compression rates. Although the complexity of a system may vary across all the compression rates, their average variation may be very small.

$$AAD_s = \frac{\sum_{n=1}^{4} |PD_n - 0|}{4} \qquad (2)$$

where AAD_s is the average absolute deviation of a given system s and PD_n is its percent deviation for the compression rate n.

6.1 Length-based readability results

Table 4 shows how selected text summarization methods affect length-based readability features. FRE measure depends on the number of tokens per

[6] We will refer to the systems developed by Lloret et al. using ExL19 for extractive and AbL15 for the abstractive one.

%compression	Luhn	SumBasic	KLSum	ExL19	LexRank	TextRank	AbL15
20	-12.36%	12.68%	-8.80%	-4.82%	-1.67%	-7.20%	-12.36%
40	-5.24%	9.16%	4.68%	-3.74%	2.25%	-2.51%	-3.38%
60	-1.53%	6.94%	8.04%	-1.41%	3.45%	-0.59%	4.32%
80	0.95%	4.85%	7.63%	0.19%	3.81%	1.35%	10.16%
Avg. abs. dev.	5.02%	8.41%	7.29%	2.54%	2.80%	2.91%	7.56%

Table 4: Flesch Reading Ease statistics

%compression	Luhn	SumBasic	KLSum	ExL19	LexRank	TextRank	AbL15
20	6.98%	-5.25%	2.06%	9.67%	2.01%	3.30%	10.65%
40	4.69%	-3.29%	-3.60%	7.72%	0.68%	2.34%	6.62%
60	2.73%	-2.58%	-4.37%	6.03%	-0.52%	1.60%	2.91%
80	0.76%	-1.62%	-3.59%	4.28%	-0.79%	0.48%	0.05%
Avg. abs. dev.	3.79%	3.18%	3.41%	6.93%	1.00%	1.93%	5.06%

Table 5: Dale-Chall statistics

sentence and syllables per words; the higher the score, the easier is the text. Thus when the percent deviation is negative, the summary is less comprehensible than the original text. This occurs across almost all of the settings where the summary comprises only 20% of the original text. The only exception is SumBasic that, with higher compression rates[7], tends to select shorter words and shorter sentences. For this readability feature SumBasic simplifies summaries across all the compression rates. However, other frequency-based summarization systems show a different tendency: Luhn and ExL19 tend to generate more complex summaries with longer words and sentences for almost all the compression rates. Graph-based approach TextRank shows the same tendency. In turn, KLSum and AbL15 for higher compression rates generate more complex summaries and for the lower compression rates more simple ones. The degree of difficulty for compression rate 20% is almost the same as the degree of simplification for 80% rate. For example, consider AbL15 that for 20% rate generated 12.36% more complex summaries, while for 80% rate 10.16% more simple.

The overall average degree of deviation from the readability of the original document also needs to be taken into account. ExL19 has the lowest average absolute deviation. It generates summaries with closest complexity to the original (2.54%) across all the compression rates.

Contrary to the findings of Lloret et al. (2019), who showed that for the DUC 2002 corpus, ExL19 generated more comprehensible summaries with respect to FRE, for the BNE corpus, ExL19 tends to select longer words and sentences. We believe the variation in original document complexity is what causes this difference because the BNE coprus contains texts simplified for L2 learners ranging mostly from A2 to B2 CEFR levels, whereas DUC 2002 is comprised of unsimplified newswire documents that are instrinsically more complex. This may indicate that present analysis of selected summarization systems and their impact on readability cannot be extended to other domains beyond L2 learner materials, since the performance of any summarizer depends on the corpus.

6.2 Lexical readability results

Lexical complexity of summaries was evaluated with the help of Dale-Chall formula; lower values of this metric indicate easier to comprehend summaries. Results in Table 5 show that in this evaluation SumBasic again simplifies summaries across all the compression rates. KLSum reveals similar values, except for the setting with 20% compression rate. Luhn, ExL19 and AbL15 include sentences with high percentage of complex words in summaries. In terms of lexical complexity, summarizers based on graphs (LexRank and TextRank) demonstrate the lowest average absolute deviation across all the compression rates and thus maintain lexical complexity of summaries that are closest to the original documents. For the length-based readability metric, graph-based approaches were also among the most similar.

6.3 Syntactic readability results

Parse tree depth statistics can be found in Table 6. For this feature as well, higher values indicate

[7] Under "higher compression rates" we understand shorter summaries; in our study compression rate of 20% is the highest

%compression	Luhn	SumBasic	KLSum	ExL19	LexRank	TextRank	AbL15
20	3.13%	-6.38%	11.37%	-13.86%	3.47%	7.87%	-15.21%
40	3.03%	-3.70%	0.05%	-8.06%	0.71%	5.50%	-16.63%
60	2.33%	-2.48%	-2.01%	-4.78%	-0.20%	4.02%	-19.87%
80	1.13%	-1.02%	-1.73%	-2.53%	-0.17%	2.46%	-23.10%
Avg. abs. dev.	2.41%	3.40%	3.79%	7.31%	1.13%	4.96%	18.70%

Table 6: Parse tree depth statistics.

summaries with more complex syntactic constructions and negative percent of deviation indicates that the respective system selects syntactically less complex sentences. For this feature SumBasic again simplifies summaries with respect to the original documents; and graph-based LexRank, by maintaining the lowest average absolute deviation, preserves the original text complexity. For parse tree depth and the other two readability features, KLSum tends to simplify summaries on lower compression rates, but for the compression rate of 20% it generates more complex summaries. Due to the integrated syntactic simplification step, abstractive system AbL15 generates sentences with shorter parse trees across all the settings. It also displays the highest average absolute deviation to the original values. Extractive system ExL19 reveals a similar tendency, namely, while selecting more complex sentences in terms of lexical complexity, it tends to include syntactically more simple sentences in summaries.

7 Conclusion and Future Work

In this paper we experimented with a new dataset of level-annotated L2 learner texts that can be used both for text simplification and text summarization tasks. We analyzed its syntactic, lexical and length-based readability features and evaluated its level annotation with a machine learning system trained on data annotated by Cambridge exam annotators.

We further conducted a novel analysis on how different extractive and abstractive summarization techniques at different compression rates affect readability of simplified L2 learner texts. Our experiments showed how this impact varied depending on the system used: 1) frequency-based system SumBasic consistently simplified summaries with respect to the original texts across all the compression rates, and thus may be considered a competitive baseline, not only in terms of recall but also readability; 2) graph-based approaches, especially LexRank, tended to maintain the same complexity as the original document; 3) Luhn's classical frequency-based method generated more complex summaries; 4) KLSum method based on Kullback-Leibler divergence produced complex summaries at higher compression rates, while simplifying them at lower compression rates; 5) integration of anaphora resolution, textual entailment and word-sense disambiguation led to syntactically more simple, but lexically more complex summaries; 6) abstractive summarizer AbL15 oversimplified syntactic structures and maintained at the same time a high complexity of lexical readability component. Hence no common pattern among the summarization approaches was identified with respect to the effect of compression rate on readability.

This work has provided an insight on the behaviour of different summarization approaches and permitted the discovery of a necessary dataset, as well as the analysis of the dataset's readability. These findings can be viewed as the first important step for designing a summarization system aimed at people with different levels of language proficiency. Future lines of research will consider integrating second language acquisition and discourse readability metrics.

Acknowledgments

This research work has been partially funded by the University of Alicante (Spain), Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVING-LANG (RTI2018-094653-B-C22), INTEGER (RTI2018-094649-B-I00) and Red iGLN (TIN2017-90773-REDT). We would also like to thank Ekaterina Kochmar from the University of Cambridge for her help with the readability assessment of the BNE corpus, as well as Sean Banville for the permission to use his website materials and the valuable insights on the process of their creation.

References

María Aranzabe, Arantza Ilarraza, and Itziar Gonzalez-Dios. 2012. First Approach to Automatic Text Simplification in Basque. In *Proceedings of the Natural Language Processing for Improving Textual Accessibility (NLP4ITA) workshop (LREC 2012)*. pages 1–8.

Sean Banville. 2005. Creating ESL/EFL lessons based on news and current events. *The Internet TESL Journal* 11(9). http://iteslj.org/Techniques/Banville-News/.

Eduard Barbu, M. Teresa Martín-Valdivia, Eugenio Martínez-Cámara, and L. Alfonso Ureña López. 2015. Language technologies applied to document simplification for helping autistic people. *Expert Syst. Appl.* 42(12):5076–5086.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *In Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*. pages 7–10.

Jeanne Sternlicht Chall and Edgar Dale. 1995. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books.

Jonathan Charteris-Black. 2002. Second language figurative proficiency: A comparative study of Malay and English. *Applied linguistics* 23(1):104–133.

Kevyn Collins-Thompson and James P. Callan. 2004. A Language Modeling Approach to Predicting Reading Difficulty. In *HLT-NAACL 2004: Main Proceedings*. Association for Computational Linguistics, Boston, Massachusetts, USA, pages 193–200. https://www.aclweb.org/anthology/N04-1025.

Council of Europe. 2001. *The Common European Framework of Reference for Languages*. Cambridge University Press. https://rm.coe.int/1680459f97.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.* 22(1):457–479.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '09, pages 229–237.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology* 32(3):221. https://doi.org/10.1037/h0057532.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '09, pages 362–370.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Association for Computational Linguistics, Rochester, New York, pages 460–467. https://www.aclweb.org/anthology/N07-1058.

Tor G. Hultman and Margareta Westman. 1977. *Gymnasistsvenska*. Liber Lromedel.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, ANLC '00, pages 310–315. https://doi.org/10.3115/974147.974190.

Yushi Boni Li and Amanda Perkins. 2007. The impact of technological developments on the daily life of the elderly. *Technology in society* 29(3):361–368.

Elena Lloret, Ester Boldrini, Tatiana Vodolazova, Patricio Martínez-Barco, Rafael Muñoz, and Manuel Palomar. 2015. A novel concept-level approach for ultra-concise opinion summarization. *Expert Syst. Appl.* 42(20):7148–7156. https://doi.org/10.1016/j.eswa.2015.05.026.

Elena Lloret, Tatiana Vodolazova, Paloma Moreda, Rafael Muñoz, and Manuel Palomar. 2019. Are better summaries also easier to understand? Analyzing text complexity in automatic summarization. In Marina Litvak and Natalia Vanetik, editors, *Multilingual text analysis: Challenges, Models, and Approaches*, World Scientific, New Jersey, pages 337–369.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.* 2(2):159–165. https://doi.org/10.1147/rd.22.0159.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of EMNLP 2004*. Association for Computational Linguistics, Barcelona, Spain, pages 404–411. https://www.aclweb.org/anthology/W04-3252.

Kumaresh Nandhini and Sadhu Ramakrishnan Balasundaram. 2014. Extracting easy to understand summary using differential evolution algorithm. *Swarm and Evolutionary Computation* 16:19 – 27. https://doi.org/10.1016/j.swevo.2013.12.004.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005* 101.

Ethel Ong, Jerwin Damay, Gerard Lojico, Kimberly Lu, and Dex Tarantan. 2008. Simplifying text in medical literature. *Journal of Research in Science, Computing and Engineering* 4(1):37–48.

Sarah E. Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2008*. ACL, pages 186–195. http://www.aclweb.org/anthology/D08-1020.

Luz Rello, Clara Bayarri, Azuki Gòrriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and Vasile Topac. 2012. Dyswebxia 2.0!: more accessible text for people with dyslexia. *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility* pages 41–44. https://doi.org/10.1145/2140446.2140455.

Luz Rello, Horacio Saggion, and Ricardo Baeza-Yates. 2014. Keyword highlighting improves comprehension for people with dyslexia. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 30–37. https://www.aclweb.org/anthology/W14-1204.

S. Rebecca Thomas and Sven Anderson. 2012. Wordnet-based lexical simplification of a document. In Jeremy Jancsary, editor, *Proceedings of KONVENS 2012*. ÖGAI, pages 80–88.

Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, New Orleans, Louisiana, pages 297–304. https://doi.org/10.18653/v1/W18-0535.

Jurgita Vizgirdaite. 2009. Filling the child-parent relationship gap via the parent self-education and intergenerational education on internet slang. *Social Sciences (1392-0758)* 64(2).

Sanja Štajner, Richard Evans, Constantin Orasan, and Ruslan Mitkov. 2012. What can readability measures really tell us about text complexity? In Luz Rello and Horacio Saggion, editors, *Proceedings of the LREC'12 Workshop: Natural Language Processing for Improving Textual Accessibility (NLP4ITA)*. European Language Resources Association (ELRA), Istanbul, Turkey.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 409–420. https://www.aclweb.org/anthology/D11-1038.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, San Diego, CA, pages 12–22. https://doi.org/10.18653/v1/W16-0502.

The Impact of Rule-Based Text Generation on the Quality of Abstractive Summaries

Tatiana Vodolazova, Elena Lloret
Dept. of Software and Computing Systems
University of Alicante
Apdo. de Correos 99
E-03080, Alicante, Spain
{tvodolazova,elloret}@dlsi.ua.es

Abstract

In this paper we describe how an abstractive text summarization method improved the informativeness of automatic summaries by integrating syntactic text simplification, subject-verb-object concept frequency scoring and a set of rules that transform text into its semantic representation. We analyzed the impact of each component of our approach on the quality of generated summaries and tested it on DUC 2002 dataset. Our experiments showed that our approach outperformed other state-of-the-art abstractive methods while maintaining acceptable linguistic quality and redundancy rate.

1 Introduction

Rapid growth of digital information increases the need for automatic text summarization methods that can digest large amounts of textual data, such as scientific articles, blogs and news articles to extract concise and relevant information from them. Text summarization methods can be classified into abstractive and extractive ones (Nenkova and McKeown, 2012). Extractive methods compose summaries from the most salient sentences of the original document. In contrast, abstractive methods generate novel or partially novel text using such techniques as sentence compression, fusion, calculation of path scores in graphs or, natural language generation tools such as SimpleNLG (Gupta and Gupta, 2018). They involve an intermediate step of deep linguistic analysis and an abstract semantic representation of the data. Extractive techniques have been intensively researched for over half a century and, according to some studies, “have more or less achieved their peak performance” (Mehta, 2016).

Over the past few years interest in the field of text summarization has shifted towards abstractive methods and quickly produced a large variety of approaches. Gupta and Gupta (2018) classify them broadly into methods based on the structure, semantics and deep learning with neural networks.

The main advantage of semantic-based approaches over deep learning ones lies in their independence from a large training corpus. Most of the available datasets for deep learning belong to the domain of news text that further restricts the application of these methods to other domains. However, semantic-based approaches rely on a parser to transform text to its semantic representation and, therefore, a poor parser performance will reduce the quality of generated summaries. Another limitation of the deep learning methods comes from the fact that they rely on statistical co-occurrence of words and are prone to semantic and grammatical errors. This is something that a reliable parser could help to avoid.

Structure-based methods, such as template and ontology based ones reveal other weaknesses. Template-based methods lack diversity. At the same time, ontology based ones rely on a time-consuming task of creating an ontology by a human expert. However, they provide highly coherent summaries and can handle uncertainties respectively. Semantic-based approaches that rely on handcrafted rules to transform text into semantic representation may be criticized for the same reason related to the human effort and time required to solve the laborious task of creating transformation rules.

It becomes clear that each abstractive approach can reliably handle only some aspects of the summarization process while revealing weaknesses in the remaining ones. Thus far, none of the approaches has been capable of offering a broad-based solution. Research in this field is mak-

Proceedings of Recent Advances in Natural Language Processing, pages 1275–1284,
Varna, Bulgaria, Sep 2–4 2019.

ing headway, each time with more elaborate algorithms and combining techniques from a number of different methods. However, Chen et al. (2016) have shown via their analysis of the reading comprehension task – another natural language processing task that requires interpretation of the text – that a straightforward approach designed around a small set of carefully selected features can obtain high, state-of-the-art accuracy.

Therefore, this study has a threefold objective. First, to design a broad-based abstractive text summarization method. Second, to evaluate whether the proposed method is capable of delivering concise and informative summaries while maintaining above-average linguistic quality and redundancy rate. Third, to compare it against other state-of-the-art abstractive methods.

The approach that we propose in this work falls into the previously mentioned semantic-based group of abstractive summarization approaches and has been inspired by the ideas of Genest and Lapalme (2011) and Lloret et al. (2015). Our contribution takes their abstractive models one step further by scoring abstract information representation without taking into account its surface representation. The proposed method incorporates syntactic text simplification, subject-verb-object concept frequency scoring, and a set of rules that transform text into its semantic representation.

This paper is structured as follows: Section 2 discusses related semantic-based abstractive summarization approaches. Section 3 describes in detail the architecture of our method. Evaluation methods and results are presented in Section 4. Section 5 describes the effect of individual components of our approach upon the quality of generated summaries. Section 6 provides a summary of the conclusions and areas for future work.

2 Related Work

All the methods in the semantic-based abstractive summarization group include the initial step of converting texts into an abstract semantic representation. For example, Genest and Lapalme (2011) introduced the concept of information item that was defined as a smallest element of coherent information and represented as a dated and located subject-verb-object triplet. Lloret et al. (2015) also base their concept representation on subject-verb-object triplets. Alshaina et al. (2017) use predicate-argument structure as their underlying information representation and extract a number of features from it that are later used for ranking. Li (2015) define the concept of Basic Semantic Unit (BSU) where each BSU is an actor-action-receiver triplet with its obligatory arguments, namely, actor and receiver of the action. The BSUs are used to construct a BSU semantic link network representation for each text.

Abstract Meaning Representation (AMR) graphs is the most recent approach to abstract semantic representation of texts (Vilca and Cabezudo, 2017). AMR nodes are represented by either words or PropBank[1] frames, and edges define relationships between them. Both the AMR graph representation and the subject-verb-object (SVO) representation depend on the efficiency of the parser. However, AMR graphs also rely on PropBank framework whose limitations pose additional constraints on AMR graphs. Furthermore, the problem of text generation from AMR graphs is still a challenge and it has not yet been solved (Li, 2015).

The summarization method based on BSUs proposed by Li (2015) overcomes the limitation of text generation faced by AMR graphs, and produces informative, coherent and compact summaries. However, as the authors state, the BSU network cannot yet handle data that express opinions rather than facts and actions, since these cases involve verbs that lack meaningful actions, such as ’be’, and the underlying representation of actor-action-receiver cannot be appropriately computed.

Alshaina et al. (2017) use K-means and agglomerative hierarchical clustering algorithms to group similar predicate-argument structures (PAS) based on semantic similarity measures, and to eventually select the most representative PAS based on a weighted set of 12 features. The PAS proposed by this approach are classified into simple and complex ones. Complex PAS are derived from sentences with multiple verbs, otherwise they are considered to be simple. Nested PAS are eliminated. One of the features that determines whether to include a PAS into the summary or not is the ”number of verbs and nouns” that gives preference to complex PAS as crucial to summary generation.

Lloret et al. (2015) propose an abstractive semantic-based approach to ultra-concise opinion summarization. It involves a syntactic sentence simplification in the preprocessing step and

[1] https://propbank.github.io/

semantic representation based on subject-verb-object triplets. Their scoring heuristics relies on subject-verb-object term frequencies.

The approaches closest to ours are those of Lloret et al. (2015); Genest and Lapalme (2011, 2010). However, the difference between them is twofold. First, the aforementioned systems use term or document frequencies for scoring. We integrate word sense disambiguation to identify similarities between subject-verb-object triplets on the conceptual level that allows us to introduce concept frequencies for scoring. Second, the architecture of our approach is characterized by a higher level of abstraction. Namely, our approach scores abstractive concepts represented in the form of enriched subject-verb-object triplets and not their surface representation. Their surface representation is integrated in the final step when all the triplets have already been assigned their score.

Unlike the approach of Alshaina et al. (2017) who give preference to sentences with more than one verb, our approach integrates syntactic sentence simplification in the preprocessing step in order to split complex sentences into simpler ones and ideally reduce syntactic structure to a single main verb. This allows us to generate various subject-verb-object triplets from a single sentence and to manipulate them in a more precise manner.

3 Abstractive Summarization Framework

The architecture of our proposed abstractive text summarization approach is illustrated in Figure 1. This section describes the role and the implementation of each of its components.

Simplification. We begin by applying syntactic simplification to the original document as a pre-processing step. Simplification targets only complex sentences, splitting their syntactic trees into simpler ones. Each newly created sentence is a fully grammatical construction that, not always but in most cases, contains one main verb and covers one single concept[2]. In the next stages our method generates an information item from each simplified sentence. Simplifying the syntactic structure of the input text allows us to have fewer, less recursive and less error-prone rules for information item extraction. And capturing as many concepts as possible benefits the process of information item selection: only the most salient bits of information are selected while the irrelevant ones are discarded. We use the Factual Statement Extractor to carry out the simplification task (Heilman and Smith, 2010).

Analysis. In this stage, we perform a linguistic analysis decomposing each supplied simplified sentence into lemmas, stems, parts of speech, senses, named entities, syntactic roles and noun phrases. This is done mainly with the help of Stanford CoreNLP (Manning et al., 2014). Additionally we use Porter stemmer for stemming (Porter, 1997), Freeling for word sense disambiguation (Padró and Stanilovsky, 2012) and Java DOM parser for noun phrase chunking.

Information Items Generation. Once the data have been analyzed we proceed to build an abstract representation of each of the sentences. We adopt the same naming convention as Genest and Lapalme (2011) and refer to them as information items (`InIts`). At the core of each `InIt` lies the main verb of the sentence accompanied by its subject and object, if they are present. Contrary to Genest and Lapalme (2011) we do not incorporate any manual rules to reject candidate `InIts`. However, a small portion of them will be lost during the surface realization stage if SimpleNLG fails to generate a sentence from an `InIt`. It happens at most to 1-2 simplified sentences per document. Preserving all `InIts` may introduce a higher rate of grammatically incorrect sentences due to the incorrect sentence parses[3]. However, since no clear pattern between syntactic linguistic phenomena and incorrect parses was observed, we could not discard such cases. Additionally, we extend the core subject-verb-object structure to include open clausal complements and prepositional phrases. Since the Stanford CoreNLP configuration that we used implements Universal Dependencies [4] for dependency parsing, our rules for transforming text into `InIts` are also designed around this annotation scheme. We implemented 5 transformation rules:

1. `ccomp` rule retains a clausal complement of a verb or adjective, rejecting the initial part. *He says that [you like to swim].*
2. `subject` and `verb` rule identifies them in the remaining sentence. It also handles copula and passive voice.

[2] Table 9 provides an example of a simplified sentence.

[3] Common mistakes provoked by this decision can be found in Section 4.2.

[4] https://universaldependencies.org/

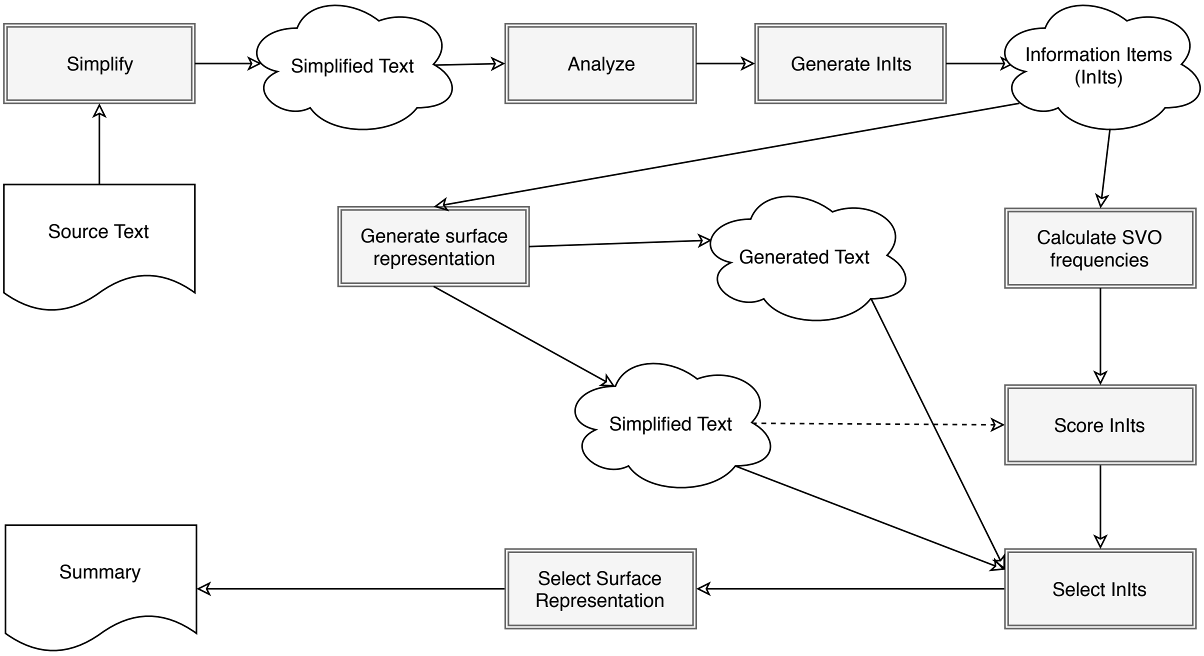


Figure 1: Our Abstractive Summarization Framework.

3. `direct` and `indirect object` rule sets corresponding objects if they exist.
4. `xcomp` rule handles open clausal complements of verbs and adjective. *She looks [very beautiful]. I consider him [a fool]. He tried [to run].*
5. `pp` rule identifies remaining prepositional phrases. *They talked [about London].*

All the `InIts` are stored internally as an ordered list.

Calculation of frequencies. In this stage we analyze `InIts` and calculate concept frequencies of all the verb, subject and object phrase heads of the input. For the purpose of evaluating effectiveness of concept frequencies, we also incorporated their term frequency scoring for comparative purposes. Our scoring strategy is based on the idea that there is "a very strong correlation between concepts of topic and subject in English." (Foley, 1994). And it has also been shown in previous research on text summarization that subjects, verbs and objects play a crucial role in content selection and cannot be dropped (Harabagiu and Lacatusu, 2010). Along with the SVO frequencies we also calculate term frequencies of named entities that represent subject or object phrase heads.

Information Items Scoring. Unlike the approaches of Genest and Lapalme (2011); Lloret et al. (2015) in our approach, `InIt` scoring and surface realization are independent from each other. We apply extracted SVO and named entity head frequencies from the previous step to score `InIts` directly. This gives us the flexibility to choose which parts of `InIts` to use for scoring. Our scoring is based on the idea that `InIts` that cover the main topic of the document contain the most frequent SVO concepts and named entities in any of their components. Given the flexibility to work with `InIts` directly and not the raw text, we experimented with scoring on SVO components and also combined them with open clausal complements and prepositional phrases. While scoring, we calculate matches not only between candidate noun phrase heads, but other phrase constituents es well.

For testing purposes we also integrate a modification of this step that, instead of scoring `InIts` directly, applies SVO and named entity frequencies to the simplified text. This configuration is indicated with the dashed arrow in Figure 1. It allows us to compare how much information is lost during the transformation and generation stages.

Text Generation. We generate sentences from `InIts` with the help of SimpleNLG realization engine (Gatt and Reiter, 2009). The order of text generation rules is defined mainly by functionalities of SimpleNLG and follows these steps:

- generate a noun phrase (NP) to represent the subject if present;
- generate the main verb;

- generate an NP to represent direct object if present;
- generate an NP for indirect object if present;
- generate prepositional phrases
- generate open clausal complements if present (xcomp transformation rule); and,
- assemble all the components and generate the verb phrase (VP).

We do not do any other modifications apart from syntactic simplification of long sentences in the preprocessing step of our approach. This means, for example, that we do not convert passive constructions into active ones. However, since we always use the same order for the text generation rules, the original order of constituents may be changed, i.e. prepositional phrases will always be generated after subject, verb or objects, despite the fact that in the original sentence they may be in a different position. Generated sentences play no role in InIts scoring or InIt selection. They remain on hold until the selection of InIts and surface representation stage.

Information Items Selection. At this stage, we inspect all the InIts and reject the ones with empty text representation generated by SimpleNLG.

Selection of Surface Representation. For all the remaining ranked InIts, starting from the highest ranked one, we add each InIt's surface representation to the final summary until the maximum allowed size has been reached. Once we reach it we reorder sentences to preserve the original order of simplified sentences that each InIt originated from and deliver the summary. For surface representation our approach allows the selection of either a representation generated with SimpleNLG or the simplified sentence. In this final stage we do not integrate additional date or location information as Genest and Lapalme (2011), but if an InIt contained them among its prepositional phrases, they are included into the generated sentence by SimpleNLG.

4 Evaluation

Our approach is evaluated on DUC 2002 dataset for the single document summarization task[5]. After discarding duplicates, the dataset consists of 530 newswire articles. Each article is accompanied by one or more manually created abstractive model summaries of approximately 100 words.

At this development phase, our approach generates summaries operating exclusively with the words present in the original text. However, as a result of the syntactic simplification, they are likely to be reorganized into shorter sentences. Moreover, some of the words are ordered differently or not included into generated sentences as a consequence of the implemented translation and surface realization rules. These operations create summaries that go beyond the literal extraction of original text fragments.

We evaluate the content selection part of our approach with ROUGE toolkit and use human evaluation to assess the linguistic quality of generated summaries as described in Sections 4.1 and 4.2 respectively.

4.1 Informativeness

Following the example of recent works on abstractive text summarization we used ROUGE toolkit (Lin, 2004) to evaluate generated summaries (Vilca and Cabezudo, 2017; Hsu et al., 2018). ROUGE-1 and ROUGE-2 are used to assess informativeness and together with ROUGE-SU4 they have been found to correlate well with human judgement. The longest common subsequence ROUGE-L is used to assess fluency. We compared our summaries to the human summaries provided for DUC 2002 corpus, and each text can be evaluated against at least 2 of them.

We also calculated average pairwise ROUGE values for human summaries to identify the highest score that an abstractive summary can obtain with ROUGE (see Table 1).

The selected baseline was implemented with the help of our method such that each original text passes through all the stages specified in Section 3, including sentence simplification and surface realization stages but avoiding the SVO and named entity scoring. To produce the baseline summary we applied tf-scoring to such regenerated sentences. This ensures that the baseline is an abstractive summary only differing in the scoring method.

We compared our approach to two state-of-the-art approaches for abstractive text summarization of a different nature: 1) Vilca and Cabezudo's (2017) approach based on AMR graphs and Rhetorical Structure Theory; and, 2) the approach proposed by Hsu et al. (2018) based on deep learning and combines abstractive and extractive components. To compare our approach with the latter

[5] http://duc.nist.gov/

	R-1	**R-2**	**R-L**	**R-SU4**
Human	0.507	0.218	0.460	0.239
Ours	**0.410**	0.154	**0.378**	**0.180**
Baseline	0.378	0.138	0.351	0.163
Hsu'18 abs	0.266	0.116	0.239	0.126
Vilca'17	0.244	**0.231**	-	0.033

Table 1: ROUGE scores for different summarization methods.

one, we used their abstractive model pre-trained on CNN/Daily Mail dataset of newswire articles.

Table 1 shows that our approach outperforms both the abstractive baseline and the approach of Hsu et al. (2018) on all the ROUGE metrics. It also outperforms Vilca and Cabezudo's (2017) approach on 3 of the 4 metrics.

To illustrate how our approach and the approach of Hsu et al. (2018) modify original sentences, we contrast an extractive term-frequency based summary with the abstractive summaries generated by both of the approaches (see Table 2). For convenience, the common chunks between the summaries are numbered and surrounded by square brackets, while the unique chunks are italicized.

Our approach: [More than 4,000 workers at a coal mine in the southern city of Jastrzebie went to demand legalization of Solidarity and higher wages on strike]^{1}. [Workers on the overnight shift at the Manifest Lipcowy mine stayed outside the mine shaft]^{2}. [The miners are demanding the legalization of Solidarity]^{4}. *The workers are calling for higher wages and better working conditions. The workers are requesting two lawyers and two economists. Workers at the Rudna copper mine near the city of Wroclaw staged a protest rally.*[Workers at factories around the northern port of Gdansk joined striking shipyard workers.]^{5}.
Extractive TF summary: *Solidarity spokeswoman Katarzyna Ketrzynska said* [[workers on the overnight shift at the Manifest Lipcowy mine stayed outside the mine shaft]^{2} all night and were joined by workers arriving for the morning shift]^{3}. *The strike began at noon today, according to Katrzynska. She said* [the miners are demanding the legalization of Solidarity]^{4} *and reinstatement of workers fired for union activities. Three members of Solidarity were barred Saturday from working. On Aug. 16, 1980,* [workers at factories around the northern port of Gdansk joined striking shipyard workers]^{5} *to form Solidarity, the first and only independent trade federation in the Soviet bloc.*
Hsu'18: [more than 4,000 workers at a coal mine in the southern city of jastrzebie went on strike today to demand legalization of solidarity and higher wages]^{1}. [[workers on the overnight shift at the manifest lipcowy mine stayed outside the mine shaft]^{2} all night and were joined by workers arriving for the morning shift]^{3}.

Table 2: An comparison of abstractive summaries with an extractive summary.

Grammaticality: 1. TAS **gave not** details of Gorbachev 's suggestion. 2. Six bodies were *founded* in the hull of the ferry by Police. 3. The Lone Star Statuette *were* built by Chicago 's Creative House Promotions.
Redundancy: 1. Martin Nelson was another meteorologist at the center *at center*. 2. Dullah Omar was an activist and family friend of the Mandelas *of Mandelas*. 3. A resolution promises reforms. A resolution promises reforms.
Completeness: 1. A quake of 6 on the scale is capable. 2. Reunification mishandled. 3. Arthur Andersen wanted.

Table 3: Examples of some of the mistakes produced by our approach.

4.2 Human Evaluation

For our preliminary human evaluation of generated summaries, we used the statistical formula to calculate the correct size of a representative sample that was proposed by Pita-Fernández (1996) and successfully applied to different NLP tasks (Vázquez et al., 2010; Lloret et al., 2019). For DUC 2002 dataset, a representative sample consists of 77 documents that we randomly chose from the corpus. They were evaluated according to the following criteria based on the DUC guidelines, but adapted to the specific task and errors:

- *grammaticality* - grammatical correctness of the summary (i.e. number agreement);
- *non-redundancy* - no unnecessary repetitions; and,
- *completeness* - completeness of grammatical construction (i.e. a missing direct object of a transitive verb).

The generated abstractive summaries were assessed on a five-point Likert scale by 3 external annotators without any knowledge about how the summaries were produced. A grammatically correct, non-redundant and complete summary would receive a score of 5-5-5 respectively. The results in Table 4 show that the summaries produced by our approach scored above the average on the three criteria.

Measure	**Score**
Grammaticality	3.60
Non-redundancy	3.71
Completeness	3.81

Table 4: Average scores for human evaluation.

Table 3 shows examples of such mistakes. Upon closer inspection we detected that the completeness errors are often caused by incorrect parses. Some of the grammatical errors are produced by SimpleNLG, whereas others refer to the cases not covered by information item extraction rules. Contrary to our predictions, the non-redundancy rate was above the average. The overall linguistic quality looks promising and reveals areas for improvement. However, there is a need for a deeper evaluation that is planned for future work.

5 Further Experiments and Discussion

In this section we analyze the impact of each of the components of our method on the informativeness of generated summaries.

5.1 Syntactic Constituents

We experimented with different configurations of our scoring module to test whether the subject, verb and object are enough for the scoring, or should be extended with open clausal complements and prepositional phrases to improve its performance. For this purpose we applied the scoring in three different contexts: exclusively SVO (SVO); the SVO extended with clausal complements (SVO+xComp); and, the SVO+xComp extended with prepositional phrases (SVO+xComp+PPs). This means that the scoring module checked occurrences of the most frequent SVO elements only in subject-object-object triplets or in the extended structures. The results in Table 5 show that there is some improvement in performance when additional syntactic components are included. We believe that this improvement may increase as the corpus increases, since a larger corpus will contain more cases of open clausal complements and prepositional phrases.

	R-1	R-2	R-L
SVO	0.4064	0.1522	0.3756
SVO xComp	0.4078	0.1523	0.3765
SVO xComp PPs	0.4102	0.1544	0.3776

Table 5: ROUGE scores for syntactic components

5.2 Generation and Recall

Another experimental setup addresses the question of how much important information is lost during the generation stage. As described in Section 3, we integrated a setting (signaled with the dashed arrow in Figure 1) that, instead of scoring InIts, applied the SVO frequencies to the simplified text and delivered it in the final summary. This setting overcomes two possible limitations of our approach: it also scores the parts of the sentence that are not included into an InIt and provides more text for the future recall evaluation with ROUGE. Results in Table 6 show a slight improvement over the InIt-based scoring, but the difference is not as high as we expected. We may conclude that our InIt extraction rules capture most of the information, and surface realization rules generate sufficient material for the ROUGE evaluation.

	R-1	R-2	R-L
InIt	0.4102	0.1544	0.3776
Simpl. text	0.4181	0.1668	0.3797

Table 6: ROUGE evaluation of text-based scoring

5.3 Effect of Concept Frequency Scoring

Word sense disambiguation and the resulting concept scoring should positively affect InIt selection as well. Table 7 shows that in this setting the difference between term and concept frequencies is almost non-existent. We believe that if we integrate the entire noun phrase when calculating SVO frequencies and not only the noun phrase head, it may lead to a more significant difference.

	R-1	R-2	R-L
SVO cf	0.4102	0.1544	0.3776
SVO tf	0.4100	0.1545	0.3777

Table 7: ROUGE scores for concept and term frequency scoring.

5.4 Simplification and Recall

Our motivation behind the integration of a syntactic simplification module was to reach a greater degree of concept granularity that would allow us to select only the most salient InIts while discarding the less relevant ones. We tested our approach both with and without simplification. The results revealed in Table 8 indicate that working with original text yields a slightly better recall.

Close inspection showed that our simplification module generates syntactically more simple sentences, but introduces more repetitions that are

	R-1	R-2	R-L
Simplified	0.4102	0.1544	0.3776
Original	0.4169	0.1588	0.3803

Table 8: ROUGE scores for simplification test.

picked up by the SVO and named entity scoring. Consider the example in Table 9:

Original sentence: Greek marine archaeologists focus on locating and surveying historic wrecks scattered around the Aegean and rarely carry out excavations.
Simplified: 1. Greek marine archaeologists focus on locating. 2. Greek marine archaeologists focus on surveying historic wrecks scattered around the Aegean. 3. Greek marine archaeologists carry out excavations.
Simplified summary: 1. Greek marine archaeologists focus. 2. Greek marine archaeologists carry out excavations.
Original summary: not included

Table 9: Simplification example.

When we split a long sentence into several shorter ones with the repeated subject, the scoring module gives them more importance by considering the repeated subject to be the topic of the document. If some of these split sentences are included in the final summary, the repeated subject noun phrase takes summary space that otherwise could be occupied by a different phrase. On the other hand if the subject of such a split phrase is the true topic of the document, our method generates a very topic-focused summary. We hypothesize that scoring should be performed on the original subject-verb-object distribution of the document so as to avoid scoring for repeated subjects.

5.5 Summary Readability

Readability is rarely studied in detail in the context of automatic text summarization. Our summarization approach integrates syntactic simplification that results in syntactically simpler summaries and concept frequency scoring that may yield summaries with richer vocabulary when compared to term frequency based ones. To assess readability of generated summaries we calculated their Flesch Reading Ease (FRE), Dale-Chall (DC) and depth of the parse tree (PTD) scores. These three metrics give us a quick but complete assessment of the length, vocabulary and syntactic complexity-based readability aspects. Higher FRE and lower PTD and DC values correspond more comprehensible texts.

Results in Table 10 show that human summaries include longer sentences and words, and are also more concept dense than the original texts. Human summaries also tend to consist of syntactically less complex sentences. Unlike human summaries, our approach generates more comprehensible texts in terms of sentence and word length. As expected, syntactic sentence simplification positively affects the parse tree depth metric. However, it also generates summaries with greater lexical density.

	FRE	**DC**	**PTD**
Ours	50.74	10.56	8.30
Human	42.76	10.45	10.51
Original	43.51	10.13	11.48

Table 10: Readability metrics for different methods.

6 Conclusions and Future Work

This paper presents a broad-based abstractive text summarization method that outperforms other state-of-the-art abstractive approaches while maintaining acceptable linguistic quality and redundancy rate. Our approach is based on the set of syntactic rules that transform text into its semantic representation as well as the combination of subject-verb-object concept frequency and named entity frequency for scoring.

The results show that some aspects of the proposed approach require improvement. Integration of the entire subject and object noun phrases for the calculation of frequencies may increase informativeness of the generated summaries. Coreference resolution and sentence fusion may help to lower the degree of redundancy introduced through the syntactic sentence simplification.

In future work, we plan to integrate these improvements and to evaluate our method on other datasets such as CNN/Daily Mail dataset. First, a larger dataset can provide more insights on the relative importance of open clausal complements, prepositional phrases and concept frequency for information item rating. Second, it will allow us to gauge the weaknesses and strengths of our approach, which is based on the concept of information items and handcrafted syntactic transformation rules, via a comparative analysis with state-of-the-art deep learning and semantic graph approaches.

Acknowledgments

This research work has been partially funded by the University of Alicante (Spain), Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVING-LANG (RTI2018-094653-B-C22), INTEGER (RTI2018-094649-B-I00) and Red iGLN (TIN2017-90773-REDT).

References

S. Alshaina, A. John, and A. G. Nath. 2017. Multi-document abstractive summarization based on predicate argument structure. In *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*. pages 1–6. https://doi.org/10.1109/SPICES.2017.8091339.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2358–2367. https://doi.org/10.18653/v1/P16-1223.

William A Foley. 1994. Information structure. In Roland E. Asher and Joy M. Y. Simpson, editors, *The Encyclopedia of Language and Linguistics*, Pergamon Press, Oxford, volume 3, pages 1678–1685.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Stroudsburg, PA, USA, ENLG '09, pages 90–93.

Pierre-Etienne Genest and Guy Lapalme. 2010. Text generation for abstractive summarization. In *Proceedings of the Third Text Analysis Conference*. National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*. Association for Computational Linguistics, Portland, Oregon, pages 64–73. https://www.aclweb.org/anthology/W11-1608.

Som Gupta and S.K. Gupta. 2018. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications* 121:49–65. https://doi.org/10.1016/j.eswa.2018.12.011.

Sanda Harabagiu and Finley Lacatusu. 2010. Using topic themes for multi-document summarization. *ACM Trans. Inf. Syst.* 28(3):13:1–13:47. https://doi.org/10.1145/1777432.1777436.

Michael Heilman and Noah A Smith. 2010. Extracting simplified statements for factual question generation. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 11–20.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. ACL, Melbourne, Australia, volume 1, pages 132–141. https://aclanthology.info/papers/P18-1013/p18-1013.

Wei Li. 2015. Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1908–1913. https://doi.org/10.18653/v1/D15-1219.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, pages 74–81. https://www.aclweb.org/anthology/W04-1013.

Elena Lloret, Ester Boldrini, Tatiana Vodolazova, Patricio Martínez-Barco, Rafael Muñoz, and Manuel Palomar. 2015. A novel concept-level approach for ultra-concise opinion summarization. *Expert Systems with Applications* 42(20):7148–7156. https://doi.org/10.1016/j.eswa.2015.05.026.

Elena Lloret, Tatiana Vodolazova, Paloma Moreda, Rafael Muñoz, and Manuel Palomar. 2019. Are better summaries also easier to understand? Analyzing text complexity in automatic summarization: Challenges, models, and approaches. In Marina Litvak and Natalia Vanetik, editors, *Multilingual text analysis challenges, models, and approaches*, World Scientific, New Jersey, pages 337–369. https://doi.org/10.1142/9789813274884_0010.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Parth Mehta. 2016. From extractive to abstractive summarization: a journey. In *Proceedings of the ACL 2016 student research workshop*. pages 100–106. https://www.aclweb.org/anthology/P16-3015.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining text data*, Springer, pages 43–76. https://doi.org/10.1007/978-1-4614-3223-4_3.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*. ELRA, Istanbul, Turkey. https://www.aclweb.org/anthology/papers/L/L12/L12-1224/.

Salvador Pita-Fernández. 1996. Determinación del tamaño muestral. *Cadernos de atención primaria* 3(3):138–141.

M. F. Porter. 1997. An algorithm for suffix stripping. In Karen Sparck Jones and Peter Willett, editors, *Readings in Information Retrieval*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 313–316.

Yoan Gutiérrez Vázquez, Antonio Fernández Orquín, Andrés Montoyo Guijarro, and Sonia Vázquez Pérez. 2010. Integración de recursos semánticos basados en wordnet. *Procesamiento del lenguaje natural* 45:161–168.

Gregory César Valderrama Vilca and Marco Antonio Sobrevilla Cabezudo. 2017. A study of abstractive summarization using semantic representations and discourse level information. In Kamil Ekštein and Václav Matoušek, editors, *Text, Speech, and Dialogue*. Springer International Publishing, Cham, pages 482–490. https://doi.org/10.1007/978-3-319-64206-2_54.

ETNLP: A Visual-Aided Systematic Approach to Select Pre-Trained Embeddings for a Downstream Task

Xuan-Son Vu[1], **Thanh Vu**[2], **Son N. Tran**[3], **Lili Jiang**[1]
[1]Umeå University, Sweden
[2]The Australian E-Health Research Centre, CSIRO, Australia
[3] The University of Tasmania, Australia;
{sonvx, lili.jiang}@cs.umu.se
thanh.vu@csiro.au, sn.tran@utas.edu.au ;

Abstract

Given many recent advanced embedding models, selecting pre-trained word embedding (a.k.a., word representation) models best fit for a specific downstream task is non-trivial. In this paper, we propose a systematic approach, called *ETNLP*, for extracting, evaluating, and visualizing multiple sets of pre-trained word embeddings to determine which embeddings should be used in a downstream task.

We demonstrate the effectiveness of the proposed approach on our pre-trained word embedding models in Vietnamese to select which models are suitable for a named entity recognition (NER) task. Specifically, we create a large Vietnamese word analogy list to evaluate and select the pre-trained embedding models for the task. We then utilize the selected embeddings for the NER task and achieve the new state-of-the-art results on the task benchmark dataset. We also apply the approach to another downstream task of privacy-guaranteed embedding selection, and show that it helps users quickly select the most suitable embeddings. In addition, we create an open-source system using the proposed systematic approach to facilitate similar studies on other NLP tasks. The source code and data are available at `https://github.com/vietnlp/etnlp`.

1 Introduction

Word embedding, also known as word representation, represents a word as a vector capturing both syntactic and semantic information, so that the words with similar meanings should have similar vectors (Levy and Goldberg, 2014). Although, the classical embedding models, such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), fastText (Bojanowski et al., 2017), have been shown to help improve the performance of existing models in a variety of Natural Language Processing (NLP) tasks like parsing (Bansal et al., 2014), topic modeling (Nguyen et al., 2015), and document classification (Taddy, 2015; Vu et al., 2018b). Each word is associated with a single vector leading to a challenge on using the vector across linguistic contexts (Peters et al., 2018). To handle the problem, recently, contextual embeddings (e.g., ELMO of Peters et al. (2018), BERT of Devlin et al. (2018)) have been proposed and help existing models achieve new state-of-the-art results on many NLP tasks. Different from non-contextual embeddings, ELMO and BERT can capture different latent syntactic-semantic information of the same word based on its contextual uses. Therefore, for completeness, in this paper, we incorporate both classical embeddings (i.e., Word2Vec, fastText) and contextual embeddings (i.e., ELMO, BERT) to evaluate their performances on NLP downstream tasks.

Given the fact that there are many different types of word embedding models, we argue that having a systematic pipeline to evaluate, extract, and visualize word embeddings for a downstream NLP task, is important but non-trivial. However, to our knowledge, there is no single comprehensive pipeline (or toolkit) which can perform all the tasks of evaluation, extraction, and visualization. For example, the recent framework called *flair* (Akbik et al., 2018) is used for training and stacking multiple embeddings but does not provide the whole pipeline of extraction, evaluation and visualization.

In this paper, we propose *ETNLP*, a systematic pipeline to extract, evaluate and visualize the pre-trained embeddings on a specific downstream NLP task (hereafter ETNLP pipeline). The ETNLP pipeline consists of three main components which are *extractor*, *evaluator*, and *visualizer*. Based on the vocabulary set within a downstream task, the extractor will extract a subset of word embeddings for the set to run evaluation

Proceedings of Recent Advances in Natural Language Processing, pages 1285–1294,
Varna, Bulgaria, Sep 2–4 2019.

and visualization. The results from both *evaluator* and *visualizer* will help researchers quickly select which embedding models should be used for the downstream NLP task. On the one hand, the *evaluator* gives a concrete comparison between multiple sets of word embeddings. While, on the other hand, the *visualizer* will give the sense on what type of information each set of embeddings preserves given the constraint of the vocabulary size of the downstream task. We detail the three main components as follows.

- **Extractor** extracts a subset of pre-trained embeddings based on the vocabulary size of a downstream task. Moreover, given multiple sets of pre-trained embeddings, how do we get the advantage from a few or all of them? For instance, if people want to use the character embedding to handle the out-of-vocabulary (OOV) problem in Word2Vec model, they have to implement their own extractor to combine two different sets of embeddings. It is more complicated when they want to evaluate the performance of either each set of embeddings separately or the combination of the two sets. The provided **extractor** module in ETNLP will fulfill those needs seamlessly to elaborate this process in NLP applications.

- **Evaluator** evaluates the pre-trained embeddings for a downstream task. Specifically, given multiple sets of pre-trained embeddings, how do we choose the embeddings which will potentially work best for a specific downstream task (e.g., NER)? Mikolov et al. (2013) presented a large benchmark for embedding evaluation based on a series of analogies. However, the benchmark is only for English and there is no publicly available *large* benchmark for low resource languages like Vietnamese (Vu et al., 2014). Therefore, we propose a new evaluation metric for the word analogy task in Section 3.

- **Visualizer** visualizes the embedding space of multiple sets of word embeddings. When having a new set of word embeddings, we need to get a sense of what kinds of information (e.g., syntactic or semantic) the model does preserve. We specifically want to get samples from the embedding set to see what is the semantic similarity between different words. To fulfill this requirement, we design two different visualization strategies to explore the embedding space: (1) side-by-side visualization and (2) interactive visualization.

The side-by-side visualization helps users compare the qualities of the word similarity list between multiple embeddings (see figure 5). It allows researchers to "zoom-out" and see at the overview level what is the main difference between multiple embeddings. Moreover, it can visualize large embeddings up to the memory size of the running system. Regarding implementation, we implemented this visualization from scratch running on a lightweight webserver called Flask (`flask.pocoo.org`).

For the interactive visualization, it helps researchers "zoom-in" each embedding space to explore how each word is similar to the others. To do this, the well-known Embedding Projector (`projector.tensorflow.org`) is employed to explore the embedding space interactively. Unlike the side-by-side visualization, this interactive visualization can only visualize up to a certain amount of embedding vectors as long as the tensor graph is less than 2GB. This is a big limitation of the interactive visualization approach, which we plan to improve in the near future. Finally, it is worth to mention that the visualization module is dynamic and it does not require to change any codes when users want to visualize multiple pre-trained word embeddings.

To demonstrate the effectiveness of the ETNLP pipeline, we employ it to a use case in Vietnamese. Evaluating pre-trained embeddings in Vietnamese is a challenge as there is no publicly available *large*[1] lexical resource similar to the word analogy list in English to evaluate the performance of pre-trained embeddings. Moreover, different from English where all word analogy records consist of a single syllable in one record (e.g., grandfather | grandmother | king | queen), in Vietnamese, there are many cases where only words formulated by multiple syllables can represent a word analogy record (e.g., ông nội | bà ngoại | vua | nữ_hoàng).

We propose a large word analogy list in Vietnamese which can handle the problems. Having that word analogy list constructed, we utilize different embedding models, namely Word2Vec, fastText, ELMO and BERT on Vietnamese Wikipedia data to generate different sets of word embeddings. We then utilize the word analogy list to select suitable sets of embeddings for the named entity recognition (NER) task in Vietnamese. We achieve

[1] There are a couple of available datasets (Nguyen et al., 2018b). But the datasets are small containing only 400 words.

the new state-of-the-art results on VLSP 2016[2], a Vietnamese benchmark dataset for the NER task.

Here are our key contributions in this work:

- Propose a systematic pipeline (ETNLP) to evaluate, extract, and visualize multiple sets of word embeddings on a downstream task.

- Release a large word analogy list in Vietnamese for evaluating multiple word embeddings.

- Train and release multiple sets of word embeddings for NLP tasks in Vietnamese, wherein, their effectiveness is verified through new state-of-the-art results on a NER task in Vietnamese.

The rest of this paper is organized as follows. Section 2 describes how different embedding models are trained. Section 3 shows how to use ETNLP to extract, evaluate, and visualize word embeddings. Section 4 explains how the word embeddings are selected for the NER task using the word analogy task. Section 5 concludes the paper followed by future work.

2 Embedding Models

This section details the word embedding models incorporated in our systematic pipeline.

- **Word2Vec (W2V)** (Mikolov et al., 2013): a widely used method in NLP for generating word embeddings.

- **W2V_C2V**: the Word2Vec (W2V) model faces the OOV issue on unseen text, therefore, we provide a character2vec (C2V) (Kim et al., 2015) embedding for unseen words. When the C2V is not available, it can be easily calculated from a W2V model by averaging all vectors where a character occurred. Our experiments further confirm this averaging approach is efficient.

- **fastText** (Bojanowski et al., 2016): it associates embeddings with character-based n-grams, and a word is represented as the summation of the representations of its character-based n-grams. Based on this design, fastText attempts to capture morphological information to induce word embeddings, and hence, deals better with OOV words.

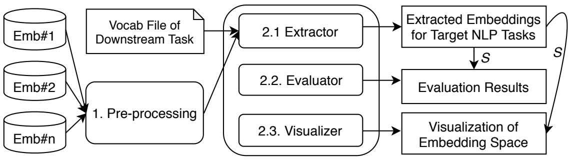

Emb#1
Emb#2
Emb#n
Vocab File of Downstream Task
1. Pre-processing
2.1 Extractor
2.2. Evaluator
2.3. Visualizer
Extracted Embeddings for Target NLP Tasks
S
S
Evaluation Results
Visualization of Embedding Space

Figure 1: General process of the ETNLP pipeline where $\mathcal{S}$ is the set of extracted embeddings for Evaluation and Visualization of multiple embeddings on a downstream NLP task.

- **ELMO** (Peters et al., 2018): a model generates embeddings for a word based on the context it appears. Thus, we choose the contexts where the word appears in the training corpus to generate embeddings for each of its occurrences. Then the final embedding vector is the average of all its context embeddings.

- **BERT_{Base, Large}** (Devlin et al., 2018): BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. Different from ELMO, the directional models, which reads the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words simultaneously. It, therefore, is considered bidirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word). BERT comes with two configurations called BERT_Base (12 layers) and BERT_Large (24 layers). To get the embedding vector of a word, we average all vectors of its subwords. Regarding contexts, similar to the ELMO model above, we choose the contexts where the word appears in the training corpus.

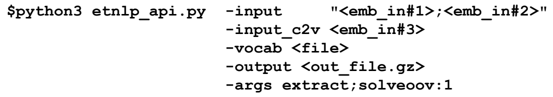
3 Systematic Pipeline

Figure 1 shows the general process of the ETNLP pipeline. The four main processes of ETNLP are very simple to call from either the command-line or the Python API.

- **Pre-processing:** since we use Word2Vec (W2V) format as the standard format for the whole process of ETNLP, we provide a pre-processing tool for converting different embedding formats to the W2V format.

- **Extractor:** to extract embedding vectors at word level for the specific target NLP task (i.e., NER task in our case). For instance, the popular implementation of Reimers and Gurevych (2017) on the sequence tagging task allows users to set location for the word embeddings. The format of the

[2] http://vlsp.org.vn/vlsp2016/eval/ner



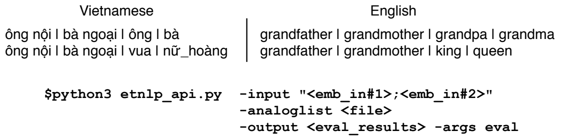
```
$python3 etnlp_api.py  -input      "<emb_in#1>;<emb_in#2>"
                       -input_c2v <emb_in#3>
                       -vocab <file>
                       -output <out_file.gz>
                       -args extract;solveoov:1
```

Figure 2: Run *extractor* to export single or multiple embeddings for NLP tasks.

file is text-based, i.e., each line contains the embedding of a word. The file then is compressed in .gz format. Figure 2 shows a command-line to extract multiple embeddings for an NLP task. The argument "-vocab" is the location to a vocabulary list of the target NLP task (i.e., the NER task) which is extracted from the task training data. The option "solveoov:1" informs the *extractor* to use Character2Vec (C2V) embedding to solve OOV words in the first embedding "<emb_in#1>". The "-input_c2v" can be omitted if users wish to simply extract embeddings from the embedding list given after the "-input_embs" argument. Output of this phase is a set of embeddings $\mathcal{S}$ to run on the next evaluation phase.

- **Evaluator** evaluates multiple sets of embeddings (i.e., $\mathcal{S}$) on the word analogy task. Based on the performance of each set of embeddings in $\mathcal{S}$, we can decide what embeddings are used in the target NLP task. To do this evaluation, users have to set the location of the word embeddings and the word analogy list. For more convenience to represent the compound words, we use " | " to separate different part of a word analogy record instead of space as in the English word analogy list. Figure 3 shows an example of two records in the word analogy in Vietnamese (on the left) and their translation (on the right). The lower part shows a command-line to evaluate multiple sets of word embeddings on this task. Regarding this *evaluator*, it is worth to note that with a huge number of possible linguistic relations (and different objectives, e.g., modeling syntactic vs. semantic properties), no embedding model is able to hold all related words close in the vector space. Therefore, only one testing schema (i.e., word analogy test) is not enough to evaluate multiple pre-trained embeddings. Thus, ETNLP is designed with the capability to be easily plugged in more tests, which makes *evaluator* more robust. However, in this paper, our experimental results showed that, word analogy task is sufficient to select good embeddings for the NER task in Vietnamese.

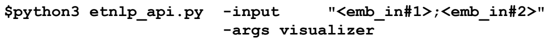
- **Visualizer:** to visualize given word embeddings in the argument "-input_embs" in both zoom-out (the side-by-side visualization) and zoom-in (the interactive visualization) manners. For the zoom-out, users type a word that they want to compare the similar words in different embedding models (see Figure 5). For the zoom-in, after the executions, embedding vectors are transformed to tensors to visualize with the Embedding Projector. Each word embedding will be set to different local port from which, users can explore the embedding space using a Web browser. Figure 6 shows an example of the interactive visualization of "Hà_Nội"$_{Hanoi}$ using ELMO embeddings. See Figure 4 for an example command-line.



```
              Vietnamese                            English
ông nội | bà ngoại | ông | bà           | grandfather | grandmother | grandpa | grandma
ông nội | bà ngoại | vua | nữ_hoàng     | grandfather | grandmother | king | queen

    $python3 etnlp_api.py  -input "<emb_in#1>;<emb_in#2>"
                           -analoglist <file>
                           -output <eval_results> -args eval
```

Figure 3: Run *evaluator* on multiple word embeddings on the word analogy task.



```
$python3 etnlp_api.py  -input      "<emb_in#1>;<emb_in#2>"
                       -args visualizer
```

Figure 4: Run *visualizer* to explore given pre-trained embedding models.

4 Evaluations: A Use-Case in Vietnamese

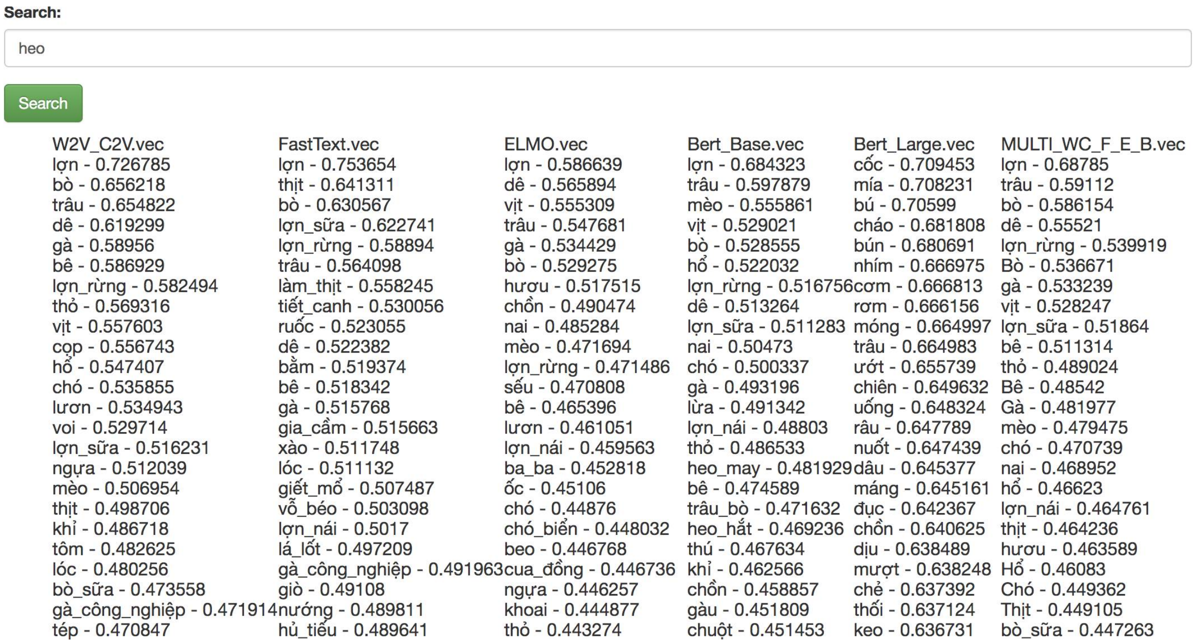
4.1 Training Word Embeddings

We trained embedding models detailed in Section 2 on the Wikipedia dump in Vietnamese[3]. We then apply sentence tokenization and word segmentation provided by VnCoreNLP (Vu et al., 2018a; Nguyen et al., 2018a) to pre-process all documents. It is noted that, for BERT model, we have to (1) format the data differently for the next sentence prediction task; and (2) use SentencePiece (Kudo and Richardson, 2018) to tokenize the data for learning the pre-trained embedding. It is worth

Table 1: Evaluation results of different word embeddings on the Word Analogy Task. P-value column shows significance test results using Paired *t*-tests. '*' means significant (p-value < 0.05) to the rest.

Model	MAP@10	P-value
W2V_C2V	0.4796	*
FastText	0.4970	See [1] & [2]
ELMO	**0.4999**	vs. FastText: 0.95 [1]
BERT_Base	0.4609	*
BERT_Large	0.4634	-
MULTI	0.4906	vs. FastText: 0.025 [2]

[3] https://goo.gl/8WNfyZ



Search:

heo

W2V_C2V.vec	FastText.vec	ELMO.vec	Bert_Base.vec	Bert_Large.vec	MULTI_WC_F_E_B.vec
lợn - 0.726785	lợn - 0.753654	lợn - 0.586639	lợn - 0.684323	cốc - 0.709453	lợn - 0.68785
bò - 0.656218	thịt - 0.641311	dê - 0.565894	trâu - 0.597879	mía - 0.708231	trâu - 0.59112
trâu - 0.654822	bò - 0.630567	vịt - 0.555309	mèo - 0.555861	bú - 0.70599	bò - 0.586154
dê - 0.619299	lợn_sữa - 0.622741	trâu - 0.547681	vịt - 0.529021	cháo - 0.681808	dê - 0.55521
gà - 0.58956	lợn_rừng - 0.58894	gà - 0.534429	bò - 0.528555	bún - 0.680691	lợn_rừng - 0.539919
bê - 0.586929	trâu - 0.564098	bò - 0.529275	hổ - 0.522032	nhím - 0.666975	Bò - 0.536671
lợn_rừng - 0.582494	làm_thịt - 0.558245	hươu - 0.517515	lợn_rừng - 0.516756	cơm - 0.666813	gà - 0.533239
thỏ - 0.569316	tiết_canh - 0.530056	chồn - 0.490474	dê - 0.513264	rơm - 0.666156	vịt - 0.528247
vịt - 0.557603	ruốc - 0.523055	nai - 0.485284	lợn_sữa - 0.511283	móng - 0.664997	lợn_sữa - 0.51864
cọp - 0.556743	dê - 0.522382	mèo - 0.471694	nai - 0.50473	trâu - 0.664983	bê - 0.511314
hổ - 0.547407	bằm - 0.519374	lợn_rừng - 0.471486	chó - 0.500337	ướt - 0.655739	thỏ - 0.489024
chó - 0.535855	bê - 0.518342	sếu - 0.470808	gà - 0.493196	chiên - 0.649632	Bê - 0.48542
lươn - 0.534943	gà - 0.515768	bê - 0.465396	lừa - 0.491342	uống - 0.648324	Gà - 0.481977
voi - 0.529714	gia_cầm - 0.515663	lươn - 0.461051	lợn_nái - 0.48803	râu - 0.647789	mèo - 0.479475
lợn_sữa - 0.516231	xào - 0.511748	lợn_nái - 0.459563	thỏ - 0.486533	nuốt - 0.647439	chó - 0.470739
ngựa - 0.512039	lóc - 0.511132	ba_ba - 0.452818	heo_may - 0.481929	dâu - 0.645377	nai - 0.468952
mèo - 0.506954	giết_mổ - 0.507487	ốc - 0.45106	bê - 0.474589	máng - 0.645161	hổ - 0.46623
thịt - 0.498706	vỗ_béo - 0.503098	chó - 0.44876	trâu_bò - 0.471632	đục - 0.642367	lợn_nái - 0.464761
khỉ - 0.486718	lợn_nái - 0.5017	chó_biển - 0.448032	heo_hắt - 0.469236	chồn - 0.640625	thịt - 0.464236
tôm - 0.482625	lá_lốt - 0.497209	beo - 0.446768	thú - 0.467634	dịu - 0.638489	hươu - 0.463589
lóc - 0.480256	gà_công_nghiệp - 0.491963	cua_đồng - 0.446736	khỉ - 0.462566	mượt - 0.638248	Hổ - 0.46083
bò_sữa - 0.473558	giò - 0.49108	ngựa - 0.446257	chồn - 0.458857	chẻ - 0.637392	Chó - 0.449362
gà_công_nghiệp - 0.471914	nướng - 0.489811	khoai - 0.444877	gàu - 0.451809	thối - 0.637124	Thịt - 0.449105
tép - 0.470847	hủ_tiếu - 0.489641	thỏ - 0.443274	chuột - 0.451453	keo - 0.636731	bò_sữa - 0.447263

Figure 5: Side-by-side visualization for the word “heo $_{\textbf{pig}}$” with multiple embeddings. From this visualization, we get the sense that W2V_C2V, ELMO, and Bert_Base mainly capture the categorical information (i.e., “heo $_{\textbf{pig}}$” is surrounded by names of other animals, e.g., "bò $_{\textbf{cow}}$", "trâu $_{\textbf{buffalo}}$") while “FastText“ captures both categorical information (i.e., surrounded by names of other animals) and related verbs to “pig” such as “xào $_{\textbf{frying}}$”, “nướng $_{\textbf{grill}}$”. Bert_Large, on the other hand, does not converge well due to the short training steps mentioned in section 4, therefore, many irrelevant words (e.g., “cốc $_{\textbf{cup}}$”, “dịu $_{\textbf{floppy}}$”) are surrounded the input word “heo $_{\textbf{pig}}$”, “keo$_{\textbf{glue}}$”.

Table 2: Example of five types of semantic and four (out of nine) types of syntactic questions in the word analogy list. “NOT AVAILABLE“ means that the syntactic phenomena do not apply in Vietnamese in comparison to the list of Mikolov et al. (2013).

	Type of relationship	Word Pair 1	Word Pair 2
Semantic	capital-common-countries	Athens | Hy_Lạp $_{\text{Greek}}$	| Baghdad | Irac
	capital-world	Abuja | Nigeria	| Thổ Nhĩ Kỳ $_{\text{Turkey}}$ | Turkey
	currency	Algeria | dinar	| Canada | đô la $_{\text{dollar}}$
	city-in-zone	Hòa Bình $_{\text{Hoa Binh}}$ | Tây Bắc Bộ $_{\text{West North}}$	| Hà Giang $_{\text{Ha Giang}}$ | Đông Bắc Bộ $_{\text{East Northern}}$
	family	cậu bé $_{\text{boy}}$ | cô gái $_{\text{girl}}$	| anh trai $_{\text{brother}}$ | em gái $_{\text{sister}}$
Syntactic	gram1-adjective-to-adverb	NOT AVAILABLE	
	gram2-opposite	chấp nhận được $_{\text{acceptable}}$ | không thể chấp nhận $_{\text{unacceptable}}$	| nhận thức $_{\text{aware}}$ | không biết $_{\text{unaware}}$
	gram3-comparative	tệ $_{\text{bad}}$ | tệ hơn $_{\text{worse}}$	| lớn $_{\text{big}}$ | lớn hơn $_{\text{bigger}}$
	gram4-superlative	lớn $_{\text{big}}$ | lớn nhất $_{\text{biggest}}$	| sáng $_{\text{bright}}$ | sáng nhất $_{\text{brightest}}$
	gram5-present-participle	NOT AVAILABLE	
	gram6-nationality-adjective	Albania | Tiếng Albania $_{\text{Albanian}}$	| Argentina | Tiếng Argentina $_{\text{Argentinean}}$
	gram7-past-tense	NOT AVAILABLE	
	gram8-plural-nouns	NOT AVAILABLE	
	gram9-plural-verbs	NOT AVAILABLE	

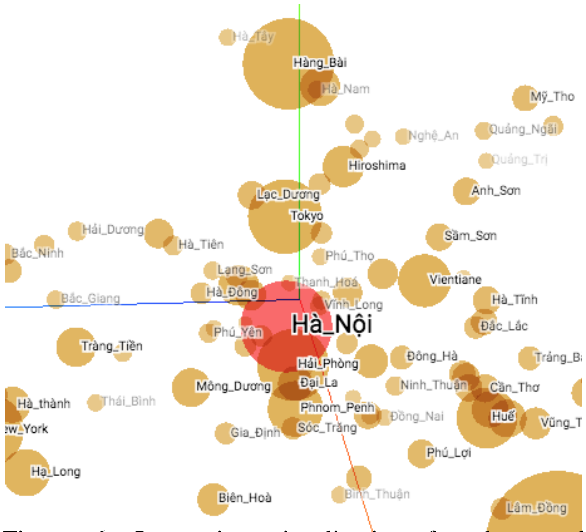


Figure 6: Interactive visualization for the word "Hà_Nội" with ELMO embeddings where near "Hà_Nội" are the names of many other cities in Vietnam (e.g., "Hải_Phòng$_{\textbf{Hai Phong}}$" as well as capital of other countries (e.g., Tokyo).

Table 3: Grid search for hyper-parameters.

Hyper-parameter	Search Space		
cemb dim (char embedding)	50	100	500
drpt (dropout rate)	0.3	0.5	0.7
lstm-s (LSTM size)	50	100	500
lrate (learning rate)	0.0005	0.001	0.005

noting that due to the limitation in computing resources, we can only run BERT_Base for 900,000 update steps and BERT_Large for 60,000 update steps. We, therefore, do not report the result of BERT_Large for a fair comparison. We also create *MULTI* embeddings by concatenating four sets of embeddings (i.e., W2V_C2V, fastText, ELMO and BERT_Base) [4].

4.2 Dataset

The named entity recognition (NER) shared task at the 2016 VLSP workshop provides a dataset of 16,861 manually annotated sentences for training and development, and a set of 2,831 manually annotated sentences for test, with four NER labels PER, LOC, ORG, and MISC. The data was published in 2016 and recently reported in Nguyen et al. (2019). It is a standard benchmark on the NER task and has been used in (Vu et al., 2018a; Dong and Nguyen, 2018). It is noted that, in the original dataset, each word representing a full personal name are separated into syllables that constitute the word. Because this annotation scheme results in an unrealistic scenario for a pipeline evaluation (Vu et al., 2018a), therefore, we tested on a "modified" VLSP 2016 corpus where we merge contiguous syllables constituting a full name to form a word. This similar setup was also used in (Vu et al., 2018a; Dong and Nguyen, 2018), the current state-of-the-art approaches.

4.3 Word Analogy Task

To measure the quality of different sets of embeddings in Vietnamese, similar to Mikolov et al. (2013), we define a word analogy list consisting of 9,802 word analogy records. To create the list, we selected suitable categories from the English word analogy list and then translated them to Vietnamese. We also added customized categories which are suitable for Vietnamese (e.g., cities and their zones in Vietnam). Different from (Mikolov et al., 2013), five categories: "Adjective to adverb", "Present Participle", "Past tense", "Plural nouns", "Plural verbs" were not used to be translated in Vietnamese since the same syntactic phenomena does not exist in Vietnamese. Table 2 shows the list of categories and their examples of the constructed word analogy list in Vietnamese. Since most of this process is automatically done, it can be applied easily to other languages. To know which set of word embeddings potentially works better for a target downstream task, we limit the vocabulary of the embeddings similar to vocabulary of the task (i.e., the NER task). Thus, only 3,135 word analogy records are being evaluated for the NER dataset (Section 4.2).

Regarding the evaluation metric, Mikolov et al. (2013) used accuracy metric to measure the quality of word embeddings on the task in which only when the expected word is on top of the prediction list, then the model gets +1 for true positive count. However, this is not a well-suited metric in low resource languages where training corpus is relatively small, i.e., 233M tokens in Vietnamese Wiki compared to 6B tokens in Google News corpus. Therefore, we change to use mean average precision (MAP) metric to measure quality of the word analogy task. MAP is widely used in information retrieval to evaluate results based on the top$K$ returned results (Vu et al., 2019). We use MAP@10 in this paper. Table 1 shows evaluation results of different sets of embeddings on the word analogy task. The *evaluator* of ETNLP also shows P-value using the paired t-tests on the raw

[4] We do not use W2V here because W2V_C2V is W2V with the use of character embedding to deal with OOV.

Table 4: Performance of the NER task using different embedding models. The $MULTI_{WC\_F\_E\_B}$ is the concatenation of four embeddings: W2V_C2V, fastText, ELMO, and Bert_Base. “wemb dim” is the dimension of the embedding model. VnCoreNLP* means we retrain the VnCoreNLP with our pre-trained embeddings.

	F1	wemb dim	cemb dim	drpt	lstm-s	lrate
BiLC3 (Ma and Hovy, 2016)	88.28	300	-	-	-	-
VNER (Dong and Nguyen, 2018)	89.58	300	300	0.6	-	0.001
VnCoreNLP (Vu et al., 2018a)	88.55	300	-	-	-	-
VnCoreNLP (*)	**91.30**	1024	-	-	-	-
BiLC3 + W2V	*89.01*	300	50	0.5	100	0.0005
BiLC3 + BERT-Base	*88.26*	768	500	0.3	100	0.0005
BiLC3 + W2V_C2V	*89.46*	300	100	0.5	500	0.0005
BiLC3 + fastText	89.65	300	500	0.3	100	0.001
BiLC3 + ELMO	89.67	1024	100	0.7	500	0.0005
BiLC3 + $MULTI_{WC\_F\_E\_B}$	**91.09**	2392	100	0.7	100	0.001

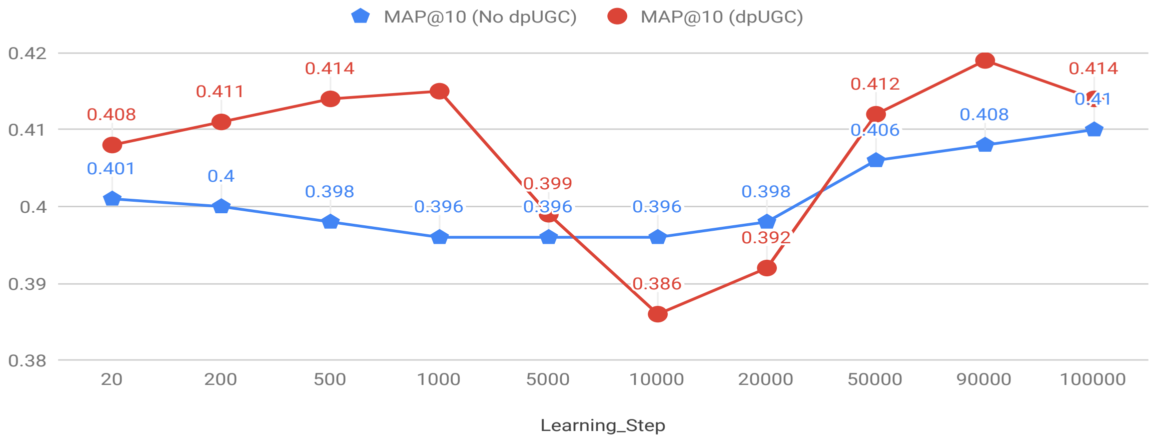


Figure 7: Evaluation results of different word embeddings trained using **dpUGC** and **No dpUGC** (i.e., one option in dpUGC to train embeddings without privacy guarantee for comparison) on the Word Analogy Task.

MAP@10 scores (i.e., before averaging) between different sets of embeddings. The P-values (Table 1) show that the performances of the top three sets of word embeddings (i.e., fastText, ELMO, and MULTI), are significantly better than the remainders but there is no significant difference between the three. Therefore, these sets of embeddings will be selected for NER task.

4.4 NER Task

Model: We apply the current most well-known neural network architecture for NER task of Ma and Hovy (2016) with no modification in its architecture, namely, *BiLSTM-CRF+CNN-char (BiLC3)*. Only in the embedding layer, a different set of word embeddings is used to evaluate their effectiveness. Regarding experiments, we perform a grid search for hyper-parameters and select the best parameters on the validation set to run on the test set. Table 3 presents the value ranges we used to search for the best hyper-parameters. We also follow the same setting as in (Vu et al., 2018a) to use the *last* 2000 records in the training data as the validation set. Moreover, due to the availability of the VnCoreNLP code, we also retrain their model with our pre-trained embeddings (*VnCoreNLP**).

Main Results: Table 4 shows the results of NER task using different word embeddings. It clearly shows that, by using the pre-trained embeddings on Vietnamese Wikipedia data, we can achieve the new state-of-the-art results on the task. The reason might be that FastText, ELMO and MULTI can handle OOV words as well as capture better the context of the words. Moreover, learning the embeddings from a formal dataset like Wikipedia is beneficial for the NER task. This also verified the fact that using our pre-trained embeddings on VnCoreNLP helps significantly boost its performance. Table 4 also shows the F1 scores of W2V, W2V_C2V and BERT_Base embeddings which are worse than three *selected* embeddings

Table 5: P-values of the paired t-tests between embeddings obtained using dpUGC at different learning step (Emb@L). "-" denotes values of these entries in the upper triangular matrix are the values of the transposed entries in the lower triangular matrix. P-values in **bold font** are statistical significance at the level of 0.05.

Emb@L	20	200	500	1000	5000	10000	20000	50000	90K	100K
20	1	-	-	-	-	-	-	-	-	-
200	0.0578	1	-	-	-	-	-	-	-	-
500	**0.0074**	0.1809	1	-	-	-	-	-	-	-
1000	**0.0053**	0.169	0.9031	1	-	-	-	-	-	-
5000	**0.0178**	**0.0009**	6.992	1.6242	1	-	-	-	-	-
10000	2.543	6.9872	2.25867	9.3987	**0.001**	1	-	-	-	-
20000	**0.0016**	**0.0001**	1.757	9.6053	0.112	0.1819	1	-	-	-
50000	0.5077	0.9023	0.73137	0.7003	**0.031**	5.0673	**0.0001**	1	-	-
90K	0.1205	0.2878	0.5127	0.5323	**0.0049**	2.4211	**0.0001**	0.2688	1	-
100K	0.3777	0.6822	0.9932	0.9764	**0.0357**	8.2638	**0.0019**	0.7274	0.2758	1

(i.e., fastText, ELMO and MULTI). This might indicate that using word analogy to select embeddings for downstream NLP tasks is sensible.

4.5 Privacy-Guaranteed Embedding Selection Task

In this section, we show how to apply ETNLP to another downstream task of privacy-guaranteed embedding selection. Vu et al. (2019) introduced dpUGC to guarantee privacy for word embeddings. The main intuition behind dpUGC is that, when the embedding is trained on very sensitive text corpus (e.g., medical text data), it has to guarantee privacy at the highest level to prevent privacy leakage. However, among many embeddings at different learning steps of dpUGC, how to choose a suitable embedding to achieve a good trade-off between data privacy and data utility is a key challenge. To this end, we propose to apply ETNLP into this scenario to select good embeddings for knowledge sharing using dpUGC.

Similar to Vu et al. (2019), we trained 20 different embeddings from 10 different learning steps while training on the same Vietnamese Wikipedia dataset as used in Section 4.1 with (dpUGC) and without privacy-guarantee (No dpUGC) to evaluate their performances. Figure 7 shows that the pre-trained embedding at learning_step 1000 (Emb@1000) seems to be a good word embedding candidate to have a good trade-off between privacy guarantee and data utility. Emb@1000 was in favor because of two reasons. Firstly, in training privacy-guaranteed embeddings, we try to stop as early as possible since the more training steps we run, the higher privacy we have to sacrifice (Vu et al., 2019). Secondly, its performance in the Word Analogy Task was more or less similar to the other good embedding at the learning step 90K (i.e., Emb@90K). In fact, from Table 5 we know that the performance between Emb@1000 and Emb@90K learning steps are not significant difference. Therefore, selecting the pre-trained embedding at the learning step 1000 is the best option for privacy-guaranteed embedding using dpUGC. In summary, in this task, we showed how ETNLP can be used to select a good word embedding candidate for privacy-guaranteed knowledge sharing. Normally, this selection process is very time consuming, however, it is much easier with ETNLP since it allows users to import multiple embeddings for running evaluations.

5 Conclusions

We have presented a new systematic pipeline, ETNLP, for extracting, evaluating and visualizing multiple pre-trained embeddings on a specific downstream task. The ETNLP pipeline was designed with three principles in mind: (1) easy to apply on any language processing task, (2) better performance, and (3) be able to handle unknown vocabulary in real-world data (i.e., using C2V (char to vec)). The evaluation of the approach in (1) Vietnamese NER task and (2) privacy-guaranteed embedding selection task showed its effectiveness.

In the future, we plan to support more embeddings in different languages, especially in low resource languages. We will also support new ways to explore the embedding spaces including at phrase and subword levels.

Acknowledgement

This work is supported by the Federated Database project funded by Umeå University, Sweden. The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at HPC2N.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ngan Dong and Kim Anh Nguyen. 2018. Attentive neural network for named entity recognition in vietnamese. *CoRR*, abs/1810.13097.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2177–2185, Cambridge, MA, USA. MIT Press.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving Topic Models with Latent Feature Word Representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Dat Quoc Nguyen, Dai Quoc Nguyen, Thanh Vu, Mark Dras, and Mark Johnson. 2018a. A Fast and Accurate Vietnamese Word Segmenter. In *Proceedings of the 2018 LREC*, Miyazaki, Japan.

Huyen Nguyen, Quyen Ngo, Luong Vu, Vu Tran, and Hien Nguyen. 2019. Vlsp shared task: Named entity recognition. *Journal of Computer Science and Cybernetics*, 34(4):283–294.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2018b. Introducing two vietnamese datasets for evaluating semantic models of (dis-)similarity and relatedness. In *Proceedings of the 2018 NAACL: Short Papers*, pages 199–205.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Matt Taddy. 2015. Document classification by inversion of distributed language representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 45–49. Association for Computational Linguistics.

Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2018a. Vncorenlp: A vietnamese natural language processing toolkit. In *Proceedings of the 2018 NAACL: Demonstrations*, pages 56–60, New Orleans, Louisiana. Association for Computational Linguistics.

Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. 2018b. Nihrio at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. In *Proceedings of SemEval 2018*.

Xuan-Son Vu, Hyun-Je Song, and Seong-Bae Park. 2014. Building a vietnamese sentiwordnet using vietnamese electronic dictionary and string kernel. In *Knowledge Management and Acquisition for Smart Systems and Services*, pages 223–235, Cham. Springer International Publishing.

Xuan-Son Vu, Son N. Tran, and Lili Jiang. 2019. dpugc: Learn differentially private representation for user generated contents. In *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing*.

Tagger for Polish Computer Mediated Communication Texts

Wiktor Walentynowicz
Wrocław University
of Science and Technology

Maciej Piasecki
Wrocław University
of Science and Technology

Marcin Oleksy
Wrocław University
of Science and Technology

{wiktor.walentynowicz,maciej.piasecki,marcin.oleksy}@pwr.edu.pl

Abstract

In this paper we present a morpho-syntactic tagger dedicated to Computer-mediated Communication texts in Polish. Its construction is based on an expanded RNN-based neural network adapted to the work on noisy texts. Among several techniques, the tagger utilises fastText embedding vectors, sequential character embedding vectors, and Brown clustering for the coarse-grained representation of sentence structures. In addition a set of manually written rules was proposed for post-processing. The system was trained to disambiguate descriptions of words in relation to Parts of Speech tags together with the full morphological information in terms of values for the different grammatical categories. We present also evaluation of several model variants on the gold standard annotated CMC data, comparison to the state-of-the-art taggers for Polish and error analysis. The proposed tagger shows significantly better results in this domain and demonstrates the viability of adaptation.

1 Introduction

Morpho-syntactic disambiguation (called also morpho-syntactic tagging) is an important pre-processing step in many text processing pipelines (e.g. terminology and information extraction), especially in the case of highly inflected languages where tagging is tightly correlated with lemmatisation. Work on the development of programs for morpho-syntactic disambiguaters, henceforth *taggers*, has been concentrated on texts written in the standard language, i.e. containing only small percentage free of jargon, extra-linguistic elements like codes or symbols, etc. Computer-mediated communication (CMC) texts, especially texts from different kinds of social media, are written in language that often is significantly different from the standard language, see Sec. 3. While in the case of texts in a standard language, a state-of-the-art tagger reaches near 95% of accuracy in disambiguation, taggers for CMC texts express much lower accuracy. Moreover, much fewer works are devoted to tagging CMC texts than text of the standard language, while the problem of processing CMC texts is continuously growing in importance and numbers of applications. Thus, the fast growth of social media requires appropriate adaptation, or even expansion, of the tagging methods, to serve growing interest and requirements in the processing of texts of such kinds. According to our best knowledge, a morpho-syntactic tagger dedicated to CMC texts in the Polish language has not been yet developed or at least made publicly available. The goal of this work is a far going adaptation of a tagger for the standard Polish to the demands of CMC texts in Polish.

In the rest of the paper, first solutions for tagging the standard Polish language and CMC texts in general are discussed. Next, we analyse characteristic features of CMC texts on the basis of a collected CMC corpus. A tagger model is proposed. Finally, we presented evaluation of the tagger and discuss perspectives for its further development.

2 Related Works

The construction of our CMC tagger has been inspired by a tagger for the Polish language that won the PolEval competition (Kobyliński and Ogrodniczuk, 2017) in 2017 that is called *Toygger* (Krasnowska-Kieraś, 2017). *Toygger* is based on a recursive neural network – a bidirectional LSTM. Recursive layers extract features based on an input

Proceedings of Recent Advances in Natural Language Processing, pages 1295–1303,
Varna, Bulgaria, Sep 2–4 2019.

vector encoding morphological information and including embedded vectors. The extracted features are next transferred to separated non-linear layers, where each represents a different part of the morpho-syntactic tag. In Sec. 4 we propose several expansions to this model which are aimed at providing better handling of noisy texts. Aside from Toygger, another tagger – *KRNNT* (Wróbel, 2017), also based on a bidirectional LSTM network, achieved very good results in the PolEval contest. The main difference between these two is in the encoding of the input text and features generated for it. Toygger uses text encoded as a sequence of embedding vectors in combination with information from the morphological analysis performed with the morpho-syntactic analyser *Morfeusz2* (Kieraś and Woliński, 2017), and KRNNT uses information from the morphological analysis from Maca (Radziszewski and Śniatowski, 2011) (that is also employing *Morfeusz* inside) in combination with predetermined features based on a set of features from the *Concraft* (Waszczuk, 2012) tool.

In a similar way, complex systems such as dependency parser *COMBO* (Rybak and Wróblewska, 2018), in their internal taggers began to use LSTM networks with similar architecture to the stand-alone taggers mentioned above. Inside the COMBO system the tagger component obtains the features extracted from a bidirectional LSTM layer and passes them through a fully connected network with one hidden layer with a softmax activation function. That network predicts a universal part-of-speech tag and a tagset specific tag (for instance a grammatical class in the case of a tagset for Polish). The morphological features have similar networks for each feature type.

Apart from the systems created for the Polish language, several solutions were proposed for other Slavic languages or even highly inflected languages in general. The winning solution of the competition organised at the VarDial 2018 conference (Zampieri et al., 2018) was a system based on a bidirectional LSTM network, which instead of classifying words with morpho-syntactic tags generates the tags in a character per character way and uses a different type of a recursive network (Silfverberg and Drobac, 2018) for this purpose. The way the network is trained ensures that a tag that has never occurred in the training data is not generated. Such a manner of getting replies from networks allows also words, which are concatenated together from several morphemes, to have a 'multi-part' tag, i.e. a cluster of several tags *per se*. This solution was inspired by the Sequence-to-Sequence architecture. It also shows the positive effect of combining embedding vectors for words and characters, which is confirmed by other works from this genre, such as (Plank et al., 2016). Another work in this field is (Ljubešić, 2018), which compares a tagger model based on Conditional Random Fields with a model based on a recursive neural network in disambiguation of Slovene, Serbian and Croatian CMC texts. The differences between the two models are small (about 0.02), which leads to the conclusion that both methods are worth considering in further research. However, a comparative study in (Östling, 2018) shows that better results are achieved with good manual processing of features than with extending and deepening the architecture.

Apart from CRF-based models, the other methods presented here do not pay attention to both sides of the context. Using a bidirectional LSTM layers does bring information about the context, but only before the word (or after the word when looking from the other direction) and the method of combining the results of these two directions does not guarantee focusing on both sides of context in the same degree. Therefore, our solution proposes to add to the network information about the context from the Brown clustering algorithm. Here, we were inspired by another work about tagger adaptation (Ljubešić et al., 2017).

3 Computer-Mediated Communication Corpus

Computer-mediated communication (CMC) texts are part of user-generated content (UGC) data. Due to the nature of this kind of data, the texts commonly include many mistakes and problematic phenomena. A linguistic analysis of the data as well as the results of similar research (see e.g (Pluwak et al., 2016)) helped us to define distinctive features, which are related to various text levels such as: notation (e.g. lack of diacritics, spelling mistakes, typos, omissions of capital letters, incorrectly connected or disconnected segments, lack of or poor punctuation), morphology and syntax (e.g. incorrect word endings, token repetition, lack of phrase elements), or lexical issues (e.g. emojis and special characters, internet slang, characters replacements, abbreviated forms, URL addresses, hashtags, mentions). Most of these phenomena require specific solutions in annotation guidelines. Since the available data in Polish is based on the sources of a different nature – they are mostly edited and officially published texts (Przepiórkowski et al., 2012) – there was a need to create a CMC corpus manually annotated with morphological information.

The *Corpus of the Colloquial Polish Language*[1] (CCPL) used in all experiments presented in this paper consists of 7,561 documents (402,810 tokens). All the source texts were posted by users on online social media platforms, so they have the characteristics of user-generated content (UGC). The texts include opinions, tweets, comments, social media posts and chat utterances.

The whole corpus was manually annotated with morphological information and next morphosyntactically disambiguated by the team of professional linguists from the Wrocław University of Science and Technology. The inter-annotator agreement for various pairs of annotators was calculated. The results ranged from 0.91 to 0.97.

4 Tagger Model

4.1 Data and Preprocessing

National Corpus of Polish (NCP) as the basic training data, and only in some experiments we expanded the training data with annotated texts from social media coming from CCPL, see Section 5. This subcorpus of NCP contains a bit above 1.2 million of manually annotated and disambiguated tokens from different sources. It is commonly used as training-testing data set for a tagging task in Polish language. For the test data set we used CCPL, which is described in Section 5.1.

We have decided not to apply text normalisation before tagging process in order to save as much information as possible from the text structure which can be helpful in next stages of processing like sentiment recognition. Many of the methods proposed in literature are based on a process: normalisation followed by tagger application. Thus, a comparison of our solution with them is difficult, as the text tagged is different. Therefore we do not handle segmentation problems in our tagging system. To deal with typos and lack of diacritics we focus on character representation together with suffix representation and choose to use the *fastText* embedding (Bojanowski et al., 2017), because it is based on the n-gram based subword word representation. In addition, we obtained cluster information to get better representation of contexts for similar words. We applied the Brown Clustering algorithm (Brown et al., 1992) to group words on the basis of the one million subcorpus of NCP (Przepiórkowski et al., 2012) (this is the same data set which is used as training data). The Brown Clustering is a method of hierarchical clustering of words based on their contexts. We assumed that words belonging to the same clusters have the same probability of the contextual occurrence under the condition of the occurrence of preceding and following words. In principle, we want to achieve good results in spite of processing noisy textual data, due to the knowledge of their structure on the coarse-grained description level based on Brown clusters of words. We assume that such representation helps to determine to which group of words an unknown or broken word may belong to. Other elements of CMC texts are emoticons, URL and e-mail addresses, hashtags and user mentions. We handle these cases with handwritten rules.

In several experiments, see Section 5, the evaluation of the whole tagging process was performed on the entire CCPL corpus, because it was not involved in training in those cases.

4.2 Input Text Representation

In order to improve the tagger ability to generalise over the training data, we used a representation based on distributional vector models. An input vector for a word from the processed sentence is constructed as a concatenation of several subvectors representing different properties of a word:

1. a morphological information vector,
2. a suffix character embedding vector,
3. a suffix index in the set of known suffixes,
4. a suffix embedding vector,
5. a word embedding vector from a *fastText*-based model,
6. a whole word character embedding,
7. a Brown cluster embedding vector.

The morphological information vector expresses jointly information collected from all tags

[1] http://hdl.handle.net/11321/637

that are possible for a given token according to the morphological analysis (in the case of unknown words, the full vector is set). Sets of possible tags are represented as a sequence of bits: every single bit of each vector represents a possible grammatical class or a value of some grammatical category (an attribute), e.g. case, number gender etc. For instance for the Polish word *jedzenie* 'food' we obtain a vector with two bits set for the two grammatical classes: *noun* (*jedzenie* as 'food') and *gerund* (*jedzenie* 'eating'), bits for the *nominal* and *accusative* genders, one bit for the *singular* number, one for the *m3* gender etc. The morphological vector is intended to be a kind of regularisation constraining the tagging process and to make the tagger decisions compatible with the morphological analyser.

The suffix character embedding vector – the part 2 – is trained during the time of learning by using an additional small (64 hidden units) biLSTM network to represent suffixes as character sequences. In all experiments, we set the size of a suffix to 3, based on the results published in the (Krasnowska-Kieraś, 2017) (who tested experimentally 4 and 5, too) and our previous experiments. This vector is aimed at recognition of different suffixes (carrying important morphological information) and their similarity.

The suffix index (3) and suffix embedding vector (4) represent also suffixes, but on the level of suffixes as tokens, not sequences of characters. Concerning the former, a set of known suffixes is first extracted from the learning data. Next each recognised suffix is represented by its index during training and testing. In the case of the latter, for suffixes as tokens their vector embeddings are trained during learning the sequential tagging task. During testing and application for each known suffix its embedding vector representation is searched for in the look-up table layer. By introducing these two components we want to emphasise the presences of more frequent suffixes that often express more specific morpho-syntactic information.

The fifth part is a word embedding vector for the whole token obtained from *fastText* model. We use a *fastText* model for Polish that was trained on a very large corpus of Polish (Kocoń, 2018). This vector introduced lexical element to the representation, but due to the nature of the distributional model, words of similar distribution receive similar vectors. Moreover, *fastText* (a subword distributional model) assigns also vectors to unknown words on the basis of n-gram structure.

The sixth part is a character embedding for whole words. Similar to suffix character embedding in part 2, it is learned during the training the whole tagger and has similar architecture. The most significant difference is that it takes whole words at the input, not just suffixes.

The last component (7) is an embedding vector for the Brown cluster of the given word. In a similar way to the suffix embedding vector, vector embeddings for Brown clusters are trained during learning the tagger. If the word does not appear in any cluster, it receives a cluster index for Out of Vocabulary (OOV) words. During testing and application the vectors are read from the look-up table layer. Brown clusters offer a coarse-grained representation of the input sequence that helps to analyse OOV words.

4.3 Network and Processing

The core part of the tagger is a deep neural network. The input vector is a sequence of the combined word vectors. It is sent to two bidirectional LSTM layers with 512 hidden units each. 50% dropout is applied to both LSTM layers. The goal of these layers is to calculate features for each word of the input sentence. Next, these features are used to feed down separated layers. These separated layers are *softmax layers*. The first one is dedicated to the grammatical class and the rest (13) to the 13 different grammatical categories (morphological attributes). The whole network is trained with the help of the *RMSprop optimizer* (Tieleman and Hinton, 2012) and the *Categorical Cross Entropy loss function*.

Due to the variability of CMC texts the main processing by the neural network is supplemented by deterministic post-processing which is performed in three steps:

1. verification of the correctness of a predicted tag,
2. final tag selection,
3. rule-based error detection and correction.

Concerning the first, the correctness of the predicted tag is checked in relation to the set of all possible tags proposed by the morphological analysis. Checking correctness of predicted tag is simple task. On the basis of the predicted grammatical class we verify if the attributes obtain values

number	accuracy
No clusters	85.21%
500	85.46%
750	85.44%
1000	85.44%
1250	85.31%
1500	85.23%
1750	**85.48**%
2000	85.36%

Table 1: Influence of the number of clusters on the tagger strict accuracy.

specified for this grammatical class. In the case of the lack of a tag exactly matching the predicted one among the tags obtained from the morphological analysis, for the final tag selection, we choose a tag that is in the minimal Levenshtein distance of its form to the form of the predicted tag, i.e. the predicted tags are somehow mapped onto tags available from the morphological analysis (in the case of out-of-vocabulary words the full set of tags is assumed as the result of the morphological analysis).

In order to improve an automatic tagging process, we developed and applied several rules. They specify morphological interpretations for selected words directly encountered in text. This concerns, in particular, emojis or internet addresses. In some cases a rule covers only the first characters which serve to identify the relevant word form, e.g. it was not possible to list all URL addresses but the rule using the expression [*if 'https://' in w*] could be applied to all word forms that begin with *https://*).

A morphological interpretation for several word forms (or types of word forms) were assigned irrespective of the interpretation automatically predicted by the tagger. An example of such a general rule is given below:

$if\,'https://'\,in\,w\,or\,'http://'\,in\,w:$

$corrected\_tag =' subst:sg:nom:m3'$

The results of the first attempt to error analysis were the basis for drawing up also specific rules for the word forms tagged initially with specific morphological information, e.g.

$if\ (w\ ==\ 'jak'\ or\ w\ ==\ 'tak')$

$and\ predicted\_tag\ ==\ 'adv:pos':$

$corrected\_tag\ =\ 'adv'$

settings	accuracy
SCE(128) + SE(64)	86.09%
SCE(128) + SE(64) + CLE(64)	*87.06%*
SCE(128) + SE(64) + CLE(64) + R	**87.87**%
CE(128) + SE(64) + CLE(64) + R	87.39%
CE(128) + SCE(128) + CLE(64) + R	87.67%

Table 2: CMC tagger strict accuracy in relation to the different ways of composing the input vector.

5 Evaluation

5.1 Experiments

In our research, we focused on two main aspects

- determining the best number of Brown clusters,
- and selecting the best possible configuration of the input vector components.

In all experiments the input vector included the components representing the morphological analyses and the *fastText* based representation of words. The *fastText* component vector size was fixed to 300 elements.

First, we performed tests to find the best number of the Brown clusters. The results of these experiments are shown in Table 1. The performance of the tagger is measured in the *strict accuracy*, i.e. only the assigned tags that completely, in relation to all its components, match the tag from the manual annotation are treated as correct solutions. The results show that the addition of clustering, regardless of its size, improves the tagging accuracy. Finally, we set the number of clusters to 1,750 in accordance with the best result obtained during the first experiments.

Next, we performed several experiments that were aimed at investigating the influence of the different joint vector components on the tagger accuracy. The results of the most important ones are presented in Table 2 where the shortcuts mean:

SCE – suffix characters embedding,

SE – suffix embedding,

CE – character embedding,

CLE – clustering embedding,

R – postprocessing rules.

The numbers in the round brackets remind about the size of the given vector component. In Table 2 the performance of the tagger is measured, like above, in the *strict accuracy*. The results show that suffix characters embedding brings the most benefits. This is consistent with the intuition, that in the case of words that are blured by noise inside them and OOV words their suffix can tell us most about their morphology. Also, adding rule-based post-processing to the tagger output increased its final accuracy.

On the basis of the experiments, for the final version of the system we chose the input vector consisting of the following components: morphological information, suffix characters embedding, suffix embedding, fastText embedding and clustering embedding for 1,750 clusters.

After selecting the network architecture parameters, we made an additional experiment. Using cross-validation we conducted a test during which we trained the tagger on the two combined data sets, namely: the manually annotated part of NCP and a subset of the manually annotated part of CCPL (i.e. the training folds). The manually annotated CCPL subcorpus was divided into five parts further on referred to as *folds*. The sub-models during the cross-validation process were trained on an NCP with four folds from CCPL and the sub-model was tested on the fifth fold from CCPL. Our tagger tested in this way achieved an average accuracy of 90.14%.

Finally, we compared the version of our CMC Tagger trained on the combined NCP and four folds of CCPL with the two taggers for Polish treated as a baseline, namely:

- *MorphoDiTa-pl* (Piasecki and Walentynowicz, 2017) is accessible at `http://ws.clarin-pl.eu/tager.shtml` and its source code at `https://github.com/ufal/morphodita`.

- *Toygger*, already mentioned, originally trained on NCP with *word2vec* embedding and suffix information feature, with 20 epochs.

The results of the tests done on the folds of CCPL are presented in Table 3.

Tagger	Accuracy (strong)
CMC Tagger	**90.14%**
MorphoDiTa	81.32%
Toygger	86.12%

Table 3: Strong accuracy (identical morphosyntactic class and values of grammatical categories) measured on CCPL corpus.

5.2 Results

We performed detailed linguistic error analysis. It covered the word forms differently tagged by human annotator and morphological tagger. 600 word forms most frequently judged inconsistently (3,675 error instances) were analysed. Among them several error types can be distinguished that mainly correspond to grammatical categories incorrectly recognised and every tuple (word form – human annotator tag – automatically ascribed tag) was assigned to one of the categories presented in Table 4.

The most common error concerned grammatical class. The most frequently confused classes are: adverb – particle-adverb (174 of 3,675 error instances) and coordinating conjunction – particle-adverb (90 instances). The word that appeared to be most difficult to judge was *to* (‘this’, ‘then’, ‘to be’, adverb) (263 error instances), which could be interpreted as adjective, predicative, noun, particle-adverb or subordinating conjunction depending on the context. Tagger had also problems with emojis correct recognition (177 instances).

Generally, the issues described above concern parts of speech that are very often the source of confusion for human annotators. Furthermore, the distribution of tagger errors is very similar to the observed distribution of inconsistencies in manual CCPL tagging. This shows that the mistakes are related to the difficult cases in general. A few use cases from the test dataset are presented below.

In the original, the fragment of sentence looks as follows:
“*[...] a mu sie nie spodoba i po wezystkm.*”
Correctly (without typing errors) this sentence would look like this:
“*[...] a mu się nie spodoba i po wszystkim.*”
(English: “*and he will not like it, and this is all.*”).

The CMC Tagger output for this sentence is shown in Table 5. Our tagger for the word *sie* (reciprocal participle *się* but written without diacritics) wrongly chose adjective as the grammatical class. This problem originated from the morphological analysis. Word form *sie* exists in the dictionary and the tagger could not recognise it

object of inconsistency	percent
number of assigned categories	4.71%
base	6.80%
grammatical class	31.16%
case	14.45%
number	2.29%
gender	22.42%
rection	4.60%
person	1.41%
aspect	0.33%
human error	9.31%
other	2.53%

Table 4: Selected categories of the CMC Tagger errors.

Form	Tag
a	conj
mu	ppron3:sg:dat:m1:ter:nakc:npraep
sie	adj:pl:nom:m2:pos
nie	qub
spodoba	fin:sg:ter:perf
i	conj
po	prep:loc
wezystkm	subst:sg:loc:m1

Table 5: Example sentence number 1

as a corrupted form of *się*, which is a reflexive pronoun or reciprocal participle (an component of a compound verb). The form *wezystkm* (i.e. in its proper form: *wszystkim* 'everything'/'all of them(case:dative)') is an unknown (OOV) word for the morphological analyser, but our tagger made only a small mistake in the gender attribute – it should be neutral.

In the case of a sentence that is grammatically constructed correctly, like the one in Table 6, our tagger works quite well. This sentence in English looks as follows: "In crediting the car purchase all went very smoothly and quite decent repayments.".

Form	Tag
W	prep:loc:nwok
kredytowaniu	ger:sg:loc:n:imperf:aff
zakupu	subst:sg:gen:m3
auta	subst:sg:gen:n
poszło	praet:sg:n:perf
bardzo	adv:pos
sprawnie	adv:pos
i	conj
całkiem	adv
przyzwoite	adj:pl:nom:f:pos
spłaty	subst:pl:nom:f
.	interp

Table 6: Example sentence number 2

6 Conclusions

We presented that standard approaches to morpho-syntactic disambiguation must be adapted to specific domains of non-standard texts, like CMC and User Generated Content texts, e.g. including social media texts. Thus, we proposed significant expansions to the state-of-the-art tagger for Polish, namely *Toygger*, that resulted in large gain in performance in CMC texts, as measured on the manually annotated gold standard for this domain. In the proposed expansions we focused on the representation and appropriate encoding in the input vector the information about: suffix types, better word embedding models (i.e. taking into account the sub-word level) and word clusters generated on the basis of standard text, but enabling coarse-grained text representation.

In future we want to focus on improving the usage of the morpho-syntactic information, e.g. in a form of partial recognition of possible dependencies. We plan to expand morphological vectors with the representation of the manually written constraints. We also want to work on semi-automatic extraction of contextual post-processing rules for improvement of the tagger performance. At the same time we want research how to expand representation of words on character level to get better recognition of noisy elements in text. An important challenge is handling of the segmentation in a correct way or applying a more advanced normalisation process before tagging. The tagger is available on the open licence from: http://hdl.handle.net/11321/634.

Acknowledgments

Partially funded by: Project "SentiCognitiveServices – next generation service for automating voice of customer and social media support based on artificial intelligence methods" (POIR.01.01.01-00-0806/16), which is co-financed by European Union through European Regional Development Fund under the Smart Growth Programme 2014 – 2020.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146. https://doi.org/10.1162/tacl_a_00051.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Comput. Linguist.* 18(4):467–479. http://dl.acm.org/citation.cfm?id=176313.176316.

Witold Kieraś and Marcin Woliński. 2017. Morfeusz2 - analizator i generator fleksyjny dla języka polskiego. *Język Polski* pages 75–83.

Łukasz Kobyliński and Maciej Ogrodniczuk. 2017. Results of the poleval 2017 competition: part-of-speech tagging shared task. In Zygmunt Vetulani and Patrick Paroubek, editors, *Human language technologies as a challenge for computer science and linguistics: 8th language & technology conference : November 17-19, 2017, Poznań, Poland : proceedings*, Fundacja Uniwersytetu im. Adama Mickiewicza, Poznań, pages 362–366. http://ltc.amu.edu.pl/book/papers/PolEval1-2.pdf.

Jan Kocoń. 2018. KGR10 FastText polish word embeddings. CLARIN-PL digital repository. http://hdl.handle.net/11321/606.

Katarzyna Krasnowska-Kieraś. 2017. Morphosyntactic disambiguation for polish with bi-lstm neural networks. In Zygmunt Vetulani and Patrick Paroubek, editors, *Human language technologies as a challenge for computer science and linguistics: 8th language & technology conference : November 17-19, 2017, Poznań, Poland : proceedings*, Fundacja Uniwersytetu im. Adama Mickiewicza, Poznań, pages 367–371. http://ltc.amu.edu.pl/book/papers/PolEval1-2.pdf.

Nikola Ljubešić. 2018. Comparing CRF and LSTM performance on the task of morphosyntactic tagging of non-standard varieties of south Slavic languages. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pages 156–163. https://www.aclweb.org/anthology/W18-3917.

Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2017. Adapting a state-of-the-art tagger for south Slavic languages to non-standard text. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*. Association for Computational Linguistics, Valencia, Spain, pages 60–68. https://doi.org/10.18653/v1/W17-1410.

Robert Östling. 2018. Part of speech tagging: Shallow or deep learning? *Northern European Journal of Language Technology (NEJLT)* 5:1–15. http://www.nejlt.ep.liu.se/2018/v5/a01/index.html.

Maciej Piasecki and Wiktor Walentynowicz. 2017. Morphodita-based tagger adapted to the polish language technology pages 377–381. http://ltc.amu.edu.pl/book/papers/PolEval1-2.pdf.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *CoRR* abs/1604.05529. http://arxiv.org/abs/1604.05529.

Agnieszka Pluwak, Wojciech Korczynski, and Marek Kisiel-Dorohinicki. 2016. Adapting a constituency parser to user-generated content in polish opinion mining. *Computer Science (AGH)* 17:23–44.

Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. 2012. *Narodowy Korpus Języka Polskiego [in Polish]*. Wydawnictwo Naukowe PWN. http://nkjp.pl/settings/papers/NKJP_ksiazka.pdf.

Adam Radziszewski and Tomasz Śniatowski. 2011. MACA. CLARIN-PL digital repository, `http://hdl.handle.net/11321/20`. http://hdl.handle.net/11321/20.

Piotr Rybak and Alina Wróblewska. 2018. Semi-supervised neural system for tagging, parsing and lematization. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium, pages 45–54. https://www.aclweb.org/anthology/K18-2004.

Miikka Silfverberg and Senka Drobac. 2018. Sub-label dependencies for neural morphological tagging – the joint submission of university of colorado and university of Helsinki for VarDial 2018. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pages 37–45. https://www.aclweb.org/anthology/W18-3904.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2):26–31.

Jakub Waszczuk. 2012. Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language. *Proceedings of COLING 2012* pages 2789–2804.

Krzysztof Wróbel. 2017. Krnnt: Polish recurrent neural network tagger. In Zygmunt Vetulani and Patrick Paroubek, editors, *Human language technologies as a challenge for computer science and linguistics: 8th language & technology conference : November 17-19, 2017, Poznań, Poland : proceedings*, Fundacja Uniwersytetu im. Adama Mickiewicza, Poznań, pages 386–391. http://ltc.amu.edu.pl/book/papers/PolEval1-6.pdf.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Ahmed Ali, Suwon Shon, James Glass, Yves Scherrer, Tanja Samardžić, Nikola Ljubešić, Jörg Tiedemann, Chris van der Lee, Stefan Grondelaers, Nelleke Oostdijk, Dirk Speelman, Antal van den Bosch, Ritesh Kumar, Bornini Lahiri, and Mayank Jain. 2018. Language identification and morphosyntactic tagging: The second vardial evaluation campaign. In Marcos Zampieri, Preslav Nakov, Nikola Ljubešić, Jörg Tiedemann, Shervin Malmasi, and Ahmed Ali, editors, *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, Association for Computational Linguistics, Santa Fe, New Mexico, USA. http://hdl.handle.net/10138/249333.

Evaluation of vector embedding models in clustering of text documents

Tomasz Walkowiak
University of Science and Technology,
Faculty of Electronics,
Wybrzeze Wyspianskiego 27,
Wroclaw 50-370, Poland
tomasz.walkowiak@pwr.wroc.pl

Mateusz Gniewkowski
University of Science and Technology,
Faculty of Computer Science and Management,
Wybrzeze Wyspianskiego 27,
Wroclaw 50-370, Poland
mateusz.gniewkowski@pwr.wroc.pl

Abstract

The paper presents an evaluation of word embedding models in clustering of texts in the Polish language. Authors verified six different embedding models, starting from widely used word2vec, across fastText with character n-grams embedding, to deep learning-based ELMo and BERT. Moreover, four standardisation methods, three distance measures and four clustering methods were evaluated. The analysis was performed on two corpora of texts in Polish classified into subjects. The Adjusted Mutual Information (AMI) metric was used to verify the quality of clustering results. The performed experiments show that Skipgram models with n-grams character embedding, built on KGR10 corpus and provided by Clarin-PL, outperforms other publicly available models for Polish. Moreover, presented results suggest that Yeo–Johnson transformation for document vectors standardisation and Agglomerative Clustering with a cosine distance should be used for grouping of text documents.

1 Introduction

A number of digital repositories of texts enlarge each year. The variety of tools for natural language processing and the quality of their performance are growing steadily. That opens possibilities for automatic categorisation of text documents in terms of the subject areas in any digital collection of documents. It is also an important problem for researchers from different areas of the humanities and social science (Eder et al., 2017).

Commonly used methods rely on representing documents with feature vectors and using clustering algorithm (Hastie et al., 2009) to assign documents to some groups. The classical feature vectors are based on the bag-of-words technique (Harris, 1954). Components of these vectors represent frequencies (weighted) of occurrences of words/terms in individual documents. The contemporary state-of-the-art technique is word2vec (Le and Mikolov, 2014), where individual words are represented by high-dimensional feature vectors trained on large text corpora. This technique is constantly being improved. This is demonstrated by the most recent propositions of algorithms like ELMo (Peters et al., 2018) or BERT (Devlin et al., 2018). Choosing the most useful clustering algorithm is not a trivial task since there is a large number of them. Just to mention the most popular ones like K-means (K.Jain, 2009), Agglomerative Hierarchical Clustering (Day and Edelsbrunner, 1984) and Spectral Clustering (Ng et al., 2002). Moreover, the results of clustering are strongly dependent on the chosen distance measure and the used method of an input data standardisation. The entire workflow, described above, expresses many factors which can influence results of the text exploration. It is difficult to control them and thus, might lead to unpredictable outcomes of the experiment. It becomes challenging for texts in an inflected language such as Polish.

The main aim of this research is the evaluation of clustering accuracy on documents in Polish, using publicly available word embedding models. We conducted our experiments to answer the following research questions: What is the best method (i.e. the word2vec model, standardisation method, distance metric and clustering algorithm) for subject grouping texts in Polish? The experiments were performed on two corpora with texts assigned to subject groups. We analysed the quality of results (defined by AMI metric (Romano et al., 2016)) in a function of method options.

Proceedings of Recent Advances in Natural Language Processing, pages 1304–1311,
Varna, Bulgaria, Sep 2–4 2019.

Some works studied the quality of word2vec models for Polish (Piasecki et al., 2018; Mykowiecka et al., 2017; Kocon and Gawor, 2019), but they focus on single words, not on an application of the word embeddings to represent whole documents for a clustering purpose.

The paper is structured as follows. In Section 2, we describe in details word embedding techniques and list the models examined in the work. Next, we provide technicalities about the methods we compared and finally in Section 4 we present test corpora used in the study as well as results of the comparative study.

2 Word Embeddings

2.1 Techniques of Word Embedding

Word embedding is an approach of text analysis based on the assumption that individual words can be represented by high-dimensional feature vectors. It is based on the hypothesis that relationships (distances) between vector representations of words can be related to semantic similarities of words. The models are built on large text corpora by observing co-occurrence of words in similar contexts. One of the most popular technique, *word2vec*, is based on neural networks (Le and Mikolov, 2014). The authors proposed two approaches: CBOW and Skipgram. In the first one, the aim is to predict a word based on context words. The Skipgram model does the opposite task, it predicts context words from a given word. In the classical word2vec (Le and Mikolov, 2014) technique each word (form from the text) is represented by a distinct vector, which might be a problem for a language with large vocabularies and rich inflexion like Polish is. The first solution to this problem was to build models based on word lemmas (Mykowiecka et al., 2017), however, such a technique requires a morphological analysis of all texts in a training corpus. Next, in (Bojanowski et al., 2017) authors extend the Skipgram model by building a vector representation of character n-grams and constructing the word representation as the sum of the character n-grams embeddings (for n-grams appearing in the word). It could decree the model size and allows to generate word embedding for words not seen in a training corpus.

The next step forward was introducing (Grave et al., 2018) the extension of the original CBOW model (Le and Mikolov, 2014) with position weights and subword information (character n-grams).

The newest approaches are inspired by deep-learning algorithms. In a recently introduced ELMo (Peters et al., 2018), word embeddings are defined by the internal states of a deep bidirectional LTSM language model (biLSTM), which is trained on a large text corpus. What is important, ELMo looks at the whole sentence before assigning an embedding to each word in it. Therefore, the embeddings are sentence aware and could solve a problem of polysemous words (words with multiple meanings). Another approach similar to ELMo is BERT (Devlin et al., 2018). It is also a bidirectional representation, but it is jointly built on both the left and the right context. The available BERT models[1] are multilingual and pre-trained on two unsupervised tasks: masked language modelling and next sentence prediction.

Word embeddings can be simply used to generate feature vectors for document clustering by averaging vector representations of individual words occurring in a document. This approach is known as *doc2vec* and used for example in fastText (Joulin et al., 2017) algorithm. In the case of BERT, we get sentence embeddings, but the approach used for word models can be repeated here as well (as an average of sentence embeddings).

2.2 Available Models for Polish

There are two groups working on publicly available word embedding models for Polish: IPI PAN[2] and Clarin-PL[3]. The IPI PAN provides[4] a set of more than 100 CBOW and Skipgram models generated from data consisting of National Corpus of Polish (NKJP) (Przepiorkowski et al., 2012) and Wikipedia (Wiki). Some of them are generated only for lemmas, others of words from texts (forms). For tests we have selected the Skipgram model (i.e. *nkjp+wiki-forms-all-300-skipg-hs-50*). The model was generated by gensim tool[5]. It assigns a distinct vector to each word.

The Clarin-PL provides[6] 16 models generated by fastText software[7] on larger than in a previous case corpus (Kocon and Gawor, 2019). They are joint models of words and character n-grams

[1] https://github.com/google-research/bert

[2] https://ipipan.waw.pl/en/

[3] https://clarin-pl.eu/en/home-page/

[4] http://dsmodels.nlp.ipipan.waw.pl

[5] https://radimrehurek.com/gensim/

[6] http://hdl.handle.net/11321/606

[7] https://fasttext.cc/

able to produce vectors for unknown words (Bojanowski et al., 2017). For tests, we have selected two Skipgram models based on forms (*KGR10*) and lemmas (*KGR10_lemma*). The second group of sources of word2vec models for Polish are web pages of word embedding tools like fastText, ELMo and BERT. They were trained on Polish Common Crawl and Wikipedia. However, the BERT model was trained on many languages in parallel. The details on used models are summarised in Table 1.

3 Methods

3.1 Metrics and Clustering

A distance measure needs to distinguish contrasting samples. Sometimes that contrast is well defined, and we know what kind of behaviour we require from the chosen function. In natural language processing commonly used function is a cosine distance as i.e. it does not distinguish documents, described as a vector of most frequently occurred words in the corpus, that have a linear dependence between features. It also works well with sparse high-dimensional space and is less *noisy* than euclidean (Kriegel H-P., 2012). Models we want to compare have different properties, so relevant distance function is even less obvious. In our work we decided to focus on *cosine*, *Bray–Curtis* and *Euclidean distance*.

We also tried few different variations of them but the results were not significantly different.

In order to group data, we decided to use following methods (mind that two of them use only Euclidean distance or L_k norm in general, because otherwise algorithms may stop converging).

1. *K-means* (K.Jain, 2009) algorithm is a classic method that assigns labels to the data, basing on a distance to the nearest centroid. Centroids are moved iteratively until all clusters stabilise.

2. *Agglomerative hierarchical clustering (AC)* (Day and Edelsbrunner, 1984) is a method that iteratively joins subgroups basing on a *linkage criterion*. In this paper, we present result for the average linkage clustering.

3. *Spectral clustering (SC)* (Ng et al., 2002) is based on the Laplacian matrix of the similarity graph and its eigenvectors. The least significant eigenvectors create new, lower dimensional space that is used with a *K-means* algorithm.

4. *Expectation–maximisation (EM)* (Bilmes et al., 1998) is used to estimate parameters in statistical models, Gaussian mixture model in our example. Gaussian mixture model assumes that data is generated from a finite number of Gaussian distributions.

3.2 Standardisation

Standardisation of input data is usually necessary, because many machine learning algorithms requires features to have a normal distribution and, probably more important in clustering algorithms, a similar scale. In our work, we decided to use following methods:

Table 2: Symbols description

X	feature vector (column of the input matrix)
$\overline{X}$	mean value of the feature
X_{min}, X_{max}	minimal/maximal value of the feature
$x_i, \hat{x}_i$	new/old value of the feature of i-th sample
σ	standard deviation of the feature
λ	power parameter that is estimated through maximum likelihood

1. Min–Max scaling - the most popular way of scaling data. Returned values are in range from 0 to 1:

$$\hat{x}_i = \frac{x_i - X_{min}}{X_{max} - X_{min}}$$

2. Z-score normalisation - one of the classic method of standardisation. It results in a distribution with a standard deviation equal to 1:

$$\hat{x}_i = \frac{x_i - \overline{X}}{\sigma}$$

3. Yeo–Johnson transformation - a member of power transform functions that allows negative values of input (Yeo and Johnson, 2000):

$$\hat{x}_i = \begin{cases} \frac{(x_i+1)^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0, x \geq 0 \\ \log(x_i + 1), & \text{if } \lambda = 0, x \geq 0 \\ -\frac{(-x_i+1)^(2-\lambda)-1}{2-\lambda}, & \text{if } \lambda \neq 2, x < 0 \\ -\log(-x_i + 1), & \text{if } \lambda = 2, x < 0 \end{cases}$$

[8] CBOW with position weights

Table 1: Used embedding models

name	method	feature	tool	size	address
IPIPAN	CBOW	forms	gensim	300	dsmodels.nlp.ipipan.waw.pl
KGR10	Skipgram	forms, character n-grams	fastText	300	hdl.handle.net/11321/606
KGR10_lemma	Skipgram	lemmas,character n-grams	fastText	300	hhdl.handle.net/11321/606
fastText	CBOW+[8]	orths, character 5-grams	fastText	300	fasttext.cc
elmo	ELMo	forms	ELMo	1024	vectors.nlpl.eu/repository/11/167.zip
bert	BERT	multilingual forms, sentences	BERT	768	github.com/google-research/bert

3.3 Quality Metrics

Evaluation of clustering quality may be performed in two different ways: with external knowledge of sample membership or without it. The first way is usually better if we have already labelled data and using supervised learning is none of our options. For example, we know that the clustering problem we want to solve concerns similar data we have labelled, which is a case in this work. We compare how different vector representation of documents, which have an assigned label to it, can be clustered.

There are plenty of clustering quality measures that have a different interpretations, like *purity*, *V-measure* or *Rand Index* (Amigo et al., 2009). In our work, we decided to use corrected for a chance measures where the result does not increase with several clusters for randomly chosen labels. Two most common metrics are *Adjusted Rand Index* (ARI) and *Adjusted Mutual Information* (AMI) (Vinh et al., 2010). According to (Romano et al., 2016) AMI is better suited to our problem as documents types are often unbalanced. It gives more weight to a clustering solutions with purer small groups than to minor mistakes in bigger ones.

Adjusted mutual information score is one of the information theoretically based measures. It is based on mutual information (MI) which comes naturally from entropy.

Table 3: Symbols description

X, Y	set of classes/clusters
H	Entropy
MI	mutual information
NMI	normalized mutual information
AMI	adjusted mutual information
x_i, y_i	i-th element of X/Y (class or cluster)
$P(x_i), P(y_i)$	probability of the document being in i-th class or cluster
$P(x_i\hat{y}_j)$	intersection of $P(x_i)$ and $P(y_j)$
$E(MI)$	expected value of MI

$$H(X) = \sum_i P(x_i) \log \frac{1}{P(x_i)}$$

$$MI(X,Y) = \sum_i \sum_j P(x_i \cap y_j) \log \frac{P(x_i \cap y_j)}{P(x_i)P(y_i)}$$

The problem with mutual information is that the maximum is reached not only when labels from one set (clusters) matches perfectly those from the other (classes), but also when they are further subdivided. The simple solution for that is to normalise MI by mean of entropy of X and Y:

$$NMI(X,Y) = \frac{MI(X,Y)}{(H(X) + H(Y))/2}$$

Normalised mutual information can be further improved by subtracting expected value of MI from nominator and denominator:

$$AMI(X,Y) = \frac{MI(X,Y) - E(MI)}{(H(X) + H(Y)/2 - E(MI)}$$

This is what is called ”corrected for a chance”. The general form of AMI was proposed in (Hubert and Arabie, 1985).

4 Experiments

4.1 Data Sets

We performed tests on two collections of text documents in Polish: $Press$ and $Rewievs$. The first corpus ($Press$) comprises Polish press news. It is a complete, high quality and well defined data set. The texts were assigned by press agency to five subject categories. All the subject groups are well separable from each other and each group contains a reasonably large number of members. In the study (Walkowiak and Malak, 2018), the authors reported 95.5% accuracy achieved on this data set in supervised classification. There are ca. $6,500$ documents in total in this corpus.

The second corpus (*Reviews*) consists of reviews of scientific works from 21 different science areas. The achieved accuracy on this data set by fastText (Joulin et al., 2017) in supervised classification was 90.7% (after division 2:1 for training and testing). There are ca. $10,500$ documents in this corpus.

4.2 Idea

The goal of our experiments was to find the best performing word embedding model in a clustering problem. In order to do that, first, we checked how standardisation affects results and picked one of the methods to use it in further tests. Then we compared several models using different clustering approaches with and without standardisation. In order to generate feature vectors for documents (doc2vec) we averaged word/sentence embeddings for every text in the dataset.

4.3 Choosing the Standardisation Method

We performed our first experiment as follows: having the documents $d_i \in D$ represented as doc2vec vectors from KGR10_lemma model, we performed several tests to evaluate the quality measure (AMI) of multiple clustering algorithms with different distance functions. The results are given in the Figure 1 and Figure 2. It can be observed that for EM, K-means and SC standardisation does not significantly improve the results. What is more, for Euclidean distance, data scaling may blur the distances between points and worsen the quality of the solution. On the other hand, usage of standardisation methods with Agglomerative Clustering (AC) algorithm improves obtained results. It is not surprising as the linkage method strongly depends on variance especially when using a cosine distance. On average (the average height of the AMI score) the best method of standardisation turned out to be Yeo–Johnson transformation, so we used it in subsequent experiments.

4.4 Model Comparison

In order to compare how the chosen model affects quality, we performed multiple tests, similar to the previously conducted. They were evaluating the quality measure due to used doc2vec representations, generated from models described in Section 2. The results of clusterisation of the original data can be observed in figures 3 and 5 alongside with the results of standardised vectors with Yeo–Johnson transformation in figures 4 and 6. It can be noticed that standardisation has minor influence on Spectral Clustering (SC). It either slightly improves or does not deteriorate results. The only exception is Euclidean distance where standardisation can blur and therefore worsen the score. It is clearly visible for Agglomerative Clustering (AC) which, after standardisation and with a cosine or a Bray–Courtis distance, works best. Using standardised version of K-means algorithm with any of proposed models is rather not suitable since standardisation does not necessarily increase the score, however this classic approach is usable with the given problem, especially that it strongly depends on centroids which usually are quite useful. We expected that standardisation will have a positive influence on Gaussian mixture model (EM) which assumes data is generated from some number of Gaussian distributions and standardisation should make that data more Gaussian-like. It is probably the case with those models that have higher score in figures 4 and 6 than in figures 3 and 5, but it is not a rule.

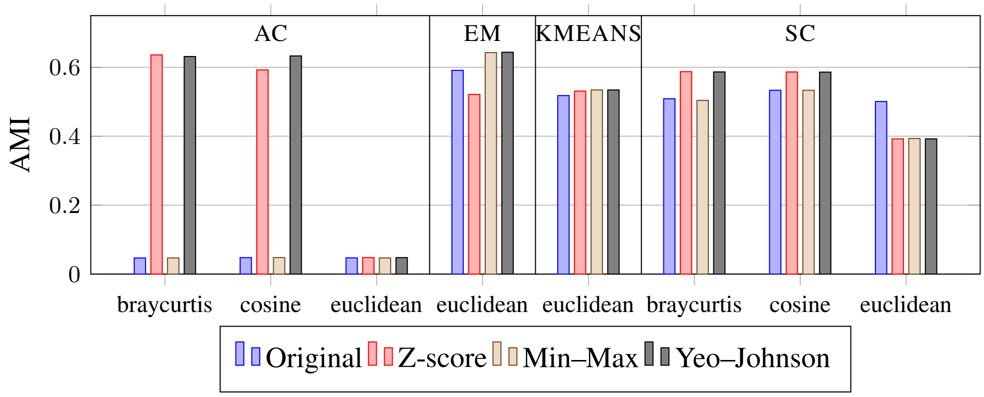


Figure 1: Standarization - PRESS

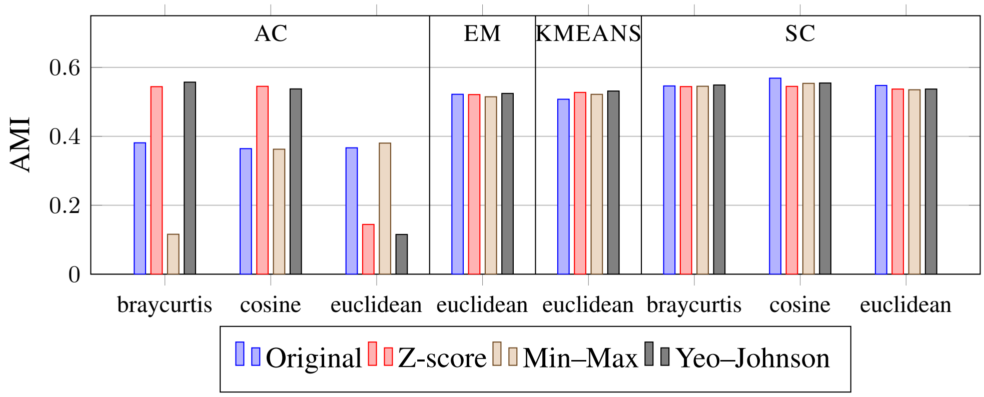


Figure 2: Standarization - REVIEWS

5 Conclusion

As a conclusion, we would like to recommend using Yeo–Johnson transformation to standardise doc2vec embedding and if a task is to group documents, Agglomerative Clustering (AC) with a cosine distance. For researchers who deal with Polish datasets, we strongly recommend using KGR10_lemma or KGR10 word2vec models (Kocon and Gawor, 2019)[9]. First of them gives better results but the second one (only slightly worse) is much faster in a usage, since it does not require a time consuming lemmatization of texts. We are now working on implementing the best selected workflow as a part of WebSty (Eder et al., 2017), an online tool[10] aimed for researchers in humanities and social science working with texts in Polish.

Although, that ELMo and BERT perform great in many tasks in NLP our results show otherwise. Two factors can be responsible for this. First, we used already trained models downloaded from the addresses shown in Table. 1. As they might not be the best quality for the Polish language, we currently training our own model to verify this hypothesis. The second reason may be that both BERT and ELMo do not work well with the discussed problem. It is hard to find any article dealing with document clustering problem using those methods.

We plan to test other methods of composing document vectors, i.e. representing documents by several concatenated vectors. We want to test two approaches. One is based on a division of documents into parts and generating doc2vec for each part. And the second, based on clustering of word embeddings into a predefined number of groups and using centroids as elements of final document vectors.

The work was funded by the Polish Ministry of Science and Higher Education within CLARIN-PL Research Infrastructure.

[9] http://hdl.handle.net/11321/606

[10] http://ws.clarin-pl.eu/websty.shtml

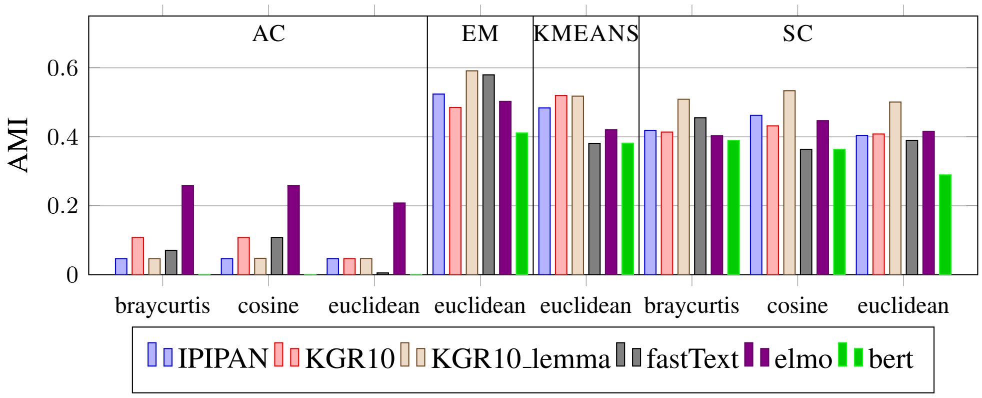


Figure 3: Model Comparison - PRESS

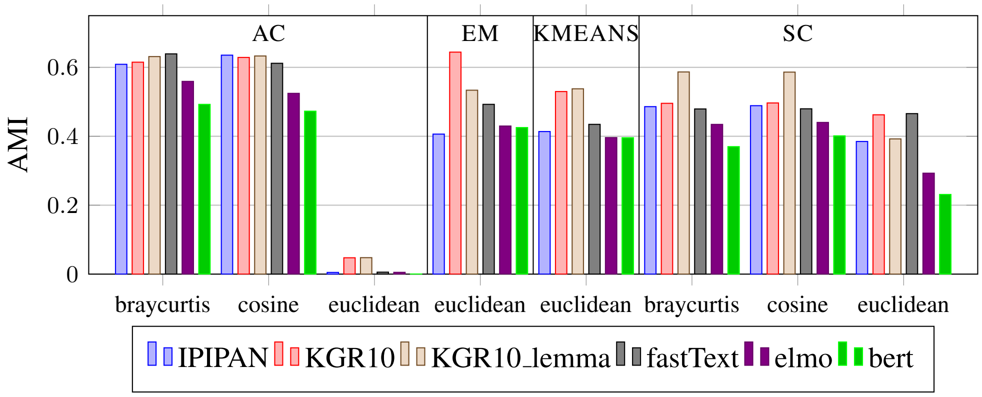


Figure 4: Model Comparison - PRESS (standardised with Yeo–Johnson transformation)

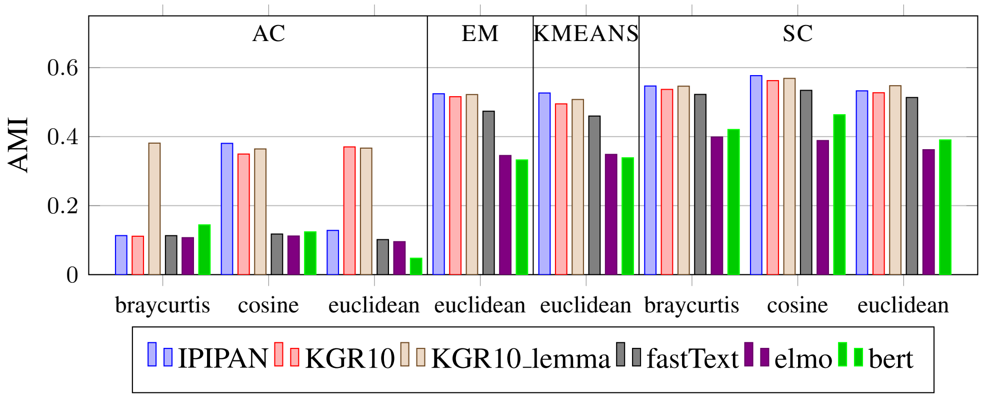


Figure 5: Model Comparison - REVIEWS

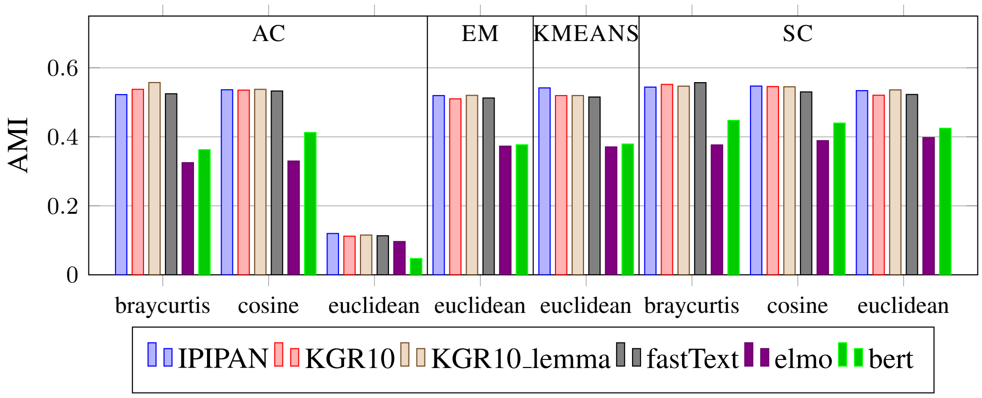


Figure 6: Model Comparison - REVIEWS (standardised with Yeo–Johnson transformation)

References

Enrique Amigo, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval* 12(4):461–486.

Jeff A Bilmes et al. 1998. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute* 4(510):126.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5:135–146.

William H. E. Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification* 1(1):7–24. https://doi.org/10.1007/BF01890115.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Maciej Eder, Maciej Piasecki, and Tomasz Walkowiak. 2017. An open stylometric system based on multilevel text analysis. *Cognitive Studies — Etudes cognitives* 17. https://doi.org/10.11649/cs.1430.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Zellig S Harris. 1954. Distributional structure. *Word* 10(2-3):146–162.

Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer series in statistics. Springer, New York. Autres impressions : 2011 (corr.), 2013 (7e corr.).

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification* 2(1):193–218. https://doi.org/10.1007/BF01908075.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pages 427–431. http://aclweb.org/anthology/E17-2068.

Anil K.Jain. 2009. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters, Volume 31, Issue 8,* pages 651–666. https://doi.org/10.1016/j.patrec.2009.09.011.

Jan Kocon and Michal Gawor. 2019. Evaluating KGR10 polish word embeddings in the recognition of temporal expressions using bilstm-crf. *CoRR* abs/1904.04055. http://arxiv.org/abs/1904.04055.

Schubert E. Zimek A. Kriegel H-P. 2012. A survey on unsupervised outlier detection. *Statistical Analysis and Data Mining* pages 363–387.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. pages 1188–1196.

A. Mykowiecka, M. Marciniak, and P. Rychlik. 2017. Testing word embeddings for polish. *Cognitive Studies — Etudes cognitives* 17. https://doi.org/10.11649/cs.1468.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*. pages 849–856.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Maciej Piasecki, Gabriela Czachor, Arkadiusz Janz, Dominik Kaszewski, and Pawel Kedzia. 2018. Wordnet-based evaluation of large distributional models for polish. In *Proceedings of the 9th Global Wordnet Conference (GWC 2018)*. Global WordNet Association.

A. Przepiorkowski, M. Banko, R. L. Gorski, and B. Lewandowska-Tomaszczyk, editors. 2012. *Narodowy Korpus Jzyka Polskiego*. Wydawnictwo Naukowe PWN, Warszawa.

Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. 2016. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research* 17(1):4635–4666.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11(Oct):2837–2854.

Tomasz Walkowiak and Piotr Malak. 2018. Polish texts topic classification evaluation. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*. INSTICC, SciTePress, pages 515–522. https://doi.org/10.5220/0006601605150522.

In-Kwon Yeo and Richard A Johnson. 2000. A new family of power transformations to improve normality or symmetry. *Biometrika* 87(4):954–959.

Bigger versus Similar: Selecting a Background Corpus for First Story Detection Based on Distributional Similarity

Fei Wang and **Robert J. Ross** and **John D. Kelleher**
Technological University Dublin
ADAPT Research Centre
d13122837@mydit.ie
{robert.ross, john.d.kelleher}@dit.ie

Abstract

The current state of the art for First Story Detection (FSD) are nearest neighbour-based models with traditional term vector representations; however, one challenge faced by FSD models is that the document representation is usually defined by the vocabulary and term frequency from a background corpus. Consequently, the ideal background corpus should arguably be both large-scale to ensure adequate term coverage, and similar to the target domain in terms of the language distribution. However, given these two factors cannot always be mutually satisfied, in this paper we examine whether the distributional similarity of common terms is more important than the scale of common terms for FSD. As a basis for our analysis we propose a set of metrics to quantitatively measure the scale of common terms and the distributional similarity between corpora. Using these metrics we rank different background corpora relative to a target corpus. We also apply models based on different background corpora to the FSD task. Our results show that term distributional similarity is more predictive of good FSD performance than the scale of common terms; and, thus we demonstrate that a smaller recent domain-related corpus will be more suitable than a very large-scale general corpus for FSD.

1 Introduction

Given a stream of documents about news events in a chronological order, the goal of First Story Detection (FSD) is to identify the very first story for each event. Each story is processed in sequence, and a decision is made for a given candidate document on whether or not it discusses an event that has not been seen in previous documents; crucially this decision is made after processing the candidate document but before processing any subsequent documents (Allan et al., 1998; Yang et al., 1998). The decision making process for each incoming document is normally based on a novelty score; namely, if the novelty score of a new document is higher than a given threshold, we say it is a first story.

Hundreds of FSD models have been proposed in prior research, and the nearest neighbour-based models, in which the novelty score is defined as the distance from the new story to the closest existing story, remain the state of the art (Wang et al., 2018). In the implementation of nearest neighbour-based FSD models, the first step is to represent each story with a sound document representation. Even though many deep learning-based document representations have been shown to achieve very good results in a range of NLP (Natural Language Processing) tasks (Goldberg, 2017), the dominant document representation model for FSD remains the traditional term vector models in which each feature represents a term in the vocabulary (Brants et al., 2003; Petrović et al., 2010; Wang et al., 2018; Kannan et al., 2018).

The majority of machine learning research assumes that the data used for building a model and making inference are sampled from the same distribution, i.e., the data generation process is stationary. However, because of its online characteristic, one challenge faced by FSD models is that the system's vocabulary (and hence document representation) cannot be derived from a target corpus, but must instead be defined by the vocabulary of a background corpus. The resultant potential difference between the background and target corpus demonstrates a non-stationary characteristic

Proceedings of Recent Advances in Natural Language Processing, pages 1312–1320,
Varna, Bulgaria, Sep 2–4 2019.

of FSD. To mitigate for potential differences between background and target data, the ideal background corpus should be both large-scale, so as to ensure an adequate number of common terms between it and the documents in the target stream (i.e., minimize *unknown* words), and similar in the sense of language distribution. In many cases, these two factors cannot be satisfied at the same time, and thus the emphasis has to be placed on the more informative one of the two, which leads to a question of "bigger or similar?". To the best of our knowledge however, there is little research addressing this question empirically, and no metrics have been proposed for the quantitative comparison of the scale and similarity between background corpora relative to a target corpus.

In this paper we examine whether the distributional similarity of common terms between corpora (background and target story stream) is more important than the scale of common terms for FSD. As a basis for our analysis we propose a set of metrics to quantitatively measure the scale of common terms and the distributional similarity between corpora. Using these metrics we rank different background corpora relative to a target FSD corpus. Finally, we apply the models based on different background corpora to the FSD task to determine the relative utility of different assumptions about the background corpus. Our contributions are thus two-fold: an investigation of background corpus similarity versus scale, and a metrics framework for making such an investigation.

2 First Story Detection

FSD as a challenge was initially defined within the Topic Detection and Tracking (TDT) competition series (Yang et al., 1998; Allan et al., 2000b); and was considered to be the most difficult challenge in all five TDT tasks (Allan et al., 2000a). Since then, the need for accurate FSD models has been greatly strengthened by the proliferation of digital content, and social media streams in particular. One of the challenges of FSD is the undefinable characteristic of a first story. We can never know what the next first story will look like; instead, we only know that it must be different to existing stories to some degree. Therefore, we normally consider FSD as an unsupervised learning application, and hence attempt to define and make use of a novelty score in a similar fashion to how novelty-style metrics are defined in other unsupervised learning applications (Wang et al., 2017).

Based on different definitions of novelty scores, it has been proposed that there are three categories of FSD models (Wang et al., 2018): Point-to-Point (P2P) models, Point-to-Cluster (P2C) models, and Point-to-All (P2A) models. P2P models, in which the novelty score is defined as the distance from the incoming story to an existing story, are normally nearest neighbour-based (Yang et al., 1998; Allan et al., 2000b), or approximate nearest neighbour-based (Brants et al., 2003; Petrović et al., 2010, 2012; Moran et al., 2016; Kannan et al., 2018). In P2C models or P2A models, the novelty score is defined respectively as the distance from the new story to a cluster of existing stories (also can be considered as the distance to an existing event), or to all the existing stories. The former is usually clustering-based (Yang et al., 1998; Allan et al., 2000b; Li et al., 2017), and the latter uses all the existing data to build a system, and applies this system to the incoming story to generate a novelty score (Schölkopf et al., 2001; Wurzer et al., 2015). Based on previous literature and research on FSD, it has been shown that nearest neighbour-based P2P models perform the best among all these three categories of FSD models (Wang et al., 2018).

2.1 Term Vector Models for First Story Detection

As presented above, the novelty score in a P2P model is calculated by comparing the incoming story to previous stories and then finding its (approximate) nearest neighbour and the corresponding closest distance. When implementing a P2P model, the first step is to convert the raw stories to document representation vectors that can be fed into the detection model; this is then followed by the quantitative comparisons between these document representations. The state of the art document representation model for P2P FSD models remains the traditional term vector models, due, in part, to their specificity of terms (Wang et al., 2018). In a term vector model, each feature (dimension) represents a term in the vocabulary, so the dimensionality of each vector is generally the same length as the corpus vocabulary.

TF-IDF is the most well-known term vector model and also the most effective one used for FSD (Brants et al., 2003; Petrović et al., 2010; Kannan et al., 2018). A TF-IDF weight is calculated for each term in a document vector as the product of the TF (term frequency) and IDF (inverse document frequency) components. The TF component captures the number of times a term was encountered in the document, while the IDF component discounts the term weights that are very common in the corpus such that these are judged to have little information relevant to the distinction of documents. A TF-IDF model always stores a vocabulary as well as an IDF dictionary in which the key is each term while the value is the corresponding IDF component for that term. When applying a TF-IDF model, it is necessary to use some corpus to build the vocabulary and the IDF dictionary before calculating the TF-IDF weight for each term in a document using some specific scheme. A widely-applied TF-IDF weighting scheme is shown as follows:

$$tf\text{-}idf(t,d) = tf(t,d) \times idf(t) \tag{1}$$

$$idf(t) = log\frac{N}{df(t)} \tag{2}$$

where $tf(t,d)$ represents the TF component, that is just the number of times the term t occurs in document d, and $idf(t)$ represents the IDF component, in which N denotes the total number of documents and $df(t)$ refers to the number of documents that contain the term t.

In the context of FSD, the labelled target corpus is always unavailable before detection because of the online characteristic of FSD, and thus a background corpus is required to build the TF-IDF model, i.e., the vocabulary and the IDF dictionary in the model. As shown in Fig. 1, we assume that a TF-IDF model is built with a background Corpus B and is applied to the FSD task for a target Corpus T. Set 2 is the overlapping term set that contains the terms common to both Corpus B and T, and Set 1 and 3 contain the terms that only exist in Corpus B or T respectively. Consequently, Set 1 and 2 constitute all terms in Corpus B, while Set 2 and 3 constitute all terms in Corpus T.

2.2 Set Overlap and FSD Modelling

In a pre-built TF-IDF model, all the terms in the vocabulary are from the background Corpus B, i.e., the terms used to generate the term vector space are those from Set 1 and 2, while those terms in Set 3 will not appear in the TF-IDF model at all. In other words, the terms in Set 3 are all the *unknown* terms with respect to the TF-IDF model. However, when the TF-IDF model is applied to FSD, all the documents to be analysed will be from the target Corpus T, which means that all the terms in Set 1 will not appear at all in the process of FSD; as a result the TF components for these terms are always zero and thus all the final TF-IDF weights of these will always be zero as well. It should be noted that we can look at all the terms of the target corpus here because we are now doing the analysis. However, during the real FSD we will never know whether a term from the background corpus appears in the target corpus or not. Therefore, we have to keep all the terms in Set 1 and 2 that are from the background corpus, even though the weights of all the terms in Set 1 are always zero.

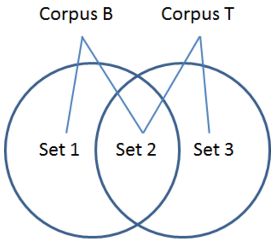


Figure 1: Term Sets within a Background Corpus B and a Target Corpus T

The comparison between TF-IDF representations is usually based on cosine distance calculations for FSD (Allan et al., 2000b; Brants et al., 2003). For such calculations, all representation vectors are normalised so that the cosine distance is not sensitive to the specific weighting schemes (Allan et al., 2000b). To clarify, given an incoming story represented by $\vec{a}$ and an existing story represented by $\vec{b}$, the cosine distance between them is defined as:

$$cosine\_distance(\vec{a}, \vec{b}) = 1 - \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\left|\vec{b}\right|} \tag{3}$$

According to the definition of cosine distance, in each document vector the terms whose weights are always zero do not have any effect on the result of calculation, so they can be ignored when we analyse the calculation. Hence, the valid terms that make sense for FSD are only those in Set 2, which are the common terms in both the background and target corpus. Given this, the effectiveness of the TF-IDF model only depends on Set 2, and specifically, two factors of Set 2: the scale and the distributional similarity between the background and target corpus. The scale describes the number of common terms between the corpora. The larger the scale of Set 2, the more informative terms are taken into account. The distributional similarity of two corpora refers to similarity of the frequencies of common terms. As the IDF components of these common terms are calculated only based on the background corpus, the more similar the background corpus is to the target corpus in terms of the language distribution, the better the generated weights can represent the common terms for FSD in the target corpus.

Therefore, the ideal background corpus for FSD should be both large-scale and similar in frequency distribution to the assumed target corpus. However, in many cases, these two factors cannot always be satisfied at the same time, and so it can be useful to determine which of these factors is more predictive of good FSD performance.

3 Quantitatively Measuring Background Corpus Suitability

To propose a method for evaluating the relative importance of the quantity of shared terms versus the similarity of language distributions between a background and target corpus, in this section we outline a set of quantitative metrics to make pairwise comparisons between different background corpora relative to the target FSD corpus.

3.1 Measuring the Scale of Common Terms

As shown earlier in Fig. 1, the scale of common terms relative to the target Corpus T can be quantitatively measured using the proportion of common terms of Set 2 relative to all the terms of Corpus T; we refer to this as the overlapping rate of the background Corpus B relative to the target Corpus T. Given any specific target corpus, the bigger the overlapping rate is for a background corpus, the more informative terms are available to be taken into account, and hence the less unknown terms occur in the FSD process.

3.2 Measuring the Distributional Similarity

While measuring the scale of common terms is relatively straightforward, the assessment of distributional similarity is somewhat more involved.

As we focus on the TF-IDF model, the distribution similarity between corpora should be based on the document frequencies. If we order the terms by document frequency for different corpora, each term will likely have a different rank within each corpus. Moreover, if we only look at the ranks of common terms in both corpora (i.e., the terms in Set 2 shown in Fig. 1), it is possible to measure the dissimilarity between two corpora based on their different lists of term ranks.

Before making rank based similarity measurements, some preparation is required. Firstly, the common terms in both corpora (background and target corpus) are extracted as the basis for the comparisons. For each corpus, these common terms are ordered in a descending order based on their document frequencies calculated with only this corpus, and then each term is assigned an index from 1 to n, where n is the number of common terms that are being taken into account. For different corpora, the order of terms will be different, as well as the index of each term. If there are no terms with the same document frequency in an ordered term list, the index of each term can be reasonably considered as its rank in this corpus. However, the fact is that many terms have the same document frequency in a corpus, so they should have the same rank. Instead of assigning different ranks to the neighbouring terms with the same document frequency, we implement some extra operations to make their ranks the same. Specifically, for the terms with the same document frequency, i.e., the terms with indices from i to j, we assign the same average rank $\frac{i+j}{2}$ to all of these, such that this does not affect the rank of any other term. If from the 1^{st} to the 4^{th} terms in the ordered term list have the same document frequency, all of them will be assigned a rank $(1 + 4)/2 = 2.5$.

After pre-processing, we count the number of inversions and calculate the distance between two ordered same-length term lists to present the dissimilarity between these two corpora:

1 **Inversion count** If the order of two different terms in one corpus is not the same as that in the other corpus, e.g., in one corpus, term X has a rank smaller than term Y, while in the other corpus, term X has a rank larger or equal to term Y, we call this situation an inversion. The inversion count metric is defined as the count of all the inversions between two different ordered rank lists.

2 **Manhattan distance** To calculate the dissimilarity between two same-length rank lists we

subtract the rank of each term in one list from the rank of the same term in the other list and sum the absolute value of each of these differences (Kelleher et al., 2015).

As both these dissimilarity metrics show the degree to which a background corpus is different from the target corpus, we expect that the greater the metric the worse the subsequent model is expected to perform on the FSD task. We only evaluate the distributional similarity based on the frequency ranks of the common terms, rather than the quantitative frequency values, because the comparisons based on the quantitative frequency values usually lead to more emphasis on the terms with high frequency values, which should be avoided. It is worth noting that in real use both of these metrics are normalised to between 0 and 1 by being divided by n^2, where n is the length of the rank lists, i.e, the number of common terms. The calculation of these two metrics requires time complexity of $O(n^2)$ and $O(n)$ respectively.

3.3 Comparison between Two Background Corpora Relative to a Target Corpus

With the metrics proposed above, we can make comparisons between different background corpora relative to a target FSD corpus. For the comparison of the scale of common terms, the overlapping rate can be applied to multiple background corpora to rank them based on their rate values. However, the situation for the comparison of the distributional similarity is more involved.

As explained in their definitions, both two dissimilarity metrics proposed above are calculated based on the common terms of a background corpus and a target corpus. If we want to compare among multiple background corpora relative to a target corpus, the calculation should be based on the common terms of all the background and target corpus to ensure the rank list for each background corpus in the same length[1]. The situation of two background corpora and a target corpus is depicted in Fig. 2, in which the calculation of dissimilarity metrics would be based on Common Set. Generally, the common terms shared by the three corpora will be less than those shared by only any two of them. For each background corpus, the terms used for the comparison (i.e., the terms in Common Set) will be less than those used in FSD (i.e., the terms in Common Set and Set 1 for Corpus B1, and the terms in Common Set and Set 2 for Corpus B2). This will lead to errors in the measures and comparisons, and the more background corpora are being compared, the greater the errors will be. In order to limit this kind of error, we restrict to pairwise comparison between background corpora so that the number of terms used for comparisons are relatively large in comparison to the terms used in FSD.

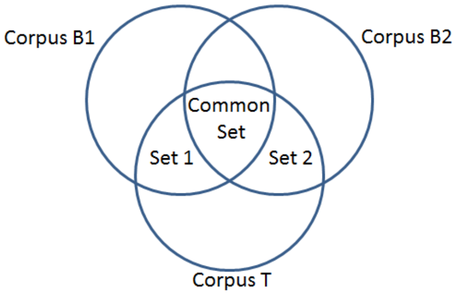


Figure 2: Common Set among two Background Corpora B1 and B2 and a Target Corpus T

4 Experiment Design

In this section, we present our experiments for comparing the scale of common terms and the distributional similarity between different background corpora relative to a target FSD corpus, and apply the models based on different background corpora to the FSD task in an attempt to determine which factor is more predictive of good FSD performance.

4.1 Corpora Used in the Experiments

The target corpus we use for FSD detection is the standard $TDT5$ corpus[2]; the contents of which are newswire stories generated from April to September 2003. The background corpora we are making use of for the current investigation are subsets of $COHA$ (Corpus of Historical American English) (Davies, 2012) and $COCA$ (The Corpus of Contemporary American English) (Davies, 2010). The former covers comprehensive historical English documents from 1810 to 2009 in different domains such as news, fiction, academia

[1] We also tried designing metrics that can be generated based on different terms, i.e., for each background corpus using the terms shared only by the target corpus and itself, rather than common terms shared by all corpora, but we failed because we could not find any valid method to normalise the metrics generated based on different numbers of terms.

[2] https://catalog.ldc.upenn.edu/LDC2006T18

and so on, and the latter is similar to $COHA$ in themes but focuses only on the contemporary contents from 1990 to present. The numbers of documents in $TDT5$, $COHA$ and $COCA$ are about 278,000, 115,000 and 190,000 respectively. As mentioned we make use of subsets of $COHA$ and $COCA$; specifically we mostly include data that predates 2003, i.e., the year of $TDT5$ collection, unless otherwise stated.

In order to answer our underlying research question, i.e., whether bigger or similar background corpora provide the clearer benefit, we carried-out three sets of experiments. In the first set, comparisons are made between $COCA$ and $COHA$ with the assumption that a contemporary corpus will be more similar to the target corpus than a historical one. The second set of experiments supplement the first set and focus on corpus temporality. Comparisons are made between two subsets of the entire COCA corpus - $COCA$ and $COCA\_After\_2003$ that respectively include only the documents before and after 2003, the year when the target corpus was collected. We assume that a corpus with future data is more similar to a target corpus than that with prior data only[3]. The last set of experiments establish comparisons between two subsets of $COCA$ - $COCA\_News$ and $COCA\_Except\_News$, in which $COCA\_News$ contains only the documents in the domain of news, which is the same as the domain of the target TDT5 corpus, while $COCA\_Except\_News$ contains the documents in other domains except news. We also assume that the domain-related corpus is more similar to the target corpus than those in different domains.

4.2 Metric Calculation

In the implementation, we apply all the metrics to each corpus mentioned above, and then make comparisons in each pair of background corpora. In addition, for the comparison of corpus similarity, we examine whether the two proposed metrics, inversion count and Manhattan distance, are consistent with each other in deciding which corpus in each pair is more similar to the target corpus, i.e., whether two metric values for a corpus are both smaller or greater than those for the other corpus in the comparison pair. We also verify whether the results of comparisons correspond with our assumptions about corpus similarity in Sect. 4.1.

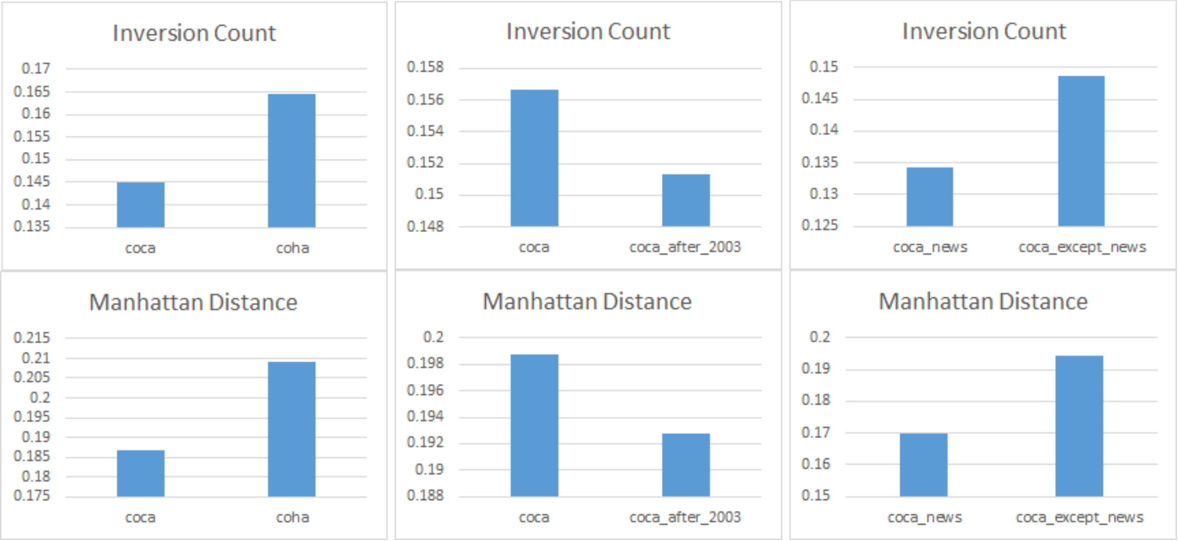
4.3 FSD Evaluation

Following background corpus metric calculation, we build TF-IDF models based on the background corpora being compared and apply these models to the FSD task.

The implementation of FSD is based on the nearest neighbour algorithm with the TF-IDF representations we described in Sect. 2.1. We also adopt the cosine distance as the dissimilarity measure between document representations. In order to reduce the effect of useless terms and different term forms, for both the background and target corpus we remove terms with very high and very low document frequency (stop words and typos), and stem all terms. Aligning with previous research (Yang et al., 1998), comparisons are only implemented with the 2000 most recent stories for each incoming story. The output of each FSD model is a list of novelty scores for each document in the target corpus TDT5. Based on these outputs, the standard evaluation method for FSD is to apply multiple thresholds to sweep through all the novelty scores. For each threshold, a missing rate and a false alarm rate are calculated, and then for all thresholds, the missing and false alarm rates are used to generate a DET (Detection Error Tradeoff) curve (Martin et al., 1997), which shows the trade-off between the false alarm error and the missing error in the detection results. The closer the DET curve is to the zero point, the better the FSD model is said to perform. It happens sometimes for the evaluation with DET curves that many curves are in a tangle – making it is difficult to figure out visually which model performs better. Therefore, we calculate Area Under Curve (AUC) for each FSD model, and the model with the lowest AUC is judged to be best.

In order to achieve more comprehensive results for this evaluation, we implement tests for set variants. Specifically, for each set of experiments, we make comparisons not only between the two background corpora being evaluated, but also between each corpus and the union of both corpora; for example for $COCA$ vs. $COHA$, we not only implement the comparison between $COCA$ and $COHA$, but also between $COCA$ and $COCA + COHA$ and between $COHA$ and $COCA + COHA$, where $COCA + COHA$ is the union of $COCA$ and $COHA$. In this way, we

[3] In real FSD, future data is always unavailable. This set of experiments are only for the use of analysis.



(a) Metric Results for $COCA$ vs. $COHA$

(b) Metric Results for $COCA$ vs. $COCA\_After\_2003$

(c) Metric Results for $COCA\_News$ vs. $COCA\_Except\_News$

Figure 3: Comparisons of Corpus Dissimilarity

have six more comparison results that can be used for the evaluation of the relations between background corpus and model performance for FSD.

5 Results & Analysis

We first look at the comparisons between corpora before looking at FSD performance for different background corpora.

5.1 Results of the Comparisons of Corpus Dissimilarity

We applied the two metrics, inversion count and Manhattan distance, to the three sets of comparisons: $COCA$ vs. $COHA$, $COCA$ vs. $COCA\_After\_2003$ and $COCA\_News$ vs. $COCA\_Except\_News$. The results are shown in Fig. 3. We find firstly that in all comparison sets that the results of the two evaluation metrics are consistent with each other, i.e., the metric values for $COCA$ are both smaller than $COHA$, but greater than $COCA\_After\_2003$, and those for $COCA\_News$ are both smaller than $COCA\_Except\_News$. Secondly, we also find that these comparison results all correspond with our assumptions that more recent domain-related corpora are more similar to the target corpus. Given this, we conclude that both metrics are effective for the comparison of the distributional similarity between background corpora relative to the target corpus, and for the sake of simplicity, we judge Manhattan distance as the most useful metric due to its ease of calculation and interpretation.

5.2 Results of the Relations between Background Corpus and Model Performance for First Story Detection

Results are shown in Table 1, 2 and 3, where the values of the overlapping rates and Manhattan distances are the values for one corresponding background corpus relative to the target corpus. The cells in bold indicate the better results in the comparisons of the scale of common terms and the common term distributional similarity between each pair of background corpora, as well as the better FSD performance. We find that all corpora that are more similar (in terms of term distributions) to the target corpus lead to better performance in FSD, except in the case of very similar performance between $COCA$ and $COCA + COHA$. However, it is worth noting that only six in nine corpora that have a larger scale of common terms correspond with better FSD performance while the other three do not. For example, in Table 3 although the corpus $COCA$ has the much larger scale of common terms, the FSD performance based on it is still worse than that based on $COCA\_News$, because $COCA\_News$ is more similar to the target corpus in terms of language distribution.

Based on these results, it can be argued that term distributional similarity is more predictive of

	coca vs. coha		**coca vs. coca+coha**		**coha vs. coca+coha**	
	coca	coha	coca	coca+coha	coha	coca+coha
Overlapping Rate	**0.3771**	0.3255	0.3771	**0.4193**	0.3255	**0.4193**
Manhattan Distance	**0.1869**	0.2090	**0.1996**	0.2068	0.2076	**0.1949**
AUC	**0.1056**	0.1100	**0.1056**	**0.1056**	0.1100	**0.1056**

Table 1: Comparisons between $COCA$ and $COHA$

	coca vs. coca_after_2003		**coca vs. coca_all**		**coca_after_2003 vs. coca_all**	
	coca	coca_after_2003	coca	coca_all	coca_after_2003	coca_all
Overlapping Rate	0.3771	**0.4077**	0.3771	**0.4583**	0.4077	**0.4583**
Manhattan Distance	0.1987	**0.1928**	0.1996	**0.1950**	**0.1997**	0.2009
AUC	0.1056	**0.1008**	0.1056	**0.1020**	**0.1008**	0.1020

Table 2: Comparisons between $COCA$ and $COCA\_After\_2003$

	coca_news vs. coca_except_news		**coca_news vs. coca**		**coca_except_news vs. coca**	
	coca_news	coca_except_news	coca_news	coca	coca_except_new	coca
Overlapping Rate	0.2932	**0.3184**	0.2932	**0.3771**	0.3184	**0.3771**
Manhattan Distance	**0.1698**	0.1943	**0.1795**	0.1996	0.1986	**0.1880**
AUC	**0.1044**	0.1078	**0.1044**	0.1056	0.1078	**0.1056**

Table 3: Comparisons between $COCA\_News$ and $COCA\_Except\_News$

good FSD performance than the scale of common terms; and, thus we can give general guidance to the selection of background corpus for FSD that a smaller recent domain-related corpus will be more suitable than a very large-scale general corpus for FSD. Of course, our research is directed only at the general situations, as the interpretations do not include extreme situations such as extremely large or small scale of common terms. It is also worth noting that we are purposefully focusing here on the case of a static background corpus and not on the case of updates being made to the TF-IDF model as the FSD process unfolds.

6 Conclusion

We conclude with a highlight of our three main contributions. We proposed a set of metrics for the quantitative evaluation of the scale of common terms and the term distributional similarity of a background corpus relative to a target corpus, and a pairwise comparison scheme between two different background corpora. We also applied the proposed metrics and comparison scheme to the comparisons between background corpora relative to the target FSD corpus, and our results indicate that term distributional similarity is more predictive of good FSD performance than the scale of common terms. Finally, we answered the research question of whether bigger or similar corpus are more useful for FSD by showing that a smaller recent domain-related corpus will be more suitable than a very large-scale general corpus for FSD.

Acknowledgment

The authors wish to acknowledge the support of the ADAPT Research Centre. The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Funds.

References

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report .

James Allan, Victor Lavrenko, and Hubert Jin. 2000a. First story detection in tdt is hard. In *Proceedings of the ninth international conference on Information and knowledge management*. ACM, pages 374–381. https://doi.org/10.1145/354756.354843.

James Allan, Victor Lavrenko, Daniella Malin, and Russell Swan. 2000b. Detections, bounds, and timelines: Umass and tdt-3. In *Proceedings of topic detection and tracking workshop*. sn, pages 167–174.

Thorsten Brants, Francine Chen, and Ayman Farahat. 2003. A system for new event detection. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, pages 330–337. https://doi.org/10.1145/860435.860495.

Mark Davies. 2010. The corpus of contemporary american english as the first reliable monitor corpus of english. *Literary and linguistic computing* 25(4):447–464. https://doi.org/10.1093/llc/fqq018.

Mark Davies. 2012. Expanding horizons in historical linguistics with the 400-million word corpus of historical american english. *Corpora* 7(2):121–157. https://doi.org/10.3366/cor.2012.0024.

Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies* 10(1):1–309. https://doi.org/10.2200/S00762ED1V01Y201703HLT037.

Jeyakumar Kannan, Ar Md Shanavas, and Sridhar Swaminathan. 2018. Real time event detection adopting incremental tf-idf based lsh and event summary generation. *International Journal of Computer Applications* 975:8887. https://doi.org/10.5120/ijca2018916252.

John D Kelleher, Brian Mac Namee, and Aoife D'arcy. 2015. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

Quanzhi Li, Armineh Nourbakhsh, Sameena Shah, and Xiaomo Liu. 2017. Real-time novel event detection from social media. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, pages 1129–1139. https://doi.org/10.1109/ICDE.2017.157.

Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Mark Przybocki. 1997. The det curve in assessment of detection task performance. Technical report, National Inst of Standards and Technology Gaithersburg MD.

Sean Moran, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2016. Enhancing first story detection using word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, pages 821–824. https://doi.org/10.1145/2911451.2914719.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2010. Streaming first story detection with application to twitter. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*. Association for Computational Linguistics, pages 181–189.

Saša Petrović, Miles Osborne, and Victor Lavrenko. 2012. Using paraphrases for improving first story detection in news and twitter. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*. Association for Computational Linguistics, pages 338–346.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13(7):1443–1471. https://doi.org/10.1162/089976601750264965.

Fei Wang, Hector-Hugo Franco-Penya, John D Kelleher, John Pugh, and Robert Ross. 2017. An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, pages 291–305. https://doi.org/10.1007/978-3-319-62416-7_21.

Fei Wang, Robert J Ross, and John D Kelleher. 2018. Exploring online novelty detection using first story detection models. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, pages 107–116. https://doi.org/10.1007/978-3-030-03493-1_12.

Dominik Wurzer, Victor Lavrenko, and Miles Osborne. 2015. Twitter-scale new event detection via k-term hashing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2584–2589. https://doi.org/10.18653/v1/D15-1310.

Yiming Yang, Tom Pierce, and Jaime G Carbonell. 1998. A study on retrospective and on-line event detection https://doi.org/10.1145/290941.290953.

Predicting Sentiment of Polish Language Short Texts

Aleksander Wawer
Institute of Computer Science
Polish Academy of Sciences
Jana Kazimierza 5
01-248 Warszawa, Poland
axw@ipipan.waw.pl

Julita Sobiczewska
University of Warsaw
Krakowskie Przedmieście 26/28
00-927 Warszawa, Poland
j.sobiczewska96@gmail.com

Abstract

The goal of this paper is to use all available Polish language data sets to seek the best possible performance in supervised sentiment analysis of short texts. We use text collections with labeled sentiment such as tweets, movie reviews and a sentiment treebank, in three comparison modes. In the first, we examine the performance of models trained and tested on the same text collection using standard cross-validation (in-domain). In the second we train models on all available data except the given test collection, which we use for testing (one vs rest cross-domain). In the third, we train a model on one data set and apply it to another one (one vs one cross-domain). We compare wide range of methods including machine learning on bag-of-words representation, bidirectional recurrent neural networks as well as the most recent pre-trained architectures ELMO and BERT. We formulate conclusions as to cross-domain and in-domain performance of each method. Unsurprisingly, BERT turned out to be a strong performer, especially in the cross-domain setting. What is surprising however, is solid performance of the relatively simple multinomial Naive Bayes classifier, which performed equally well as BERT on several data sets.

1 Introduction

Automated sentiment analysis usually involves training machine learning or deep learning models in supervised fashion. Typically, studies involve one data type and report high accuracy. For example, the seminal machine learning studies on IMDB movie reviews of (Pang et al., 2002) indicated accuracy over 80%. Recently, the accuracy of a deep learning system reported on this data set exceeded 95% (Howard and Ruder, 2018). Similarly, sentence-level sentiment predictions exceeded 95% accuracy on binary version of the popular Stanford Sentiment Treebank (abbreviated as SST-2) (Liu et al., 2019).

However, multiple studies indicate that the models trained on such data sets are in fact very far from being applicable universally. Machine and deep learning models tend to fit to specific type of texts and language. When applied to different language types in terms of both style and vocabulary, the performance drops sharply. This issue is often explored under the name of domain dependency or data set dependency. Several methods have been proposed to address it, such as for example (Selmer et al., 2013). In another stream of related studies, authors considered the task of cross-domain sentiment analysis: adaptation of a model to target domain or text type that is different from the domain or text type that the model was trained on. Examples include (Peng et al., 2018).

The goal of this article is to evaluate and compare supervised techniques of sentiment analysis of short texts using all available resources in the Polish language. We utilize three generations of machine learning:

- machine learning models using bag-of-words vector representations, algorithms such as Naive Bayes and Support Vector Machines,
- recurrent deep neural networks based on LSTM with and without pre-trained word embeddings,
- finally the most recent generation of deep neural networks, pre-trained on language modeling tasks (BERT and ELMO).

Proceedings of Recent Advances in Natural Language Processing, pages 1321–1327,
Varna, Bulgaria, Sep 2–4 2019.

The questions that our paper addresses are: (1) to what extent is each of the method usable for training a universal sentiment analysis model (how well it performs in cross-domain setting) and (2) how good it is for predicting sentiment within the same text type it has been trained on (how well it performs in in-domain setting).

The motivation for a universal classifier might not be obvious at first, as clearly the best performance is achieved by in-domain classification (Twitter may need tweet classifier, Facebook needs posts classifier, IMDB needs review classifier, and so on). However, universal character of sentiment classification models allows for better performance in cases when source data distribution is different than target data distribution. This happens on every day basis, for instance Twitter posts change their topic over time and optimum performance requires new train sets to match the newest data. These issues are well-known to machine learning community and explored under topics of domain adaptation, dataset shift and semantic drift. Ability to apply the model universally to various data types is a great advantage from the practical point of view.

Our intention is to focus on supervised learning. Specifically, this means working on collections of short texts such as single sentences, tweets or short reviews with labeled sentiments. The goal of our task is to classify sentiment of the whole written utterance (as for example, a sentence, review or tweet) into three classes: as positive, neutral or negative.

When computing sentiment we rely only only on learning from provided text examples and do not use any resource which might help in sentiment predictions such as a sentiment dictionary.

The paper is organized as follows: Section 2 contains a description of sentiment datasets, Section 3 describes the methods used for sentiment prediction. Section 4 describes the results obtained in our experiments. Finally, Section 5 contains conclusions and a discussion of possible future work.

2 Data Sets

The experiments described in this paper are based on several resources with manually labeled sentiment. The texts in our experiments are "short": tweets do not exceed 140 characters, Treebank sentences are arbitrarily long in terms of tokens but syntactically and semantically correct, reviews contain from 1 to 3 sentences.

2.1 Polish Sentiment Treebank (TW)

The first resource is Polish language dependency treebank with sentiment annotations ("Treebank Wydźwięku" abbreviated as TW). It is available to download [1]. Similar to Stanford's Sentiment Treebank (SST) (Socher et al., 2013), Treebank Wydźwięku (TW) was intended to study compositional phenomena in sentiment analysis. There are several notable differences between the two:

- Dependency trees (TW) instead of constituency binary parse tress (SST),
- 3-class sentiment (TW) instead of 5-class (SST),
- Open-domain (TW) instead of one domain of movie reviews (SST).

In TW, overall sentiment of each sentence corresponds to the sentiment of its root. Sentences in this data set often contain mixed sentiment, even opposite polarities: one part may be positive and the other negative. This makes the task of predicting overall sentiment more difficult.

2.1.1 TW Version 1.0

Initial version of the TW treebank contained sentences from Składnica Treebank[2] (part referred to as **sklad**) and sentences from two types of product reviews: perfumes and clothes (part called **rev**).

The first release of the treebank was published as a part of Task 2: Sentiment analysis in PolEval 2017 campaign on evaluation of natural language processing tools for Polish. In this competition, submitted tools competed against one another within certain tasks selected by organisers, using provided data. Solutions were evaluated according to common procedures.

The intended use of the treebank was as follows: given a set of syntactic dependency trees, the goal was to provide the correct sentiment for each subtree (phrase). Phrases correspond to sub-trees of dependency parse tree. Annotations assign sentiment values to whole phrases (and in some cases, sentences), regardless of their type. The PolEval

[1] http://zil.ipipan.waw.pl/TreebankWydzwieku

[2] http://zil.ipipan.waw.pl/Sk%C5%82adnica

part of the treebank and related evaluation script may be freely downloaded[3].

Three systems participated in the tasks, all of them based on TreeLSTM (Tai et al., 2015). The description of systems and evaluation methodology can be found in (Wawer and Ogrodniczuk, 2017). Due to different nature of these tasks (computing sentiment of each sub-phrase and each sentence vs sentiment of sentences only) the results of these systems are not directly comparable to results reported in our paper.

2.1.2 TW Version 2.0

In August 2018, a new batch of sentences has been added to TW. It contains following new parts:

- Test (evaluation) sentences from PolEval 2017 sentiment task (part called **polevaltest**),
- 2 x 500 sentences collected from various web sources, mostly difficult, mixed sentiments and negative ones (parts called **neg** and **jun18**).

To our best knowledge, our paper is the first to describe and use the new version of this resource.

2.2 Twitter Data

This data set contains one thousand Polish language tweets, gathered and manually labeled as to their sentiment during the TrendMiner project[4]. Many tweets are related to publicly discussed events, some of them originate from politicians, journalists and public figures. Many of them are tweets of simple Twitter users, including teenagers. Overall, the dataset appears to be a fairly representative sample of communication occurring on Twitter in the Polish language. Due to Twitter's policy it can not be made publicly downloadable.

2.3 Movie Reviews

This data set consists of one thousand manually collected movie reviews from Polish website: http://www.filmweb.pl. Most of them are very short texts (1-2 sentences), the rest is a collection of reviews containing up to 5 sentences. Each review has a corresponding numeric score (number of stars) from 1 to 10, assigned by the review's author. Each category (number of stars) has exactly one hundred reviews.

All scores were re-scaled into three categories: stars from 1 to 3 were re-scaled into negative category (-1), stars from 4 to 6 into neutral (0), stars 7 to 10 into positive (1).

What should be noted, however, is that the reviews very often contain spelling mistakes, words with omitted Polish diactric marks, often contain slang or sarcasm. All that makes them problematic for automated analysis.

2.4 Label Frequencies

Most of the datasets are not balanced in terms of sentiment label distributions. Twitter data set contains mostly neutral texts (tweets). Filmweb's balance is nearly perfect, as only the positive class has slight advantage in terms of frequency. Also TW's balance is far from perfect distribution between three sentiment classes as most sentences are neutral. Some pieces of TW treebank have been deliberately created to address imbalance issues in the sentiment treebank, such as for example part of the data called neg in TW 2.0, which contains many negative sentences.

Table 1 presents sentiment label distribution in each of the data set.

Table 1: Distribution of sentiment classes in each data set

file	-1	0	1	all
jun18-TW	59	240	202	501
neg-TW	252	245	3	500
polevaltest-TW	40	215	95	350
rev-TW.	7	868	90	965
sklad-TW	3	230	2	235
twitter	80	854	66	1000
filmweb	300	300	400	1000

3 Machine and Deep Learning Methods

3.1 Bag-of-Words and Machine Learning

Machine learning methods described in this subsection used bag-of-words text representations. We converted text to word vectors using word-level unigram vectorizer with TF-IDF weights.

We have focused on two machine learning algorithms: Naive Bayes and Support Vector Machines.

[3] http://2017.poleval.pl/index.php/tasks/

[4] https://cordis.europa.eu/project/rcn/100752/factsheet/en

Multinomial Naive Bayes (NB) Classifier is a well-known supervised machine-learning algorithm with an assumption of independence among predictors.

Support Vector Machine (SVM) is also a well-known supervised machine learning algorithm. We used linear kernels and implementation from the liblinear library[5].

3.2 LSTM Neural Network (NN)

We used two approaches to implement the first layer of the neural network.

In the first approach we used untrained, random initialized embedding layer that uses 32 length vectors to represent each word. This method is marked as NN in the results.

In the second approach we changed this layer to pre-trained word2vec 100-dimensional word embeddings for Polish[6]. The embeddings were generated using gensim package (Řehůřek and Sojka, 2010) with skip-gram model architecture for two large text corpora: The National Corpus of Polish[7] and Polish edition of Wikipedia. We marked this approach as NN+E subsequently.

The structure of the neural network was as follows: word embedding layer, LSTM layer with 100 memory units, two Dense layers: the first with 100 neurons, the second with 3 output values, one for each class, with dropout regularization between them. As an output layer we used the softmax activation function.

3.3 ELMO

In one of the approaches we used ELMo method (Peters et al., 2018) to represent texts. ELMo, as it's authors put it, is a deep contextualized word representation that models both (1) characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are extracted from states of a deep bidirectional language model, which is pre-trained on a large text corpus. We used the ELMO implementation and models described in (Che et al., 2018). The model was trained on Polish language Wikipedia.

ELMO, although currently often superseded by other alternatives, contributed to state-of-the-art results in multiple natural language processing tasks such as question answering or paraphrase detection.

To obtain ELMO representation of each text, we computed average vector from 3 neural network layers, which resulted in a vector of 512 numbers. In the second step, these vectors were used to classify sentiment of an input text. Here, we experimented with multiple well-known machine learning methods such as Logistic Regression (LR), Random Forest (RF) with 200 trees and Support Vector Machine classifier (SVC) with a linear kernel. Each of these variants is subsequently marked as ELMO-LR, ELMO-RF and ELMO-SVC.

3.4 BERT

BERT is an example of the newest generation of pre-trained neural networks based on transformer architecture (Devlin et al., 2018). BERT acronym stands for Bidirectional Encoder Representations from Transformers, also obtained state-of-the-art results on multiple NLP tasks such as natural language inference and question answering. BERT model comes pre-trained on a very large corpus of unlabeled data, can be subsequently fine-tuned to a task with a limited amount of data such as sentiment analysis.

In our experiments we used smaller version of BERT. It contains 110M parameters and has support for 104 languages, 12 layers, the size of each hidden layer is 768, 12 self-attention heads (bert-base-multilingual-cased). We set maximum sequence length parameter to 128, which is enough to cover several sentences, and tested 3 and 4 training epochs. We tested BERT in a scenario which adds a sequence classification head on top. BERT Transformer is pre-trained, the sequence classification head is only initialized and has to be trained.

4 Results

This section illustrates the results of three experimental modes tested in our paper.

4.1 In-Domain

In the first mode (in-domain), we examine the performance of models trained and tested on the same text collection using standard cross-validation. On one hand, the performance of in-domain is driven up by lexical and structural similarity: training data are likely to be similar to test data in terms of vocabulary and syntactic structures. On the other,

[5] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

[6] http://dsmodels.nlp.ipipan.waw.pl/dsmodels

[7] http://nkjp.pl

	Method							
data set	NB	SVC	NN	NN+E	ELMO+LR	ELMO+RF	ELMO+SVC	BERT
jun18-TW	44.3%	44.3%	45.9%	45.9%	40.9%	44.7%	43.5%	**47.9%**
neg-TW	42%	42%	46%	44.8%	47.8%	46.8%	46.8%	**50%**
polevaltest-TW	55.4%	54.2%	**61.1%**	**61.1%**	49.1%	**61.1%**	46.9%	53.4%
rev-TW	88.2%	92.2%	92%	92%	**93%**	92%	90.8%	89.9%
sklad-TW	**100%**	**100%**	96.7%	96.7%	97.9%	97.9%	98%	97.8%
all-TW	**71.5%**	67.9%	71.2%	71.2%	67.3%	70.9%	63.2%	70.4%
twitter	84.6%	**85.4%**	84%	84%	75.5%	80.2%	70.5%	85.3%
filmweb	63.2%	**69.6%**	40%	40%	58.7%	55.2%	55.3%	40%

Table 2: In-domain average accuracy in 5-fold cross-validation

	Method							
data set	NB	SVC	NN	NN+E	ELMO+LR	ELMO+RF	ELMO+SVC	BERT
jun18-TW	**47.9%**	44.9%	42.7%	**47.9%**	43.9%	47.5%	43.5%	**47.9%**
neg-TW	**49%**	41.4%	40.8%	41.6%	40.6%	48%	37.8%	**49%**
polevaltest-TW	**61.4%**	57.4%	54.3%	58.9%	57.1%	60.9%	57.1%	**61.4%**
rev-TW	89.7%	76.7%	69%	82.5%	69.1%	88.4%	65%	**89.9%**
sklad-TW	**97.9%**	83.8%	73.2%	95.7%	74%	**97.9%**	66.4%	97.8%
twitter	85.3%	78.3%	64.1%	84.5%	60.1%	79.8%	52%	**85.4%**
filmweb	30%	37.7%	34.3%	30.2%	36.9%	30%	**39.1%**	30%

Table 3: One vs rest cross-domain accuracy

models do not utilize information contained in other available data sets.

The results of this mode are presented in Table 2. The best results were achieved by BERT (two treebank datasets: jun18 and neg) and SVC classifier (treebank dataset sklad, twitter and filmweb). Except for filmweb data set with over 30% discrepancy between the best and worst methods, the differences between methods were not large, usually did not exceed several percents.

On the whole TW sentiment treebank (marked as all-TW) the surprising winer was Naive Bayes, that managed to outperform other methods including BERT by a small margin.

4.2 One vs Rest Cross-Domain

In the second mode (one vs rest cross-domain) we train models on all available data except the given test collection, which we use as a test set. In this mode, models did not have a chance to learn from data similar to test data (maybe except some parts of TW treebank which may be considered similar). However, can utilize and possibly benefit from information contained in all other data sets available. Table 3 presents the results of one vs rest cross-domain experiments. In this mode, the best performers were BERT (the best results for 5 out 7 data sets) and surprisingly, Naive Bayes algorithm (best performance for 4 out of 7). Recurrent neural networks (NN) did not manage to reach state-of-the-art results, however it is easy to notice strong positive influence of pre-trained word2vec word embeddings (NN+E), with accuracy gains from several to as much as twenty percentage points in the case of twitter. One can also note that the performance of ELMO with Random Forest (ELMO+RF) is significantly better than the two other ELMO variants we tested.

4.3 One vs One Cross-Domain

	all-TW	tweets	filmweb
all-TW	-	70.4%	15.3%
tweets	85.4%	-	30.6%
filmweb	30%	30%	-

Table 4: One vs one cross-domain accuracy of BERT method

	all-TW	tweets	filmweb
all-TW	-	67.9%	32.1%
tweets	79.1%	-	37%
filmweb	38.9%	30.5%	-

Table 5: One vs one cross-domain accuracy of NB method

In the third mode (one vs one cross-domain), we train a model on one data set and apply it to another one, repeating for each combination. In this mode we tested only the highest performing methods, such as BERT and Multinomial Naive Bayes. In this experiment we merged all sub-parts of the sentiment treebank (jun18, neg, polevaltest, rev, sklad) into one data set presented as TW. It consists of 2500 sentences.

Table 4 contains one vs one results for BERT classifier. Rows refer to test data sets and columns to train data sets. As we can see, the accuracy of 85.4% on tweets with models trained on TW is high and identical to SVC of in-domain 5-fold cross-validation and also identical to BERT in one vs rest mode. The performance on filmweb indicates that the models did not start to learn effectively. Training on filmweb did not help the performance on TW treebank (only 15.3%). Since filmweb dataset is reasonable in size and balanced, we can only hypothesize that the language is too different.

Table 5 contains one vs one results for Naive Bayes classifier. As before, rows refer to test data sets and columns to train data sets. In some cases Naive Bayes outperformed BERT. It was the case with training on filmweb movie reviews, models trained on this data set performed better than BERT both on tweets and on sentiment treebank TW. Also in the scenario of training on TW and testing on filmweb, Naive Bayes turned out to be better.

5 Conclusions and Future Work

The main point of this paper was to use all available Polish language data sets to seek the best possible performance in supervised sentiment analysis of short texts. We compared three generations of methods: machine learning with bag-of-words representation, recurrent neural networks (with and without pre-trained word embeddings) and finally deep neural networks pre-trained on language modeling task, including the newest transformer architecture BERT.

In sentiment classification, data available at training time is often different from data we intend to analyze in production environments. Ideally, classifiers should be capable of predicting sentiment on multiple types of data, covering various topics and texts of varied length without the need of re-training. In practice, achieving the best possible performance requires training or re-training on data very similar to those we intend to analyze.

To explore the limits of this approach we experimented with cross-domain setting in which we train the model on one text type and apply it to another text type (one vs one cross-domain). We confirmed that this setting poses a problem often leading to substantial performance degradation.

Using several sentiment-labeled data sets as training data may in theory improve classifier's accuracy and robustness. In our paper we investigated possible benefits from training models on data less similar to the test set (cross-domain one-vs-rest mode) and compared this to model training on smaller amounts of highly similar data (in-domain, models trained on the same type of data).

We found that for some data sets (TW treebank sub-sets, twitter) the results are comparable for cross-domain and in-domain setting, while for movie reviews in-domain setting turned out to be almost 30% better. Here, similar training data played a more significant role than using other less similar data sets to learn from. The best-performing in-domain method turned out to be somewhat old SVC with simple bag-of-words representation.

BERT, transformer-based neural network with pre-training, turned out to benefit from large amounts of less similar data, with top performance in one-vs-rest cross-domain setting. Interestingly, multinomial Naive Bayes method turned out to perform on a very similar level with far less model parameters, which may be a viable alternative in more speed oriented environments without GPU processors.

Some of the issues raised in this paper are worth pursuing in further work. The first problem is the amount of pre-training and architecture changes needed to reach acceptable cross-domain performance. Apparently, this problem has still not been solved and more efforts are needed to reach high accuracy.

The second problem worth investigating is the matter of how suitable are sentiment treebanks, designed for experiments on within-sentence compositional sentiment phenomena (for example, studying sentiment propagation in sentence structure or mixed sentiments) for predicting single label of sentence-level sentiment. As a part of this future study, we intend to compare the performance of Tree-LSTM methods such as those reported in PolEval 2017 (Wawer and Ogrodniczuk, 2017) on sentence-level sentiment with methods reported in our work.

References

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*. Association for Computational Linguistics, Brussels, Belgium, pages 55–64. http://www.aclweb.org/anthology/K18-2005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805. http://arxiv.org/abs/1810.04805.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 328–339. https://www.aclweb.org/anthology/P18-1031.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *CoRR* abs/1901.11504. http://arxiv.org/abs/1901.11504.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 79–86.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuanjing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, pages 2505–2513. https://www.aclweb.org/anthology/P18-1233.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. `http://is.muni.cz/publication/884893/en`.

Øyvind Selmer, Mikael Brevik, Björn Gambäck, and Lars Bungum. 2013. NTNU: Domain semi-independent short message sentiment classification. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 430–437. https://www.aclweb.org/anthology/S13-2071.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1631–1642. http://aclweb.org/anthology/D13-1170.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1556–1566. https://doi.org/10.3115/v1/P15-1150.

Aleksander Wawer and Maciej Ogrodniczuk. 2017. Results of the PolEval 2017 competition: Sentiment Analysis shared task. In Zygmunt Vetulani and Patrick Paroubek, editors, *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu, Poznań, Poland, pages 406–409.

Improving Named Entity Linking Corpora Quality

Albert Weichselbraun[1], **Adrian M.P. Braşoveanu**[2], **Philipp Kuntschik**[1], and **Lyndon J.B. Nixon**[2]

[1]Swiss Institute for Information Science, University of Applied Sciences, Chur, Switzerland
{*albert.weichselbraun,philipp.kuntschik*}*@htwchur.ch*
[2]MODUL Technology GmbH, Vienna, Austria
{*adrian.brasoveanu,lyndon.nixon*}*@modul.ac.at*

Abstract

Gold standard corpora and competitive evaluations play a key role in benchmarking named entity linking (NEL) performance and driving the development of more sophisticated NEL systems.

The quality of the used corpora and the used evaluation metrics are crucial in this process. We, therefore, assess the quality of three popular evaluation corpora, identifying four major issues which affect these gold standards: (i) the use of different annotation styles, (ii) incorrect and missing annotations, (iii) Knowledge Base evolution, (iv) and differences in annotating co-occurrences. This paper addresses these issues by formalizing NEL annotations and corpus versioning which allows standardizing corpus creation, supports corpus evolution, and paves the way for the use of lenses to automatically transform between different corpus configurations. In addition, the use of clearly defined scoring rules and evaluation metrics ensures a better comparability of evaluation results.

1 Introduction

Named Entity Linking (NEL) systems identify mentions of named entities and link them to Knowledge Bases (KBs) (Ji et al., 2017). Their evaluation heavily depends upon annotated gold standards and competitions such as TAC-KBP (Ji et al., 2017) or Open Knowledge Extraction (Nuzzolese et al., 2016) which help drive research and advance the state of the art. Knowledge Bases, annotation guidelines and gold standards, NEL tools, as well as the evaluation systems themselves, were found to introduce errors into NEL evaluations (Braşoveanu et al., 2018). The most critical issues are related to corpora quality due to wrong, partial or insufficient annotations (van Erp et al., 2016; Jha et al., 2017). Annotation guidelines used to produce a gold standard come with different rules for describing whether the annotation systems should take into account overlapping mentions, co-references or general concepts mentioned in a KB (Rosales-Méndez et al., 2018). These guidelines are domain-specific and often depend on the application and task-specific context. Semantic search, for instance, would not require co-references, but they are relevant for relation extraction tasks (Rosales-Méndez et al., 2018).

The mentioned issues can be approached from various angles. General improvements such as a clear definition for NEL will affect multiple components of a NEL system. More specific suggestions, such as KB refinements typically affect only corpora that link entities to that particular KB. In all these cases the objective should be improving the quality and transparency of both the corpus and the evaluation processes.

A serious issue with current NEL evaluations is the lack of flexibility during the evaluation process which forces NEL systems to adapt to the used evaluation corpora. For example, if a KB has more name variants (e.g., Bobby Kennedy and RFK for Robert F. Kennedy) than the corpus annotators have considered, NEL systems able to correctly detect these name variants will be penalized since they do not occur in the corpus and are, therefore, considered errors. This paper, therefore, suggests solutions such as lenses (Section 3) and corpus versioning (Section 4.2) to address this issue. In particular, our main contribution is describing the innovations that can be applied at a corpus and evaluation systems level in order to create more flexible and expressive evaluations.

Proceedings of Recent Advances in Natural Language Processing, pages 1328–1337,
Varna, Bulgaria, Sep 2–4 2019.

The rest of the paper is organized as follows: Section 2 describes related work; Section 3 collects empirical evidence for common issues with NEL corpora and provides a concise problem description; Section 4 presents our approach towards building flexible evaluation systems; whereas the final section discusses the conclusions.

2 Related Work

An early analysis on the effect entity overlap between different data sets, confusability (the number of meanings a surface form can take) and dominance in several data sets was performed in (van Erp et al., 2016). Most current gold standards are known to suffer from *annotation mistakes*, *lack of updates* (typically due to the fact that there are no clear updating guidelines or funds for this operation), *popularity bias* (tools return most popular candidates), *small volume* (only several hundred examples), and are typically focused on a small set of languages (Ngomo et al., 2018).

Early methods to improve the quality and speed of human annotations have included: dynamic sentence selection in combination with iterative pre-annotation and qualitative checks (Tsuruoka et al., 2008) or crowdsourcing (Sabou et al., 2014); whereas more recent techniques include hypergraphs in order to highlight various name variants (Katiyar and Cardie, 2018). Several recent linguistic-driven techniques for improving gold standard quality discussed in (Sakor et al., 2018) include effect of capitalization, of implicit/explicit entities, of number of words or hidden relations. Automatically generated benchmarks like those provided with BENGAL (Ngomo et al., 2018) can help improve annotation speed, but only if deployed together with advanced debugging and error analysis tools, as otherwise there will be a risk of increasing bias.

A set of KB improvements can also be devised to aid domain experts and NEL systems in identifying mentions of named entities. KBs, for example, can be used to expand upon the number of name variants from a corpus by including multiple annotations for each entity to cover cases of embeddings, overlap and extensions (Odoni et al., 2018). At the named entity disambiguation level, such techniques include collecting all name variants from multiple Knowledge Graphs (Ehrmann et al., 2016) or using hypergraphs and multi-layer bi-LSTMS to detect the nested entities (Katiyar and Cardie, 2018).

The lack of parallel language corpora like Europarl (Koehn, 2005) for NEL (except for some smaller corpora like MeanTime (Minard et al., 2016)) is another serious issue that is rarely discussed, though such corpora are available for NER (Agerri et al., 2018).

3 Lenses and Corpus Quality

Today's NEL evaluation tools automatically compare the performance of multiple NEL systems on various data sets using a wide variety of experiments (e.g., NER - Named Entity Recognition, Entity Typing, D2KB - matching entity mentions to certain KB, etc) and indicators (e.g., precision, recall, F1, accuracy, etc). GERBIL (Röder et al., 2018) and its extensions (Waitelonis et al., 2019) were designed to support multiple experiment types using black box evaluation techniques. The neleval system (Hachey et al., 2014) based on the TAC-KBP guidelines provides primary error explanations. As an alternative, visual evaluation systems such as Orbis (Odoni et al., 2018) and VEX (Heinzerling and Strube, 2015), also allow close inspection of the evaluation results and help designers improve system performance. Upon analyzing these packages, we came to the conclusion that none of them provides suitable tools for handling multiple annotation styles and updating gold standards. Therefore, reuse of old gold standards can lead to problematic results (e.g., entities declared NIL in the gold can currently exist in the current KB version and can be retrieved by annotator tools) or even unfair evaluations (e.g., tools that use an old KB should not be compared with those who use the latest updates).

We think NEL systems should be evaluated against both updated gold standards and KBs, therefore we suggest the application of lenses over the existing data, i.e. transformations between different KB versions (e.g., DBpedia 2015-10 and DBpedia 2017-10), KBs (e.g., DBpedia and Wikidata) and annotation styles (e.g., always take the longest strings or only annotate non-NIL entities). The following section presents an analysis of three popular NEL corpora, discusses different use case for lenses and the corresponding transformation rules that are required to transfer corpora from one representation to the other.

3.1 Corpus Quality

Creating high quality gold standard data is a difficult task due to: (i) *incomplete annotation rules* - i.e. cases that have not been properly covered in the annotation rules; (ii) *errors* - present on multiple levels, from human or automated annotation errors, to process errors (e.g., errors in the annotation guidelines) or KB errors; as well as (iii) *decay* - KBs used for the annotation might become obsolete or outdated, forcing the corpus maintainer to consider KB evolution or even KB migration.

In order to understand the extent to which a corpus may require updating, we have analyzed several well-known corpora from our field: (i) *OKE2016 Task 1* (Nuzzolese et al., 2016) - focuses on short biographical sentences extracted from Wikipedia; (ii) *Reuters128* (Röder et al., 2014) - a set of texts extracted from the classic Reuters corpora; and (iii) the English partition of the *MeanTime* (Minard et al., 2016) corpus that covers events extraction.

Table 1 illustrates several attributes related to the number of entities in a corpus that directly reflect *incomplete annotation rules*, *errors* and *decay*: (i) the number of original annotations; (ii) the number of NIL annotations at publication date; (iii) updated NIL counts which indicate the impact of KB evolution; (iv) count of potential overlaps within the original annotations that need to be handled according to the used annotation guidelines; (v) potential missing entities based on the annotation the guideline (if such a document exists).

Empty columns (marked with -) were not filled due to lack of available data or guideline. Counts from rows (i) and (ii) were taken directly from the corpora; row (iii) count was estimated based on SPARQL queries that aim at linking NIL entities to the KB; and counts for columns (iv) and (v) were estimated based on annotating samples from each data set.

Two annotators have independently annotated a quarter of the documents that have been selected using random sampling. Only the entities mentioned in the article or guideline related to a data set were used for the respective counts (e.g., Reuters128 was only annotated with Person (PER), Organization (ORG) and Location (LOC) based on (Röder et al., 2014)). The analysis only considers NIL entities that were marked as such (e.g., NIL or similar designation). Consequently, the OKE 2016 counts shows no NIL entities, since they were not included in the original dataset. The updated NIL counts are obtained by using SPARQL queries that determine whether NIL entities have become available in new KB versions.

Table 2 examples were extracted from the Reuters128 corpus and illustrate (i) missing and wrong groundings, (ii) KB evolution, and (iii) surface forms deviations due to different annotation rules.

Overlaps tend to appear in cases related to LOC and ORG entities. Quite often, an overlap is identified in long names such as *Chattanooga State Technical College* or *City University of New York Graduate Center*. In similar cases a surface form expansion that will contain the longest possible string should correctly match the entity from the gold standard. Complicated cases like the following: *Loyola's University in Belgium, Economics* (from OKE2016) can be interpreted in multiple ways. This example can either be rendered as (i) one long entity that corresponds to the whole string; (ii) one entity that describes the University (*Loyola's University in Belgium*); (iii) two entities (*Loyola's University* and Belgium); (iv) two entities again (*Loyola's University in Belgium* and a string *Economics*); even (v) three entities (*Loyola's University* and *Belgium* and *Economics*). The phrasing of the examined sentence suggests that the fourth version is the correct one. Without a thorough text analysis such instances are extremely difficult to disambiguate for both humans and machines.

Even if we leave aside difficult cases that typically show a low inter-rater agreement such as *Potential Overlap* (Table 1, row iv; e.g., agreement of 0.40 for Reuters128) or *confusability* (van Erp et al., 2016), there are still many mentions that are not spotted in the original corpus such as those from the *Missing Entities Guideline* (Table 1, row v) for which experts also exhibit better inter-rater agreements (e.g., 0.61 for Reuters128).

These findings suggest that the methods and processes used for annotating documents need to be updated. Applying lenses would be one of the methods that could address some of the mentioned shortcomings, as they would help both accounting for multiple points of views when annotating, as well as for KB evolution. Such lenses coupled with well-defined metrics for measuring NEL performance are key towards reliably assessing a system's performance and driving its development.

No	Rule	OKE2016	Reuters128	MeanTime
i	Original Annotation ALL	176	880	853
ii	Original NIL Only	-	230	554
iii	Updated NIL Count	-	175	465
iv	Potential Overlap	84	104	221
v	Missing Entities Guideline	76	180	272

Table 1: Estimated entity counts based on different criteria in three corpora. All data sets are in English.

surface	gold link	correct link	error
[Volkswagen AG] **[VOWG.F]**, [VW], is due ...	NIL	dbr:Volkswagen	Missing Annotation
bid for **[Avondale Mills]** ...	NIL	dbr:Avondale_Mills	KB evolution
[The Chicago Mercantile Exchange], **[CME]**, said ...	dbr:CME_Group	dbr:Chicago _Mercantile_Exchange	Incorrect Link
... of **[Salem**, Ore.]	dbr:Salem,_Oregon	dbr:Salem,_Oregon	Different surface form

Table 2: Examples of dataset errors. Gold entity spans are marked by parentheses. Errors are presented in bold. Abbreviations or tickers should be separate entities. Locations can include states.

3.2 Context-Specific and Application-Specific Transformation Rules for Lenses

A named entity linking system links a mention $m_{[s_i]}^{e_i,KB}$ or $m_{[x_i,y_i]}^{e_i,KB}$ of a named entity with surface form s_i within a document d to the corresponding entity e_i in a knowledge base KB. The variable x_i indicates the mention's start position within the document and y_i the corresponding end position.

Mentions may overlap and the specification of the knowledge base KB can be omitted, if it is not relevant for the application (e.g. if we do not consider different KB versions in the given use case).

The system distinguishes between

1. $m_{[s_i]}^{e_i,KB}$ surface forms s_i that were linked to an entity e_i within a knowledge base KB,
2. $m_{[s_i]}^{nil}$ mentions of Named Entities (NEs) that are not available in the KB and, therefore, are not linked (i.e. NIL entities), and
3. $m_{[s_i]}^{\emptyset}$ candidate mentions with surface form s_i that do not refer to a named entity.

3.2.1 Different Annotation Styles

Annotation styles specify rules that aid annotators in assessing if (i) a candidate mention should be considered a mention of a named entity, and (ii) the extent of the corresponding surface form.

Although a trivial design decision for isolated mentions, the consistent handling of nested mentions requires more thought. For instance, the text snippet *University of Western Australia Cricket Club* may contain, dependent on the applied annotation rule, up to four overlapping mentions (*Australia*, *Western Australia*, *University of Western Australia*, *University of Western Australia Cricket Club*). In addition, annotation styles might be entity type specific even within a single corpus.

We consider the following three annotation styles, as illustrated based on the annotation of the text snippet *Vienna, VA*:

1. *∅MIN* disregards overlapping entities and tries to extract the minimum number of entities: $m_{\text{[Vienna, VA]}}^{dbr:Vienna,\_Virginia}$, i.e. links the snippet to the *Vienna, Virginia* DBpedia entity.
2. The annotation style *∅MAX*, in contrast, extracts the maximum number of entities from a given text snippet: $m_{\text{[Vienna]}}^{dbr:Vienna,\_Virginia}$, $m_{\text{[VA]}}^{dbr:Virginia}$
3. The style *OMAX* allows for overlaps and, again, extracts the maximum number of entities: $m_{\text{[Vienna, VA]}}^{dbr:Vienna,\_Virginia}$, $m_{\text{[VA]}}^{dbr:Virginia}$

The presented rules only consider borderline cases, even though combinations of them can also be used within a corpus. For instance, a corpus might use the *OMAX* rule for LOC entities but apply *∅MIN* for all other entity types, therefore only yielding $m_{\text{[ETH Zurich]}}^{dbr:ETH\_Zurich}$ rather than

$m^{dbr:ETH\_Zurich}_{\text{[ETH Zurich]}}$ and $m^{dbr:Zurich}_{\text{[Zurich]}}$ for the text snippet *ETH Zurich.*

Table 3 outlines transformation rules between different annotation styles.

3.2.2 Knowledge Base Evolution

Lenses are also able to capture KB evolution, i.e. the case where a KB evolves due to changes or extended coverage of the underlying domain. Changes to the KB may

1. introduce new entities (e.g. the company Alphabet Inc. in October 2015),
2. lead to the deletion of entities that are no longer considered relevant, or
3. drive the introduction of a more fine grained or coarser mapping for existing entities.

Table 4 introduces the corresponding transformation rules. Newly introduced entities may enable the grounding of *NIL* entities to the extended knowledge base. The removal of an entity, in contrast, may transform an existing grounding to a *NIL* entity since the corresponding KB entity is no longer available. Finally, changes in granularity may either lead to the introduction of additional entities, or to the deletion of links to the KB.

3.2.3 Knowledge Base Migration

KB migration is the case in which a corpus that has been initially annotated with one KB is used to evaluate a component that links mentions to another KB. Many well maintained knowledge bases such as DBpedia, GeoNames and Wikidata contain links to indicate equivalent entities (e.g., *owl:sameAs*, *skos:exactMatch*, etc). These links and techniques such as ontology alignment may be used to automatize the transformation of a mentions $m^{e_i,KB}_{[x_i,y_i]}$ within a KB (KB) to the corresponding mention $m^{e_i,KB'}_{[x_i,y_i]}$ in the target KB (KB'). KB migration draws at the same set of transformation rules as the KB evolution use case.

3.2.4 Co-references

Co-references play a crucial role in natural language processing tasks such as relation extraction. A co-reference is a mention with surface form s'_i that refers to the same entities e_{ij} as other mentions $m^{e_{ij}}_{[s_{ij}]}$ within the document. Its surface form often contains prepositions or noun-phrases that on their own do not provide enough context to determine the referred entities e_{ij}. For anaphora and cataphora the co-reference $m^{e_i,KB}_{[s_i]}$ points to a single entity, for split antecedents the co-reference refers to multiple NEs. For instance, in the text: *"Berlin, Rome and Paris are capitals of European countries. These cities are also popular tourist destinations."* the surface form *These cities* refers to three previous named entities and is, therefore, annotated as $m^{dbr:Berlin}_{\text{[These cities]}}$, $m^{dbr:Rome}_{\text{[These cities]}}$ and $m^{dbr:Paris}_{\text{[These cities]}}$. Systems and corpora that do not support co-reference resolution, therefore, consider co-references as candidate mentions $m^{\emptyset}_{[s_i]}$ that do not link to any named entity.

3.3 Limitations of Corpus Transformation with Lenses

The rules outlined in the previous section can be used to translate between different corpus representations. Translations from expressive representations to less expressive ones can be done automatically and exposed to users as lenses. For example, a transformation from the OMAX to the ∅MIN annotation style, from a corpus with annotated co-references to a corpus which does not considers them, and the KB migration use case for NIL entities may be performed automatically.

Otherwise, corpus versioning (Section 4.2) is required to record any changes added by manual or semi-automatic processes.

4 Method

This section discusses options for improving corpus quality by (i) introducing semi-automatic tools that support corpus creation and evaluation by automatically spotting violations of the annotation style and suspicious entities (Section 4.1), and (ii) suggesting guidelines for versioning corpora ensuring that improvements and extensions are incorporated in a meaningful and backward compatible way (Section 4.2).

4.1 Corpus Analysis Tools

Software developers frequently use static code analysis tools such as pylint[1], findbugs[2] and checkstyle[3] as part of the build pipeline to enforce coding style guidelines and to spot potential bugs in an early stage.

[1] www.pylint.org

[2] findbugs.sourceforge.net

[3] checkstyle.sourceforge.net/

Table 3: Lense transformation rules between different annotation styles.

Annotation style	∅MIN	∅MAX	OMAX
Corpus entity	$m^{e_1,KB}_{[x1,y1]}$	$m^{e_1,KB}_{[x1,y11]}, \dots, m^{e_n,KB}_{[x1n,y1]}$	$m^{e_1,KB}_{[x1,y1]}, \dots, m^{e_n,KB}_{[x1,y1]}$
Transformation to			
∅MIN	$m^{e_1,KB}_{[x1,y1]}$	$m^{e_1,KB}_{[x1,y1]}$	$m^{e_1,KB}_{[x1,y1]}$
∅MAX	$m^{e_1,KB}_{[x1,y11]}, \dots, m^{e_n,KB}_{[x1n,y1]}$	$m^{e_1,KB}_{[x1,y11]}, \dots, m^{e_n,KB}_{[x1n,y1]}$	$m^{e_1,KB}_{[x1,y11]}, \dots, m^{e_n,KB}_{[x1n,y1]}$
OMAX	$m^{e_1,KB}_{[x1,y1]}, \dots, m^{e_n,KB}_{[x1,y1]}$	$m^{e_1,KB}_{[x1,y1]}, \dots, m^{e_n,KB}_{[x1,y1]}$	$m^{e_1,KB}_{[x1,y1]}, \dots, m^{e_n,KB}_{[x1,y1]}$

Table 4: Lense transformation rules for knowledge base evolution and knowledge base migration.

Task	new entity	deleted entity	more fine grained entity mapping	coarser entity mapping
Corpus entity	$m^{nil,KB}_{[x_i,y_i]}$	$m^{e_i,KB}_{[x_i,y_i]}$	$m^{e_i,KB}_{[x_i,y_i]}$	$m^{e_{i1}}_{[x_{i1},y_{i1}]}, \dots, m^{e_{in},KB}_{[x_{in},y_{in}]}$
Transformation	$m^{e_i,KB'}_{[x_i,y_i]}$	$m^{nil,KB'}_{[x_i,y_i]}$	$m^{e_{i1},KB'}_{[x_{i1},y_{i1}]}, \dots, m^{e_{in},KB'}_{[x_{in},y_{in}]}$	$m^{e_i,KB'}_{[x_i,y_i]}$

We strongly believe that similar tools could be highly beneficial for aiding researchers in the creation and validation of NLP corpora, by

1. automatically locating violations of annotation styles (e.g. overlaps in case of a non-overlapping annotation style).

2. drawing upon POS tagging and dependency parsing for marking unusual annotations such as NEs that do not contain a noun to flag potentially incorrect annotations.

4.2 Corpus Versioning

Even corpora that are frequently used in NEL evaluation suffer from quality issues (see Table 1). Although addressing these issues is important, backward compatibility of refined corpora is key to their usefulness since it ensures that results can be compared to previously published work.

We, therefore, suggest corpus versioning to promote the improvement of corpora. Publishing multiple corpus versions will enable researchers to run evaluations against these versions and, therefore, provides means to compare the gathered results to other work.

Corpus versioning is needed for cases where an automatic translation to the desired gold standard representation via lenses is not feasible:

- addressing data quality issues and mistakes in the original corpus,

- the linking of *NIL* entities to a knowledge base entity due to knowledge base evolution or knowledge base migration, and

- a new more expressive annotation style.

Versioning should enable researchers to address these issues while ensuring

- a clear relation to the original corpus that makes comparison with previous versions feasible;

- support for multiple versions and version trees that have been contributed by different people and organizations (Figure 1)

- that corpus metadata provides (i) information on the relations between different corpus versions, (ii) easily traceable contributions, (iii) credits for contributors, and (iv) easy re-use of refined corpora for further evaluations;

- easily accessible corpus versions, e.g. by uploading them to research data platforms and providing a digital object identifier (DOI) for each version, so that researchers can easily cite, locate and re-use corpora.

Table 5: Lense transformation rules for co-reference resolution.

Task	single co-reference	split antecedents
Corpus entity	$m^{e_i}_{[s'_i]}$	$m_{[s'_i}]^{e_{i1}}, \dots m_{[s'_i}]^{e_{in}}$
No co-reference resolution	$m^{\emptyset}_{[s'_i]}$	$m^{\emptyset}_{[s'_i]}$

Table 6: Suggested corpus metadata

Metadata	Description	Example
corpus_name	A name that identifies the corpus.	OKE2018
corpus_url	The corpus archive URL.	http://github.com/fhgr/oke2016-dbpedia-2019-02-01_v1.zip
creator	A comma-separated list of persons or organizations that created the corpus.	Sue May <sue@myorg.edu>
date	The corpus's publishing date.	2019-05-30
description	A description of the current corpus version.	Adapted OKE2016 to DBpedia 2019-02-01 and integrated bug fixes from 2018-03-07.
final	Is it usable for official evaluations?	false
parent_corpus_url	The URL of the parent corpus (if any)	http://github.com/fhgr/oke2016-dbpedia-2018-09-03_v2.zip
considers_corpus_url	A list of URLs pointing to related corpus versions.	http://github.com/sue/oke2016-dbpedia-fixes-2018-03-07.zip
annotation_style	A list of annotation styles per supported named entity type.	PER: NOMIN, GEO: OMAX, ORG: OMIN
annotators_per_document	Number of annotators per document.	3
annotator_agreement	Inter-rater-agreement between annotators computed using the Fleiss' kappa.	0.61

The key here is making evaluations easier to replicate and increase the benefit they provide to the community. Currently evaluations are often tightly designed to a specific context such as a competition or an application domain. These kinds of results are helpful for determining the best performing system under tight restrictions, but they unnecessarily restrict the scope of the evaluation. For instance, such results do not provide information on how systems cope with different annotation rules, settings and use cases. Automatically performing evaluations with all available lenses, corpus versions and scoring rules (Section 4.3) could address this issue. Scientists could still publish the results for their particular application context in the research paper but would in addition provide a DOI to the full results that cover also settings for which their system hasn't been optimized. Ultimately such an approach would improve the usefulness of evaluations since it would provide (i) much more context on the strengths and weaknesses of NEL systems, and (ii) broader insights into the effects of the suggested methods and design decisions.

4.2.1 Publishing a Corpus Version

We recommend a standardized directory structure for publishing corpora that contains:

1. a `corpus` directory containing all corpus data and annotations in the NIF format.
2. a `METADATA.yaml` file that describes the corpus based on the metadata introduced in Table 6.
3. a `README.md` file which provides additional unstructured information.

Popular version control services such as github and gitlab offer a release feature that automatically

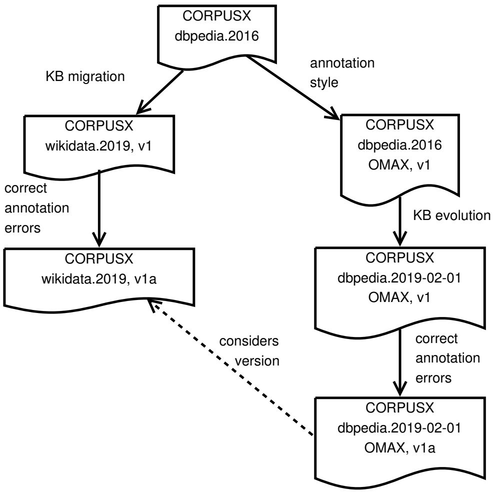


Figure 1: Common corpus versioning use cases.

publishes an archive of the released repository version which can be used to publish a certain corpus version. Another option would be publishing corpus versions in research data repositories such as zenodo.org which also provide a DOI and bibliographical metadata to data artifacts.

4.3 Scoring Rules

Scoring rules outline the conditions under which a gold standard corpus mention $m^c := m^{e_c,KB}_{[x_c,y_c]}$ and a mention returned by the NEL system $m^s := m^{e_s,KB}_{[x_s,y_s]}$ are considered equivalent to each other. The following three scoring rules are frequently used in NEL evaluations:

1. perfect match $\mathcal{P}$ - the entities refer to the same KB entity e_i, and the exactly same surface form s_i.

2. contained match $\mathcal{C}$ - both entities refer to the same KB entity e_i and the surface form of the mention returned by the NEL system m^s is contained in the surface form of the corpus mention m^c, i.e. $x^s_i \geq x^c_i$ and $y^s_i \leq y^c_i$.

3. overlapping match $\mathcal{O}$ - this case is equivalent to the contained match but further relaxes the restrictions on the surface form, so that even an overlap (i.e. $y^s_i \geq x^c_i$ and $x^s_i \leq y^c_i$) between entities is considered a valid match.

The used scoring rule have a significant impact on the computation of the NEL system's performance metrics such as precision and recall.

5 Conclusion

This paper discusses approaches for addressing the issues of corpus quality and the comparability of evaluations that have been performed with these corpora, together with associated annotations[4]. We discuss (i) factors that seriously affect the accuracy of evaluation corpora such as different annotation styles, missing and wrong annotations, KB evolution and co-reference handling.

In addition we also shed light on the issue of KB migration, which is relevant if the evaluation corpus and the NEL system use different KBs. Afterwards we (ii) introduce a formalization that captures these factors, and (iii) present transformation rules between different corpus configurations.

These transformation rules that expose different corpus configurations as lenses in conjunction with corpus analysis tools and corpus versioning are key towards improving corpus quality. Well-defined scoring rules and evaluation metrics are further steps towards standardizing evaluations and improving their validity and reproducibility.

Future research will focus on (i) applying these guidelines to the NEL evaluation of annotation of TV-related content in the ReTV project[5] so that results can be compared and evolved in the future if need be,(ii) the creation of tools that support corpus creation and evaluation processes, (iii) adding support for corpus versioning and the parallel analysis of multiple corpus versions to evaluation tools such as Orbis (Odoni et al., 2018) and GERBIL (Röder et al., 2018), and (iii) proving a research data infrastructure for publishing evaluation corpora and evaluations that have been performed on these corpora.

Acknowledgments

This work has been supported by the MedMon project (https://www.htwchur.ch/medmon) funded by Innosuisse and the ReTV project (https://retv-project.eu/) funded through the European Union's Horizon 2020 Research and Innovation Programme under GA No 780656 and through the FFG project EPOCH (https://epoch-project.eu).

[4] Annotations discussed in Section 3 are available at https://github.com/orbis-eval/corpus_quality_paper

[5] To be published in D1.2 deliverable at `https://retv-project.eu/deliverables/`

References

Rodrigo Agerri, Yiling Chung, Itziar Aldabe, Nora Aranberri, Gorka Labaka, and German Rigau. 2018. Building named entity recognition taggers via parallel corpora. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Kôiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*. European Language Resources Association (ELRA). http://www.lrec-conf.org/lrec2018.

Adrian M. P. Braşoveanu, Giuseppe Rizzo, Philipp Kuntschick, Albert Weichselbraun, and Lyndon J.B. Nixon. 2018. Framing named entity linking error types. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), Paris, France, pages 266–271. http://www.lrec-conf.org/proceedings/lrec2018/summaries/612.html.

Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors. 2016. *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA). http://www.lrec-conf.org/lrec2016.

Maud Ehrmann, Damien Nouvel, and Sophie Rosset. 2016. Named entity resources - overview and outlook. In (Calzolari et al., 2016). http://www.lrec-conf.org/proceedings/lrec2016/summaries/987.html.

Ben Hachey, Joel Nothman, and Will Radford. 2014. Cheap and easy entity evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. The Association for Computer Linguistics, pages 464–469. http://aclweb.org/anthology/P/P14/P14-2076.pdf.

Benjamin Heinzerling and Michael Strube. 2015. Visual error analysis for entity linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, System Demonstrations*. ACL, pages 37–42. http://aclweb.org/anthology/P/P15/P15-4007.pdf.

Kunal Jha, Michael Röder, and Axel-Cyrille Ngonga Ngomo. 2017. All that glitters is not gold - rule-based curation of reference datasets for named entity recognition and entity linking. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*. volume 10249 of *Lecture Notes in Computer Science*, pages 305–320. https://doi.org/10.1007/978-3-319-58068-5_19.

Heng Ji, Xiaoman Pan, Boliang Zhang, Joel Nothman, James Mayfield, Paul McNamee, and Cash Costello. 2017. Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the 2017 Text Analysis Conference, TAC 2017, Gaithersburg, Maryland, USA, November 13-14, 2017*. NIST. https://tac.nist.gov/publications/2017/papers.html.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*. Association for Computational Linguistics, pages 861–871. https://aclanthology.info/papers/N18-1079/n18-1079.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Begoña Altuna, Marieke van Erp, Anneleen Schoen, and Chantal van Son. 2016. Meantime, the newsreader multilingual event and time corpus. In (Calzolari et al., 2016). http://www.lrec-conf.org/proceedings/lrec2016/summaries/488.html.

Axel-Cyrille Ngonga Ngomo, Michael Röder, Diego Moussallem, Ricardo Usbeck, and René Speck. 2018. BENGAL: an automatic benchmark generator for entity recognition and linking. In Emiel Krahmer, Albert Gatt, and Martijn Goudbeek, editors, *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*. Association for Computational Linguistics, pages 339–349. https://aclanthology.info/papers/W18-6541/w18-6541.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Robert Meusel, and Heiko Paulheim. 2016. The second open knowledge extraction challenge. In Harald Sack, Stefan Dietze, Anna Tordai, and Christoph Lange, editors, *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*. Springer, volume 641 of *Communications* *in Computer and Information Science*, pages 3–16. https://doi.org/10.1007/978-3-319-46565-4_1.

Fabian Odoni, Philipp Kuntschik, Adrian M. P. Braşoveanu, and Albert Weichselbraun. 2018. On the importance of drill-down analysis for assessing gold standards and named entity linking performance. In Anna Fensel, Victor de Boer, Tassilo Pellegrini, Elmar Kiesling, Bernhard Haslhofer, Laura Hollink, and Alexander Schindler, editors, *Proceedings of the 14th International Conference on Semantic Systems, SEMANTICS 2018, Vienna, Austria, September 10-13, 2018*. Elsevier, volume 137 of *Procedia Computer Science*, pages 33–42. https://doi.org/10.1016/j.procs.2018.09.004.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - A collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*. pages 3529–3533. http://www.lrec-conf.org/proceedings/lrec2014/summaries/856.html.

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. GERBIL - benchmarking named entity recognition and linking consistently. *Semantic Web* 9(5):605–625. https://doi.org/10.3233/SW-170286.

Henry Rosales-Méndez, Barbara Poblete, and Aidan Hogan. 2018. What should entity linking link? In Dan Olteanu and Barbara Poblete, editors, *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018.*. CEUR-WS.org, volume 2100 of *CEUR Workshop Proceedings*. http://ceur-ws.org/Vol-2100/paper10.pdf.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland, pages 859–866. http://www.lrec-conf.org/proceedings/lrec2014/pdf/497_Paper.pdf.

Ahmad Sakor, Saeedeh Shekarpour, Maria-Esther Vidal, Jens Lehmann, and Sören Auer. 2018. Old is gold: Linguistic driven approach for entity and relation linking of short text pages 339–349.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2008. Accelerating the annotation of sparse named entities by dynamic sentence selection. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*. Association for Computational Linguistics, Columbus, Ohio, pages 30–37. https://www.aclweb.org/anthology/W08-0605.

Marieke van Erp, Pablo N. Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. 2016. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*. pages 4373–4379.

Jörg Waitelonis, Henrik Jürges, and Harald Sack. 2019. Remixing entity linking evaluation datasets for focused benchmarking. *Semantic Web* 10(2):385–412. https://doi.org/10.3233/SW-180334.

Sequential Graph Dependency Parser

Sean Welleck
New York University
wellecks@nyu.edu

Kyunghyun Cho
New York University
CIFAR Azrieli Global Scholar
Facebook AI Research
kyunghyun.cho@nyu.edu

Abstract

We propose a method for non-projective dependency parsing by incrementally predicting a set of edges. Since the edges do not have a pre-specified order, we propose a set-based learning method. Our method blends graph, transition, and easy-first parsing, including a prior state of the parser as a special case. The proposed transition-based method successfully parses near the state of the art on both projective and non-projective languages, without assuming a certain parsing order.

1 Introduction

Dependency parsing methods can be categorized as graph-based and transition-based. Typical graph-based methods support non-projective parsing, but introduce independence assumptions and rely on external decoding algorithms. Conversely, transition-based methods model joint dependencies, but without modification are typically limited to projective parses.

There are two recent exceptions of interest here. (Ma et al., 2018) developed the Stack-Pointer parser, a transition-based, non-projective parser that maintains a stack populated in a top-down, depth-first manner, and uses a pointer network to determine the dependent of the stack's top node, resulting in a transition sequence of length $2n-1$. Recently, (Fernández-González and Gómez-Rodrguez, 2019) developed a variant of the Stack-Pointer parser which parses in n steps by traversing the sentence left-to-right, selecting the *head* of the current node in the traversal, while incrementally checking for, and prohibiting, cycles.

We take inspiration from both graph-based and transition-based approaches by viewing parsing as sequential graph generation. In this view, a graph is incrementally built by adding edges to an edge set. No distinction between projective and non-projective trees is necessary. Since edges do not have a pre-specified order, we propose a set-based learning method. Like (Fernández-González and Gómez-Rodrguez, 2019), our parser runs in n steps. However, our learning method and transitions do not impose a left-to-right parsing order, allowing easy-first (Tsuruoka and Tsujii, 2005; Goldberg and Elhadad, 2010) behavior. Experimentally, we find that the proposed method can yield a sequential parser with preferred, input-dependent generation orders and performance gains over strong one-step methods.[1]

2 Graph Dependency Parser

Given a sentence $x = x_1, \ldots, x_N$, a dependency parser constructs a graph $G = (V, E)$ with $V = (x_0, x_1, \ldots, x_N)$ and $E = \{(i,j)_1, \ldots (i,j)_N\}$, where x_0 is a special root node, and $E \subset \mathcal{E}$ forms a dependency tree.[2]

We describe a family of sequential graph-based dependency parsers. A parser in this family generates a *sequence* of graphs where V is fixed and $E = \bigcup_{t=1}^{T} E_t$:

$$H^{\text{enc}} = f_{\text{enc}}(x_0, \ldots x_N) \quad (1)$$

$$H_t^{\text{head}}, H_t^{\text{dep}} = f_{\text{V}}(H^{\text{enc}}, E_{<t}, h_{t-1}) \quad (2)$$

$$S_t = f_{\text{E}}(H_t^{\text{head}}, H_t^{\text{dep}}, S_{t-1}) \quad (3)$$

$$E_t = f_{\text{dec}}(S_t, E_{<t}). \quad (4)$$

Steps (2-4) run for $T \le N$ time-steps. At each time-step, first f_V generates head and dependent

[1] Code will be made available at https://github.com/wellecks/nonmonotonic_parsing.

[2] See properties (1-5) in Appendix A.

representations for each vertex, $H_t^{\cdot} \in \mathbb{R}^{V \times d_H}$, based on vertex representations $H^{\text{enc}} \in \mathbb{R}^{V \times d}$, previously predicted edges $E_{<t}$, and a recurrent state $h_{t-1} \in \mathbb{R}^d$. Then f_E computes a score for every possible edge, $S_t \in \mathbb{R}^{V \times V}$, and the scores are used by f_{dec} to predict a set of edges E_t.

This general sequential family includes the biaffine parser of (Dozat and Manning, 2017) as a one-step special case, as well as a recurrent variant which we discuss below.

2.1 Biaffine One-Step

The Biaffine parser of (Dozat and Manning, 2017) is a one-step variant, implementing steps (1-4) using a bidirectional LSTM, head and dependent neural networks, a biaffine scorer, and a maximum spanning tree decoder, respectively:

$$\begin{aligned} H^{\text{enc}} &= \text{BiLSTM}(x_1, \ldots, x_N) \\ H^{\text{head}}, H^{\text{dep}} &= \text{MLP}^{\text{h}}(H^{\text{enc}}), \text{MLP}^{\text{d}}(H^{\text{enc}}) \\ S &= \text{BiAffine}(H^{\text{head}}, H^{\text{dep}}) \\ E &= \text{MST}(S), \end{aligned}$$

where each row of scores $S^{(i)}$ is interpreted as a distribution over i's potential head nodes:

$$p((j \rightarrow i)|x) \propto \text{softmax}_j(S^{(i)}),$$

and $\text{MST}(\cdot)$ is an off-the-shelf maximum-spanning-tree algorithm. This model assumes conditional independence of the edges.

2.2 Recurrent Weight

We propose a variant which iteratively adjusts a distribution over edges at each step, based on the predictions so far. A recurrent function generates a weight matrix W which is used to form vertex embeddings and in turn adjust edge scores.

Specifically, we first obtain an initial score matrix S_0 using the biaffine one-step parser (2.1), and initialize a recurrent hidden state h_0 using a linear transformation of f_{enc}'s final hidden state. Then f_V is defined as:

$$\begin{aligned} W, h_t &= \text{LSTM}(f_{\text{emb}}(E_{t-1}), h_{t-1}) \\ H_t^{\text{head}} &= \text{emb}_h(0, \ldots, N)W \\ H_t^{\text{dep}} &= \text{emb}_d(0, \ldots, N)W, \end{aligned}$$

and $f_{\text{E}}(H_t^{\text{head}}, H_t^{\text{dep}}, S_{t-1})$ is defined as:

$$\begin{aligned} S_t^{\Delta} &= \text{BiAffine}(H_t^{\text{head}}, H_t^{\text{dep}}) \\ S_t &= S_{t-1} + S_t^{\Delta}, \end{aligned}$$

where t ranges from 1 to N, $W \in \mathbb{R}^{d_{\text{emb}} \times d_H}$, and each $\text{emb}_{(\cdot)} : \mathbb{N} \rightarrow \mathbb{R}^{d_{\text{emb}}}$ is a learned embedding layer, yielding $\text{emb}_{(\cdot)}(0, \ldots, N)$ in $\mathbb{R}^{V \times d_{\text{emb}}}$. We use a bidirectional LSTM as f_{enc}.

The scores at each step yield a distribution over all $V \times V$ edges, which we denote by π:

$$\pi((i \rightarrow j)|E_{<t}, x) \propto \text{softmax}(\text{flatten}(S_t)). \quad (5)$$

Unlike the one-step model, this recurrent model can predict edges based on past predictions.

Inference We must ensure the incrementally decoded edges $E = \bigcup_{t=1}^{T} E_t$ form a valid dependency tree. To do so, we choose f_{dec} to be a decoder which greedily selects valid edges,

$$E_t = f_{\text{valid}}(S_t, E_{<t}),$$

which we refer to as the **valid decoder**, detailed in Appendix A. We only predict one edge per step ($|E_t| = 1$), leaving the setting of multiple predictions per step as future work.

Embedding Edges We embed a predicted edge $E_t = \{(\widehat{i, j})\}$ as:

$$\begin{aligned} f_{\text{emb}}(E_t) &= e_{\text{edge}}; e_{\text{head}}; e_{\text{dependent}} \\ e_{\text{edge}} &= W_e H_{(i)}^{\text{enc}} - W_e H_{(j)}^{\text{enc}} \\ e_{\text{head}} &= \text{emb}_h(i) \\ e_{\text{dependent}} &= \text{emb}_d(j), \end{aligned}$$

where $H_{(\cdot)}^{\text{enc}} \in \mathbb{R}^d$ are row vectors, $W_e \in \mathbb{R}^{d_e \times d}$ is a learned weight matrix, $\text{emb}_{(\cdot)}$ are learned embedding layers, and ; is concatenation.

Future Work The proposed method does not specifically require a BiLSTM encoder, LSTM, or the BiAffine function. For instance, f_{V} could use a Transformer (Vaswani et al., 2017) to output states that are linearly transformed into H^{head} and H^{dep}. Additionally, partial graphs $(V, E_{<t})$ might be embedded using neural networks specifically designed for graphs (Gilmer et al., 2017). Finally, predicting edge sets of size greater than 1 could potentially be achieved using a partially-autoregressive model, trained with a 'masked

edges' objective, similar to recent work in machine translation with conditional masked language models (Ghazvininejad et al., 2019). Each call to f_V would involve a separate forward pass which calls a Transformer f_{enc}. The partial tree is encoded via non-masked inputs to f_{enc}. f_{E} corresponds to having V outputs, each a distribution over V edges. The multi-step decoder (Appendix A) might be used at test time.

3 Learning

In this paper, we restrict to the case of predicting a single edge $\widehat{(i,j)}$ per step, so that the recurrent weight model generates a sequence of edges with the goal of matching a target edge set, i.e. $\bigcup_{t=1}^{N} \widehat{(i,j)}_t = E$. Since the target edges E are a set, the model's generation order is not determined *a priori*. As a result, we propose to use a learning method that does not require a pre-specified generation order and allows the model to learn input-dependent orderings.

Our proposed method is based on the multiset loss (Welleck et al., 2018) and its recent extensions for non-monotonic generation (Welleck et al., 2019). The method is motivated from the perspective of learning-to-search (Daumé III et al., 2009; Chang et al., 2015), which involves learning a *policy* π_θ that mimics an *oracle policy* π^*. The policy maps *states* to distributions over *actions*.

For the proposed graph parser, an action is an edge $(i, j) \in \mathcal{E}$, and a state s_t is an input sentence x along with the edges predicted so far, $\hat{E}_{<t}$. The policy is a conditional distribution over $\mathcal{E}$,

$$\pi_\theta((i,j)|\hat{E}_{<t}, x),$$

such as the distribution in equation (5).

Learning consists of minimizing a cost, computed by first sampling states from a *roll-in* policy π^{in}, then using a *roll-out policy* π^{out} to estimate cost-to-go for all actions at the sampled states. Formally, we minimize the following objective with respect to θ:

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{s_1,\ldots,s_{|x|}\sim\pi^{\text{in}}}\mathcal{C}(\pi_\theta, \pi^{\text{out}}, s_t). \tag{6}$$

This objective involves sampling a sentence x from a dataset, sampling a sequence of edges from the roll-in policy, then computing a cost $\mathcal{C}$ at each of the resulting states. We now describe our choices of $\mathcal{C}$, π^{out}, π^*, and π^{in}, and evaluate them later in the experiments (4).

3.1 Cost Function and Roll-Out

Following (Welleck et al., 2018, 2019) we use a KL-divergence cost:

$$\mathcal{C}(\pi_\theta, \pi^{\text{out}}, s) = D_{\text{KL}}(\pi^{\text{out}}(\cdot|s)||\pi_\theta(\cdot|s)). \tag{7}$$

We use the oracle π^* as the roll-out π^{out}.

3.2 Oracle

Based on the free labels set in (Welleck et al., 2018), we first define a *free edge set* containing the un-predicted target edges at time t:

$$E^t_{\text{free}} = E \setminus \bigcup_{t'=1}^{t-1} \widehat{(i,j)}_{t'}, \tag{8}$$

where $E^0_{\text{free}} = E$. We then construct a family of oracle policies that place non-zero probability mass only on free edges:

$$\pi^*((i,j)|E^t_{\text{free}}) = \begin{cases} p_{ij} & (i,j) \in E^t_{\text{free}} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We now describe several oracles by varying how p_{ij} is defined.

Uniform This oracle treats each permutation of the target edge set as equally likely by assigning a uniform probability to each free edge:

$$\pi^*_{\text{unif}}((i,j)|E^t_{\text{free}}) = \begin{cases} \frac{1}{|E^t_{\text{free}}|} & (i,j) \in E^t_{\text{free}} \\ 0 & \text{otherwise.} \end{cases}$$

Coaching Motivated by (He et al., 2012; Welleck et al., 2019), we define a coaching oracle which weights free edges by π_θ:

$$\pi^*_{\text{coaching}}((i,j)|E^t_{\text{free}}) \propto \pi^*_{\text{unif}}(\cdot|E^t_{\text{free}})\pi_\theta(\cdot|E_{<t}, X).$$

This oracle prefers certain edge permutations over others, reinforcing π_θ's preferences. The coaching and uniform oracles can be mixed to ensure each free edge receives probability mass:

$$\beta\pi^*_{\text{unif}} + (1-\beta)\pi^*_{\text{coaching}}, \tag{10}$$

where $\beta \in [0, 1]$.

Annealed Coaching This oracle begins with the uniform oracle, then anneals towards the coaching oracle as training progresses by annealing the β term in (10). This may prevent the coaching oracle from reinforcing sub-optimal permutations early in training.

Linearized This oracle uses a deterministic function to linearize an edge set E into a sequence E_{seq}. The oracle selects the t'th element of E_{seq} at time t with probability 1. We linearize an edge set in increasing edge-index order: (i_1, j_1) precedes (i_2, j_2) if $(i_1, j_1) < (i_2, j_2)$. This oracle serves as a baseline that is analogous to the fixed generation orders used in conventional parsers.

3.3 Roll-In

The roll-in policy determines the state distribution that π_θ is trained on, which can address the mismatch between training and testing state distributions (Ross et al., 2011; Chang et al., 2015) or narrow the set of training trajectories. We evaluate several alternatives:

1. **uniform** $(i, j) \sim \pi^*_{\text{unif}}$
2. **coaching** $(i, j) \sim \pi_\theta \odot \pi^*_{\text{unif}}$
3. **valid-policy** $(i, j) \sim \text{valid}(\pi_\theta)$

where valid(π_θ) is the set of edges that keeps the predicted tree as a valid dependency tree. The coaching and valid-policy roll-ins choose edge permutations that are preferred by the policy, with valid-policy resembling test-time behavior.

4 Experiments

In Experiments 4.1 and 4.2 we evaluate on English, German, Chinese, and Ancient Greek since they vary with respect to projectivity, size, and performance in (Qi et al., 2018). Based on these development set results, we then test our strongest model on a large suite of languages (4.3).

Experimental Setup Experiments are done using datasets from the CoNLL 2018 Shared Task (Zeman et al., 2018). We build our implementation from the open-source version of (Qi et al., 2018)[3], and use their experimental setup (e.g. pre-processing, data-loading, pre-trained vectors, evaluation) which follows the shared task setup. Our model uses the same encoder from (Qi et al., 2018). For the (Qi et al., 2018) baseline, we use their pretrained models[4] and evaluation script. For the (Dozat and Manning, 2017) baseline, we use the (Qi et al., 2018) implementation with auxiliary outputs and losses disabled, and train with the default hyper-parameters and training script. For our models only, we changed the learning rate schedule (and model-specific hyper-parameters), after observing diverging loss in preliminary experiments with the default learning rate. Our models did not require the additional AMSGrad technique used in (Qi et al., 2018). We evaluate validation UAS every 2k steps (vs. 100 for the baseline). Models are trained for up to 100k steps, and the model with the highest validation unlabeled attachment score (UAS) is saved.

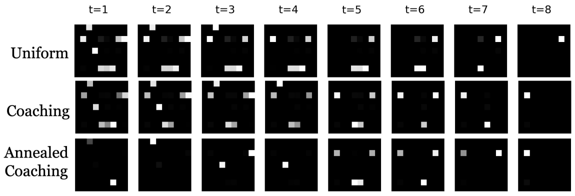


Figure 1: Per-step edge distributions from recurrent weight models trained with the given oracle.

4.1 Multi-Step Learning

In this experiment we evaluate the sequential aspect of the proposed recurrent model by comparing it with one-step baselines. We compare against a baseline ('One-Step') that simply uses the first step's score matrix S_0 from the recurrent weight model and minimizes (6) for one time-step using a uniform oracle. At test time the valid decoder uses S_0 for all timesteps. We also compare against the biaffine one-step model of (Dozat and Manning, 2017) which uses Chu-Liu-Edmonds maximum spanning tree decoding instead of valid decoding. Since we only evaluate UAS, we disable its edge label output and loss. Finally, we compare against (Qi et al., 2018) which is based on (Dozat and Manning, 2017) plus auxiliary losses for length and linearization prediction.

Results are shown in Table 1, including results for

[3] https://github.com/stanfordnlp/stanfordnlp.

[4] https://stanfordnlp.github.io/stanfordnlp/installation_download.html.

	En	De	Grc	Zh
D & M (2017)	91.14	90.38	78.99	86.50
Qi et al. (2018)	92.11	89.46	81.35	86.73
One-Step	91.74	91.07	79.60	86.61
Recurrent (U)	91.92	91.02	79.15	86.69
Recurrent (C)	91.99	91.19	79.93	86.77

Table 1: Development set UAS for single vs. multi-step methods. (U) is uniform oracle and roll-in, (C) is coaching with greedy valid roll-in ($\beta = 0.5$). D & M (2017) is an abbreviation for (Dozat and Manning, 2017).

a recurrent model trained with coaching ('Recurrent (C)') using a mixture (eq. 10) with $\beta = 0.5$. The one-step baseline is strong, even outperforming the uniform recurrent variant on some languages. The recurrent weight model with coaching, however, outperforms the one-step and (Dozat and Manning, 2017) baselines on all four languages. Adding in auxiliary losses to the (Dozat and Manning, 2017) model yields improved UAS as seen in the (Qi et al., 2018) performance, suggesting that our proposed recurrent model might be improved further with auxiliary losses.

Temporal Distribution Adjustment Figure 1 shows per-step edge distributions on an eight-edge example. The recurrent weight variants learned to adjust their distributions over time based on past predictions. The model trained with the uniform oracle has a decreasing number of high probability edges per step since it aims to place equal mass on each free edge $(i, j) \in \hat{E}^t_{\text{free}}$. The model trained with coaching learned to prefer certain free edges over others, but with $\beta = 0.5$ the uniform term in the loss still encourages placing mass on multiple edges per step. By annealing β, however, the coaching model exhibits vastly different behavior than the uniform-trained policy. The low entropy distributions at early steps followed by higher entropy distributions later on (e.g. $t \in \{5, 6\}$) may indicate easy-first behavior.

4.2 Oracle and Roll-In Choice

In this experiment, we study the effects of varying the oracle and roll-in distributions. Table (2) shows results on German, analyzed below. Models trained with coaching (C) use a mixture with $\beta = 0.5$, after observing lower UAS in preliminary experiments with lower β. The π^*_{coach} and $\pi^{\text{valid}}_\theta$ roll-ins use a mixture with $\beta = 0.5$ and greedy decoding, which generally outperformed stochastic sampling.

Oracle	Roll-in	UAS	Loss
Linear	π^*_{linear}	81.03	0.04
U	π^*_{unif}	91.02	0.35
C	π^*_{unif}	91.04	0.17
U	π^*_{coach}	90.93	0.45
C	π^*_{coach}	91.17	0.33
CA	π^*_{coach}	90.89	0.34
U	$\pi^{\text{valid}}_\theta$	90.99	0.51
C	$\pi^{\text{valid}}_\theta$	**91.19**	0.31
CA	$\pi^{\text{valid}}_\theta$	90.91	0.30

Table 2: Varying oracle and roll-in policies on German. (U), (C), (A) refer to uniform, coaching, and annealing, respectively. The π^*_{coach} and $\pi^{\text{valid}}_\theta$ roll-ins are mixtures with a uniform oracle, with $\beta = 0.5$ for coaching (C), and β linearly annealed by 0.02 every 2000 steps for annealing (CA).

Set-Based Learning The model trained with the linearized oracle (UAS 81.03), which teaches the model to adhere to a pre-specified generation order, significantly under-performs the set-based models (UAS $\geq$ 90.89), which do not have a pre-specified generation order and can in principle learn strategies such as easy-first.

Coaching Models trained with coaching (C, UAS $\geq$ 91.04) had higher UAS and lower loss than models trained with the uniform oracle (U, UAS $\leq$ 91.02), for all roll-in methods. This suggests that for the proposed model, weighting free edges in the loss based on the model's distribution is more effective than a uniform weighting.

Annealing the β parameter generally did not further improve UAS (CA vs. C), possibly due to the annealing schedule or overfitting; despite lower losses with annealing, eventually validation UAS *decreased* as training progressed.

Roll-In With the coaching oracle (C), the choice of roll-in impacted UAS, with coaching roll-in (π^*_{coach}, 91.17) and valid roll-in ($\pi^{\text{valid}}_\theta, 91.19$) achieving higher UAS than uniform oracle roll-in ($\pi^*_{\text{unif}}, 91.04$). This suggests that when using coaching, narrowing the set of training trajectories

	Ours	Qi et al. (2018)
AR	88.22	**88.35**
CA	**94.13**	**94.13**
CS (CAC)	**93.53**	93.22
CS (PDT)	**93.80**	93.21
DE	**88.39**	87.21
EN (EWT)	**91.28**	91.21
ES	**93.70**	93.38
ET	**89.56**	89.40
FR (GSD)	**91.07**	90.90
GRC (Perseus)	80.90	**82.77**
HI	**96.78**	**96.78**
IT (ISDT)	94.06	**94.24**
KO (KAIST)	**91.02**	90.55
LA (ITTB)	**93.66**	93.00
NO (Bokmaal)	**94.63**	94.27
NO (Nynorsk)	**94.44**	94.02
PT	91.22	**91.67**
RU (SynTagRus)	**94.57**	94.42
ZH	87.31	**88.49**

Table 3: Test set results (UAS) on datasets from the CoNLL 2018 shared task with greater than 200k examples, plus the Ancient Greek (GRC) and Chinese (ZH) datasets. Bold denotes the highest UAS on each dataset.

to those preferred by the policy may be more effective than sampling uniformly from the set of all correct trajectories. Based on these results, we use the coaching oracle and valid roll-in for training our final model in the next experiment.

4.3 CoNLL 2018 Comparison

In this experiment, we evaluate our best model on a diverse set of multi-lingual datasets. We use the CoNLL 2018 shared task datasets that have at least 200k examples, along with the four datasets used in the previous experiments. We train a recurrent weight model for each dataset using the coaching oracle and valid roll-in. We compare against (Qi et al., 2018) which placed highly in the CoNLL 2018 competition, reporting test UAS evaluated using their pre-trained models.

Table 3 shows the results on the 19 datasets from 17 different languages. The proposed model trained with coaching achieves a higher UAS than the Qi et al. (2018) model on 12 of the 19 datasets, plus two ties.

5 Related Work

Transition-based dependency parsing has a rich history, with methods generally varying by the choice of transition system and feature representation. Traditional stack-based arc-standard and arc-eager (Yamada and Matsumoto, 2003; Nivre, 2003) transition systems only parse projectively, requiring additional operations for pseudo-non-projectivity (Gómez-Rodríguez et al., 2014) or projectivity (Nivre, 2009), while list-based non-projective systems have been developed (Nivre, 2008). Recent variations assume a generation order such as top-down (Ma et al., 2018) or left-to-right (Fernández-González and Gómez-Rodrguez, 2019). Other recent models focus on unsupervised settings (Kim et al., 2019). Our focus here is a non-projective transition system and learning method which does not assume a particular generation order.

A separate thread of research in sequential modeling has demonstrated that generation order can affect performance (Vinyals et al., 2015), both in tasks with set-structured outputs such as objects (Welleck et al., 2017, 2018) or graphs (Li et al., 2018), and in sequential tasks such as language modeling (Ford et al., 2018). Developing models with relaxed or learned generation orders has picked up recent interest (Welleck et al., 2018, 2019; Gu et al., 2019; Stern et al., 2019). We investigate this for dependency parsing, framing the problem as sequential set generation without a pre-specified order.

Finally, our work is inspired by techniques for improving upon maximum likelihood training through error exploration and dynamic oracles (Goldberg and Nivre, 2012, 2013), and related techniques in imitation learning for structured prediction (Daumé III et al., 2009; Ross et al., 2011; He et al., 2012; Goodman et al., 2016). In particular, our formulation is closely related to the framework of (Chang et al., 2015), where our oracle can be seen as an optimal roll-out policy which computes action costs without explicit roll-outs.

6 Conclusion

We described a family of dependency parsers which construct a dependency tree by generating a sequence of edge sets, and a learning method that does not presuppose a generation order. Experimentally, we found that a 'coaching' method, which weights actions in the loss according to the model, improves parsing accuracy compared to a uniform weighting and allows the parser to learn preferred, input-dependent generation orders. The model's sequential aspect, along with the coaching method and training on a state distribution which resembles the model's own behavior, yielded improvements in unlabeled dependency parsing over strong one-step baselines.

References

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based Structured Prediction. Technical report.

Timothy Dozat and Christopher D Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *International Conference on Learning Representations (ICLR)*.

Daniel Fernández-González and Carlos Gómez-Rodrguez. 2019. Left-to-right dependency parsing with pointer networks.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George E Dahl. 2018. The importance of generation order in language modeling. *arXiv preprint arXiv:1808.07910*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. Technical report.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order.

He He, Jason Eisner, and Hal Daume. 2012. Imitation learning by coaching. In *Advances in Neural Information Processing Systems*, pages 3149–3157.

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised Recurrent Neural Network Grammars.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. 2018. Learning deep generative models of graphs. In *ICML 2018*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-Pointer Networks for Dependency Parsing. Technical report.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Workshop on Parsing Technologies (IWPT*, pages 149–160, Nancy, France.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359. Association for Computational Linguistics.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 467–474. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order Matters: Sequence to sequence for sets.

Sean Welleck, Kiant Brantley, Hal Daum III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation.

Sean Welleck, Kyunghyun Cho, and Zheng Zhang. 2017. Saliency-based sequential image attention with multiset prediction. In *Advances in neural information processing systems*.

Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. 2018. Loss functions for multiset prediction. In *Advances in Neural Information Processing Systems*, pages 5788–5797.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector machines. In *The 8th International Workshop of Parsing Technologies (IWPT2003)*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

A Sequential Valid Decoder

We wish to sequentially sample $E_1, E_2, \dots, E_T$ from score matrices $S_1, S_2, \dots, S_T$, respectively, such that $E = \bigcup_t E_t$ is a dependency tree. A dependency tree must satisfy:

1. The root node has no incoming edges.
2. Each non-root node has exactly one incoming edge.
3. There are no duplicate edges.
4. There are no self-loops.
5. There are no cycles.

We first consider predicting one edge per step $|E_t| = 1$, then address the case $|E_t| \geq 1$.

One Edge Per Step Let $x = x_0, x_1, \dots, x_N$ where x_0 is a root node. We define a function $f_{\text{valid}}(S_t, E_{<t}) \rightarrow (i, j)$ which chooses the highest scoring edge (i, j) such that $E_{<t} \cup \{(i, j)\}$ is a dependency tree, given edges $E_{<t}$ and scores S_t. We represent $E_{<t}$ as an adjacency matrix $A_{<t}$, and implement $f_{\text{valid}}(S_t, A_{<t})$ by masking S_t to yield scores $\tilde{S}$ that satisfy (1-5) as follows:

1. $\tilde{S}_{\cdot,0} = -\infty$
2. $A_{i,j} = 1$ implies $\tilde{S}_{\cdot,j} = -\infty$
3. $A_{i,j} = 1$ implies $\tilde{S}_{i,j} = -\infty$
4. $\tilde{S}_{i,i} = -\infty$ for all i
5. $R_{i,j} = 1$ implies $\tilde{S}_{j,i} = -\infty$, where $R \in \{0, 1\}^{N \times N}$ is the reachability matrix (transitive closure) of A. That is, $R_{i,j} = 1$ when there is a directed path from i to j. [5]

The selected edge is then $\arg\max_{(i,j)} \tilde{S}_{i,j}$.

A full tree is decoded by calling f_{valid} for T steps, using the current step scores S_t and an adjacency matrix $A_{<t} = \bigcup_{t'=1}^{t-1} \{(i, j)_{t'}\}$.

Multiple Edges Per Step To decode multiple edges per step, i.e. $|E_t| \geq 1$, we propose to repeatedly call f_{valid}, adding the returned edge to the adjacency matrix after each call, and stopping once the returned edge's score is below a pre-defined threshold τ.

[5] The reachability matrix R can be computed with batched matrix multiplication as $\sum_{k=1}^{t} A^k$ where t is the maximum path length; other methods could potentially improve speed.

Term-Based Extraction of Medical Information: Pre-Operative Patient Education Use Case

Martin Wolf[1], **Volha Petukhova**[2] **and Dietrich Klakow**[2]
Computational Linguistics[1], Spoken Language Systems[2], Saarland University, Germany
martinw@coli.uni-saarland.de;
{v.petukhova;dietrich.klakow}@lsv.uni-saarland.de

Abstract

The processing of medical information is not a trivial task for medical non-experts. The paper presents an artificial assistant designed to facilitate a reliable access to medical online contents. Interactions are modelled as doctor-patient Question Answering sessions within a pre-operative patient education scenario where the system addresses patient's information needs explaining medical events and procedures. This implies an accurate medical information extraction from and reasoning with available medical knowledge and large amounts of unstructured multilingual online data. Bridging the gap between medical knowledge and data, we explore a language-agnostic approach to medical concepts mining from the standard terminologies, and the data-driven collection of the corresponding seed terms in a distant supervision setting for German. Experimenting with different terminologies, features and term matching strategies, we achieved a promising F-score of 0.91 on the medical term extraction task. The concepts and terms are used to search and retrieve definitions from the verified online free resources. The proof-of-concept definition retrieval system is designed and evaluated showing promising results, acceptable by humans in 92% of cases.

1 Introduction

Nowadays, digital online services possess the dominant role delivering widely accessible applications at limited costs. For instance, recent technological advances make the provision of various eHealth services feasible. Using these applications, patients can stay informed searching content outside hospital business hours in a more convenient manner. A doctor, who conducts 120,000 - 160,000 interviews in the course of a 40-year career (Lipkin et al., 1995), meets then 'competent' patients who understand their medical needs and potential consequences of medical decisions. In the healthcare sector, language barriers and domain complexity may result in poor understanding of diagnosis, low compliance with recommendations, a significantly greater likelihood of a serious medical event and lower patient satisfaction (Bonacruz Kazzi and Cooper, 2003; Cohen et al., 2005; Pitkin Derose et al., 2009). The mainstream online services are therefore required to be reliable, accessible and to account for the diversity in individual needs, educational backgrounds, personal preferences, cognitive and physical limitations.

This paper addresses the needs in the reliable access to verified multilingual complex medical information. As a use case, we simulate pre-operative Question Answering (QA) sessions between doctors and patients. As a core part of these medical encounters, Patient Education Forms (PEFs) need to be filled in and the patient's informed consent signed. It is of chief importance that the forms are properly understood, medical procedures and risks are explained. PEFs contain many medical terms including those in Latin and as abbreviations. These terms have to be detected and corresponding definitions retrieved from available electronic medical documents. Although a number of biomedical entities recognition systems (Zhang and Elhadad, 2013; Björne et al., 2013; Sahu and Anand, 2017) and medical resources exist (Gurulingappa et al., 2010; Ohta et al., 2012), they are mostly built for English. We explore a language-agnostic approach to medical concepts mining based on the existing

Proceedings of Recent Advances in Natural Language Processing, pages 1346–1355,
Varna, Bulgaria, Sep 2–4 2019.

de-facto standard terminologies and dictionaries, and the collection of the corresponding German seed terms in a distant supervision setting. The extracted concepts and terms are used to search and retrieve definitions from the verified online free text sources. The proof-of-concept definition retrieval system is designed and evaluated.

The paper is structured as follows. In Section 2, we provide an overview of the state-of-the-art in Biomedical Named Entity Recognition (BM-NER). In Section 3, we discuss the conceptual design of a medical domain. Section 4 defines the medical term extraction task, assesses various resources for medical information extraction and presents the overall QA system architecture. Section 5 proposes the experimental design by specifying the collected and simulated data, and discusses the obtained results. Finally, Section 6 summarizes our findings and outlines directions for the future research and development.

2 Biomedical Named Entity Recognition

In 1995, the 6th Message Understanding Conference (MUC-6) focused on the Information Extraction (IE) from unstructured textual data (Grishman and Sundheim, 1996) and defined the Named Entity Recognition and Classification (NERC) task, see (Nadeau and Sekine, 2007) for a comprehensive overview. The relevant entities comprised names of persons, organizations and locations defined as ENAMEX (Entity Name Expression), extended later with TIMEX (Time Expressions) and NUMEX (Numerical Expressions).

In the early 2000s, the interest in bioinformatics lead to enriching the categories with concepts from biomedical domains focusing on the recognition of biological and genetic terms, disease and drug names and other medical or clinical entities (Settles, 2004; Shen et al., 2003). Biomedical named entity recognition is a key step in biomedical language processing.

Early (BM-)NER approaches were largely *rule-based* detecting entities based on the observed contextual and orthographic patterns. Such systems are especially useful if no or little training examples are available (Sekine and Nobata, 2004), are often straightforward to implement, suited for the entity classes or domains where the regularities in orthography or morphology can be exploited, and have other important advantages (Chiticariu et al., 2013). Although they achieve a rather high precision, recall is often low as the rule sets are rarely exhaustive. `AbGene` system of Tanabe and Wilbur (2002) uses a POS tagger extended to include gene and protein names as tag types. The system was trained on the manually labelled biomedical text. In its second iteration, it applies manually defined post-processing rules.

Another successful approach underlies the so-called *dictionary-based* systems. Here, the decision whether an entity is of an interest is made by matching against the entries in a dictionary, i.e. gazetteer or word list. To expand the coverage, linguistic methods (e.g. stemming or lemmatization), as well as fuzzy or exact matching strategies are used. `cTakes` of Savova et al. (2010) is an open-source information extraction tool from Electronic Health Records (EHR) which NER component is based on a dictionary look-up approach.

Dictionaries are also used supplementary to *machine learning* approaches (Tsuruoka and Tsujii, 2003), which are particularly useful if there is a high variability in entities observed. *Supervised* models like Hidden Markov Models (Zhou and Su, 2002), Support Vector Machines (Björne et al., 2013), Conditional Random Fields (Settles, 2004) and Neural Networks (Sahu and Anand, 2017) are reported to show the state-of-the-art performance. These approaches rely on large amounts of the annotated training data. To perform BM-NER some resources are created: the NCBI Disease Corpus (Doğan et al., 2014), the GENIA corpus (Kim et al., 2003) for molecular biology, the i2b2[1] corpus of clinical notes. The data for languages other than English is still an issue. Techniques which allow to automatically generate labelled training data like bootstrapping and distant supervision (Mintz et al., 2009) methods are proposed to build models in *semi-supervised* or *weakly supervised* way. For example, Dembowski et al. (2017) extract word lists from Wikipedia to label the data for an NER model training. The trained classifier outperforms the simple dictionary baseline.

Unsupervised approaches do not require any labelled training data, but rely on external resources like knowledge-bases or semantic nets (Alfonseca and Manandhar, 2002), lexical patterns (Evans and Street, 2003), and distributional semantics. Zhang and Elhadad (2013) applied a distributional semantics method to clinical notes and biological texts. The final system yields competitive results

[1] https://www.i2b2.org/NLP/DataSets/

Category	Frequency (in %)	Seed Terms Examples	
		German	English
Body-organ	19.6	Bronchien, Kopf	bronchia, head
Body-related	6.5	Atem, Hören	breath, hearing
Condition	4.0	gesund, schläfrig	healthy, sleepy
Disease	3.1	Hepatitis, Schlaganfall	hepatitis, stroke
Drug	8.0	Aspirin, Schlafmittel	aspirin, sleeping pills
Effect	2.9	Wärmegefühl	warm sensation
Institution	0.5	Intensivstation, Aufwachraum	intensive care unit, recovery room
Instrument	7.8	Nadel, Larynxmaske	needle, laryngeal mask
Person	2.5	Arzt, Pflegepersonal	doctor, nursing staff
Procedure	17.6	Eingriff, Narkose	intervention, narcosis
Procedure-related	2.2	intravenös, operativ	intravenous, operative
Purpose	0.7	muskelentspannend, Schmerzausschaltung	muscle relaxing, pain relief
Symptom	21.9	Atemnot, Juckreiz	shortness of breath, itching
Misc	2.7	medizinisch, peripher	medical, peripheral

Table 1: The taxonomy and distribution (relative frequencies, in %) of semantic concepts categories illustrated with examples of German and English seed terms.

on the i2b2 biomedical dataset of clinical notes and the GENIA corpus of biological literature, and outperforms the dictionary-matching baseline. This approach incorporates the collection of *seed terms*. The seed term sets are gathered from external terminologies and grouped into entity classes that represent the domain the best. For a QA application, it means that the classes of domain-specific semantic concepts can be used to generate signature vectors and the semantic similarity with the signature vectors of the answer candidates can be computed for retrieval and ranking. The concept classes can be also translated into the Expected Answer Types (EATs) to query the structured or unstructured data to retrieve an answer in a supervised or rule-based way.

3 Conceptual Domain Modelling

To facilitate an accurate information extraction from and reasoning with large amounts of (un-)structured data, it is important to specify and model real world entities and relations between them. This knowledge is often represented as ontologies, terminologies with semantic concepts groupings and taxonomic relations between them, and semantic networks. In many knowledge-based QA systems, high level semantic representations are used to query databases or other types of structured data. For example, Wilensky et al. (1988) developed the *Berkeley Unix Consultant*, for the domain related to the UNIX operating system where questions are analysed and transformed into an internal representation which are used to generate hypothesis about the user's information needs. A knowledge-based QA system as used by Apple Siri[2] and Wolfram Alpha[3] also first builds a query representation and then maps it to structured data like ontologies, gazeteers, etc. The Watson a DeepQA system of IBM Research (Ferrucci et al., 2010) incorporates content acquisition, question analysis, hypothesis generation, etc. Inside the hypotheses generation, it relies on NE detection, triple store and reverse dictionary look-up to generate candidate answers.

Alternative approaches advocate that intelligent behaviour is a result of the processing of stimuli rather than symbols. Sub-symbolic modelling is based on uninterpreted input and distributed representations by dynamic connection weights, e.g. Artificial Neural Networks comprise interconnected networks of simple processing units. In QA, so-called Neural Question Answering currently dominates the field, see e.g. (Weston et al., 2015). Based on neural network models, the systems involve relatively small pipeline, but require a significant amount of annotated data.

Recently, a number of approaches have been devised proposing a combination of symbolic and sub-symbolic processing. It has been shown that fundamental to human cognitive abilities is the capacity to process *concepts* which emerge from a distributed connectionist representation at a lower level where stimuli are processed, and are combined to form symbolic structures at the highest level to support understanding and reasoning, see e.g. (Gärdenfors, 2004). For a QA system, it implies that questions understanding and answers retrieval can be modelled at a higher level of semantic abstraction mining key concepts from available

[2] http://www.apple.com/ios/siri/
[3] www.wolframalpha.com

(e.g. medical) taxonomies, mapping them to (parts of) EATs, which on their turn can be used to perform data-driven entity recognition and semantic relation classification tasks.

Over the past few years, the community proposed different approaches to generate taxonomies which range from flat lists of mutually exclusive categories to hierarchical taxonomies with coarse categories subdivided into fine-grained classes. For example, Srihari and Li (1999) used the defined MUC NER categories to derive their taxonomy. Kim et al. (2001) created a taxonomy for semantic categorization of questions and candidate answers based on the WordNet categories. Chuang and Chien (2003) clustered queries with similar information needs into groups. Hereby, higher ranked results from web search engines were used as features to create multi-way trees via hierarchical clustering.

Chilton et al. (2013) applied crowdsourcing techniques to generate taxonomies based on three Human Intelligence Tasks (HITs) where different groups of participants: (1) generate a category for each shown item; (2) decide which items and the generated categories fit the best; and (3) decide for each category whether an item fits in it or not.

The conceptual complexity of medical domains, can make it difficult for users of information systems to comprehend and interact with the knowledge embedded in those systems (Wickens et al., 1998). To give an example, the Unified Medical Language System (UMLS)[4] integrates over 2 million names for 900 000 concepts from more than 60 families of biomedical vocabularies, as well as 12 million relations among these concepts. The UMLS semantic network reduces the complexity of this construct by grouping concepts according to the semantic types that have been assigned to them McCray et al. (2001). Medical knowledge bases, ontologies, standard terminologies and lexicons can facilitate many NLP and AI tasks, and are exploited in this work.

4 Methodology

4.1 Medical Term Extraction: The EAT Taxonomy and Seed Terms

Ideally, doctors want to meet competent patients who understand their medical needs and potential consequences of medical decisions, so that doctors can be sure that the patient's consent is well-informed. It is a common practice nowadays that before meeting a doctor who will plan an operative medical procedures, patients often have to fill in Patient Education Forms (PEF) to understand procedures and risks involved, and ask their doctors more precise and in-depth questions. To model system's QA behaviour for our use case, the reference PEF[5] was analysed to extract the domain-specific semantic concepts and grouped them into 14 categories using the UMLS semantic network. The form consists of 1,886 tokens, from which 448 tokens (261 unique tokens) were identified as medical entities. Thus, in theory a patient can ask 261 question to the system requesting additional information or explanation. The resulted taxonomy (Table 1) was used to annotate 64 PEFs in German, cluster dictionary terms and to define the EAT to classify questions and retrieve definitions for the system's answers.

Source	Trustworthy	Available	Accessible
Pschyrembel	+	(+)	-
Wikipedia	(+)	+	+
Wiktionary	(+)	+	+
Roche	+	(+)	-
MedlinePlus	+	+	(+)

Table 2: Overview of the assessed medical online resources. (+ stands for 'yes', – for 'no', and (+) for 'partially'. see Section 4.2

The semantic categories were populated with relevant seed terms. For this, the dictionary was created using a medical word list available on Wiktionary[6]. We first matched the PEFs seed terms to the lexicon entries using dictions (i.e. case, number), lemma, and/or stem, and enriched it further with synonyms, hyponyms, and hypernyms using the Wiktionary relations.

To improve the coverage of the proposed term set, we augmented the list with entities from available online unstructured medical data in a distant supervision setting. The classifiers, Naive Bayes (NB) and Multinominal Naive Bayes (MNB), were trained operating on different types of lexical and linguistic (e.g. words, lemmas, stems and as-

[4] https://www.nlm.nih.gov/research/umls/

[5] The form in English, German, French, Italian, Serbian and Turkish can be found here: https://www.oegari.at/arbeitsgruppen/arge-praeoperatives-und-tagesklinisches-patientenmanagement/937.html

[6] https://de.wiktionary.org/wiki/Verzeichnis:Deutsch/Medizin An XML dump of the German Wiktionary was used: https://dumps.wikimedia.org/dewiktionary/20181001/

signed POS tags), orthographic (e.g. capitalization information and word length), morphological (e.g. inflections) and contextual features where preceding and following words as well there POS tags are encoded as bi- and tri-grams.

4.2 Resources for Definition Retrieval

Users of medical QA systems want to get medical information which is accurate, not misleading or fake. The verified sources, in our view, need to fulfil the following criteria:

1. **Trustworthiness**: the resource is accepted as medical information source;
2. **Availability**: the resource is distributed under non-exclusive license agreements with no costs associated with its use;
3. **Accessibility**: the resource can be crawled from the website or there are APIs available.

For example, the **Pschyrembel**[7] is the most referred clinical German database. Although there exists a free online test version, a license is required for a complete access. The website can not be crawled. Pschyrembel is an excellent source for medical terminology even for laymen, since the definitions are very well explained and concise, include synonyms and an English translation.

Wikipedia[8] and **Wiktionary**[9], published by the *Wikimedia Foundation*[10], are freely available databases. Generally, the Wikimedia databases are good information sources, however can not be considered as trusted medical resources. Both resources are available in many different languages enabling terms alignment and translation. Wiktionary definitions are mostly one-sentence short explanations capturing the term meaning in general, whereas Wikipedia often provides long and detailed descriptions of multiple related aspects. There are interfaces available to access the data.

Other surveyed medical resources are **Roche Lexikon Medizin**[11] for German and **MedlinePlus**[12] of the *US National Institute of Health* for English. However, their trustworthiness comes with a price, see Table 2 for a comparison.

Dataset	#texts	#tokens	#NE
Training set			
PEFs	64	55,280	7,333
Wikipedia articles	6,865	4,017,388	262,337
Full training dataset	6,929	4,072,668	263,568
Test set			
PEF	1	1,886	448

Table 3: Training and test datasets.

4.3 QA System Architecture

The designed QA system consists of three core modules performing pre/post-processing, term extraction and definition retrieval. A general overview of the system is depicted in Figure 1.

Patient's input and available resources are *preprocessed*, e.g. tokenized, segmented; language models and (multilingual) word embeddings are computed. As output, vectors representing questions and documents are generated.

The next step is concerned with medical entities recognition. The *medical term extractor* exists in two versions (Section 5.2.1). The dictionary-based (DB) extractor annotates tokens depending on their presence in the dictionary. The module takes different parameters specific for the matching process such as word, dictions, lemma, stem, and case-(in-)sensitive matching. The machine learning (ML) classifier operates on the computed features discussed above, applies the trained prediction models and extracts the relevant entities.

To query online resources either unstructured online contents or available medical knowledge bases, queries are formulated containing the EAT concepts extended with the collected (multilingual) seed terms. The expanded queries are also transformed into the signature vectors to measure the semantic similarity with the previously computed document vectors. Multiple *definitions* can be *retrieved* and ranked. For the generation of system's answers, definitions can be summarized (Hardy et al., 2002), simplified or lexically/syntactically adapted, see e.g. (Wang et al., 2016).

5 Experiments

5.1 Data

The data used in our IE experiments is of two types: (1) dictionaries, and (2) medical free texts as training and test data for classifiers.

[7] https://www.pschyrembel.de/

[8] https://de.wikipedia.org/wiki/Wikipedia:Hauptseite

[9] https://de.wiktionary.org/wiki/Wiktionary:Hauptseite

[10] https://wikimediafoundation.org/

[11] https://www.roche.de/lexikon/index.htm?loc=www.roche.de&content=/lexikon/suche.html

[12] https://medlineplus.gov/healthtopics.html

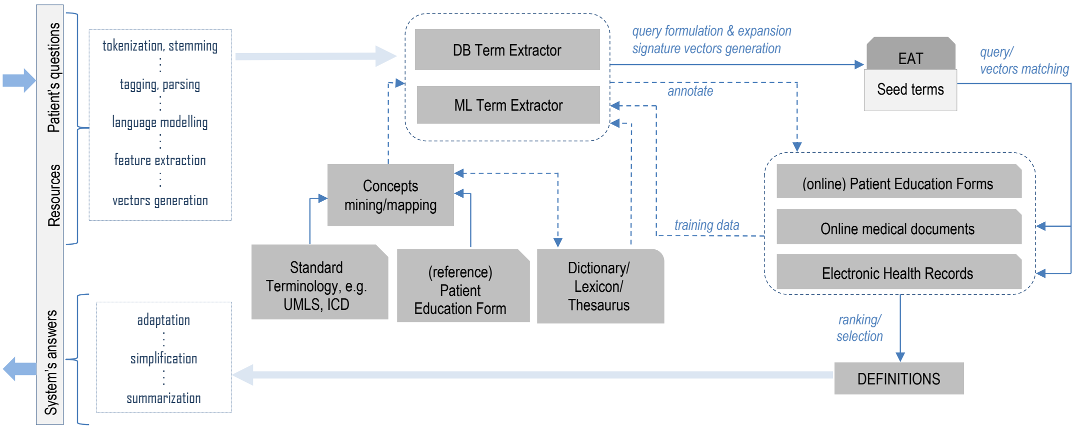


Figure 1: Proposed QA system architecture. From left to right: patient's questions and available medical documents are processed. Medical terms are extracted using available concepts taxonomies, terminologies, medical dictionaries and are learned from the annotated data. Concepts and terms are mapped to the EAT to formulate and expand the query. Signature vectors for questions and answer candidates are computed. The verified (un-)structured data/knowledge sources are queried to retrieve definitions which are ranked and post-processed before returning to the patient.

Dictionary comprises the initial word list of 2,035 Wiktionary medical term entries.[13] The word list covers different fields of medical work from general medicine to dentistry, and contains a mix of Latin and German names of medical procedures, tools and events. We augmented this list with the Wiktionary technical terms[14] and Wikipedia medical terms[15]. The resulting cleaned term list comprises 12,711 terms, see Table 4.

Word list	#terms	P	R	F1
Wiktionary medical	2035	0.947	0.120	0.213
Wiktionary medical + technical	2485	**0.949**	0.125	0.220*
Wikipedia medical	11041	0.928	0.285	0.436*
Complete list	12711	0.915	**0.312**	**0.465***

Table 4: Results of word list experiments. Here and in the further Tables, P stands for precision, R - for recall, F1 - for F-scores. *differs significantly from the baseline obtained on the smallest word list according to the McNemar's test, $\alpha < 0.05$

The **training data** consists of the 64 online PEFs and 6,865 Wikipedia articles[16] extracted with the Wikipedia term list. The test data as described above was constructed from a single PEF, see Table 3 for details.

5.2 Results

5.2.1 Medical NE Recognition

We conducted: (1) the dictionary-based, and (2) machine learning NER experiments. For evaluation, the standard metrics of precision, recall, and F-scores were used. The McNemar's tests were performed to measure statistical significance (McNemar, 1947).[17]

Relation depth: synonym,hypernym,hyponym	P	R	F1
0,0,0	0.92	0.31	0.47
0,0,1	0.93	0.41	0.57*
1,1,1	0.85	0.48	**0.61***
1,1,2	0.85	0.48	**0.61***
3,2,2	0.63	0.48	0.55*
2,2,3	0.76	0.48	0.59*
3,1,1	0.63	0.48	0.55*

Table 5: Results of the dictionary-based experiments: the assessment of the relation depth levels. *differs significantly from the baseline 0,0,0 setting according to the McNemar's test, $\alpha < 0.05$

Dictionary-based (DB) term recognition experiments were performed in two steps. First, the dictionary was gradually expanded to improve its coverage. We evaluated the performance using the word lists compiled from *Wiktionary* medical terms and technical medical terms, *Wikipedia* medical terms and combinations of those. Further, we experimented with the depth of Wik-

[13] https://de.wiktionary.org/wiki/Verzeichnis:Deutsch/Medizin

[14] https://de.wiktionary.org/wiki/Verzeichnis:Deutsch/Medizin/Fachwortliste

[15] https://de.wikipedia.org/wiki/Portal:Medizin/Index

[16] The Wikipedia articles dump of September 10, 2018 https://dumps.wikimedia.org/dewiki/20181001/ was used.

[17] The null hypothesis for our tests states that two algorithms, applied to the same data, retrieve the same results. The test statistic has a distribution of χ^2 with one degree of freedom. A significance level of $\alpha = 0.05$ was set.

Setting	P	R	F1
baseline	**0.855**	0.477	0.611
+diction	0.810	0.635	0.712*
+lemma	0.787	0.666	0.721*
+stem	0.807	0.641	0.715*
-case	0.847	0.481	0.614
+diction -case	0.802	0.639	0.711*
+lemma -case	0.789	0.673	**0.726***
+stem -case	0.713	0.657	0.684*
+diction+lemma +stem-case	0.703	**0.679**	0.691*
Naive Bayes	0.639	0.670	0.653
Multinominal Naive Bayes	**0.853**	**0.859**	**0.851***

Table 6: Results of the **dictionary-based** experiments assessing of various matching strategies with relation depth 1,1,1 (best results) and the **classification** performance. *differs significantly from the baseline according to the McNemar's test, $\alpha < 0.05$

tionary *synonyms*, *hypernyms* and *hyponyms* relations. For example, a depth of 2 in the hypernym relation means that the hypernym of the word, and the hypernym of the hypernym is added to the dictionary.

Subsequently, different matching strategies were tested tuning parameters like *lemma*, *stem* and different *inflections* types and combinations of those. The matching was also conducted in *case-sensitive* and *case-insensitive* setting.

From the results presented in Table 4 can be observed that larger dictionaries result in a better system performance in terms of higher F-scores. The expanded dictionary coverage leads to a higher recall, as more relevant terms can be found. The precision, by contrast, drops slightly when larger dictionaries are used, due to the larger amount of false positives. For our use case, we assume that the system's acceptance will depend on its ability to explain as many terms as possible than missing many relevant of them.

In the second set of experiments, we assessed the impact of the relation depth on the term extraction performance. As Table 5 shows that encoding the relation information of 1,1,1 and 1,1,2 types resulted in the best performance (F-scores of 0.611). We concluded that recall increases with the increased relation depth. Deeper relations, however, generate more out-of-domain terms causing the precision drop. For example, the further up the hypernym relation gets, the more general the terms become. Considering synonyms of all word senses introduce further noise in the training data, e.g. the German word 'Nase / nose' is also a fish and the synonym list does not only contain 'Riechorgan / olfactory organ' or 'Zinken / beak', but also 'Näsling / common nase', which is a kind of carp and is unlikely to occur in PEFs.

Features	P	R	F1
word	0.875	0.879	0.876
+POS	0.876	0.880	0.877
+Suffix	0.879	0.882	0.879*
+Prefix	**0.882**	**0.885**	**0.883***
+nextBigramPOS	0.877	0.880	0.877
+prevBigramPOS	0.879	0.882	0.880*
Best features	**0.909**	**0.909**	**0.909***

Table 7: Classification performance on different feature sets. Note: only features that improved the previously obtained results are reported here. *differs significantly from the word baseline according to the McNemar's test, $\alpha < 0.05$

In the final dictionary experiments, we assessed the impact of the word-based matching strategies. The experiments showed that using lemmas and case-insensitive strategies yielded the best results. The best overall F-score of 0.726 was achieved using the complete Wikipedia and Wiktionary word list, a relation depth set to 1,1,1 for synonyms, hypernyms, and hyponyms respectively, lemmatising and ignoring capitalizations in the input. The performance of the best dictionary-based extractor outperforms the Wiktionary medical baseline by broad margins, compare Tables 4 and 6.

For our **machine learning** (ML) experiments, we generated the training data using the distant supervision approach and the best version of the dictionary-based extractor. The MNB classifier outperformed the NB classifier by broad margins, achieving F-scores of 0.85 comparing to 0.65, consider two last rows of Table 6.

Finally, the impact of different feature combinations on the classifier performance was evaluated. For this, each feature was tested individually in combination with the *word* feature. Results showed that only few features contributed to the improvement of the overall classification performance, see Table 7 for an overview. The best feature combination was found to be a combination of word features and POS information of previous, current and next word, as well as the morphological information concerning prefixes and suffixes.

Our experiments showed that the built classifiers outperformed the dictionary-based extractors. The overall F-scores improvement of 0.183 was achieved. More importantly, the recall was drastically improved from 0.236 (dictionary baseline)

Dictionary-based NE recognition		Machine-learning based NE recognition	
Configuration	**F1**	**Configuration**	**F1**
Baseline: Wiktionary medical data	0.213	Baseline: PEF training data	0.851
Best Lexicon: Wikipedia & Wiktionary data	0.465	Best training data: PEF & Wikipedia articles	0.876
Best relation depth: 1,1,1	0.611	Best feature pair: word+prefix	0.883
Best matching strategy: +lemma, -case	0.726	Best feature combination: word, +POS trigram, +inflexion	0.909

Table 8: Summary of the best obtained results for medical entities extraction.

Source	**# retrieved definitions** (in % of all PEF terms)	**# accepted definitions** (in % of all retrieved)
Wiktionary	133 (51.0)	123 (92.4)
Wikipedia	124 (47.5)	93 (75.0)
Both resources	134 (51.3)	123 (92.4)

Table 9: Coverage and quality of the retrieved Wiktionary and Wikipedia definitions.

and from 0.673 (best dictionary-based system) to 0.909 of the best classification model. Table 8 summarizes the key experimental results.

5.2.2 Definition Retrieval

The proof-of-concept definition retrieval was implemented using Wiktionary and Wikipedia resources that contain clear understandable definitions and are available in many different languages. The methods developed for German and English can be used for many other languages.

On a technical note, the Wikipedia and Wiktionary APIs are available to retrieve the summary part of the corresponding Wikipedia article, and the sense of Wiktionary. **Coverage** of the reference PEF medical terms and the **quality** of the retrieved definitions were evaluated.

Both resources covered 51.3% of the annotated PEF terms: 47.5% for Wikipedia and 51.0% for Wiktionary. The retrieved definitions were evaluated on their acceptability: whether the definition is *correct*, *clear* and *sufficient*. The evaluation was performed by three human raters. Out of the 133 Wiktionary definitions, 123 (92.4%) definitions were evaluated as acceptable: wording and sentence structure were simple, i.e. not containing other complex terminology and more than one subordinate clause. The retrieved Wikipedia definitions were, by contrast, evaluated as less acceptable: multi-sentence definitions are frequent with complex sentence structures using other medical expressions. The assessment results for the definition coverage and quality from the respective sources can be found in Table 9.

6 Conclusions and Future Work

In this paper, we addressed medical terms and definitions extraction simulating Patient Education QA sessions. We assessed two core methods to medical terms extraction: based on the standard medical terminologies and available dictionaries, and applying a machine-learning approach to extract German seed terms in a distant supervision setting expanding the system's coverage. We also proposed criteria to test and select medical resources for a QA application. A proof-of-concept definition retrieval systems was implemented and evaluated. The work contributes to a closed-domain QA system design to facilitate access to verified multilingual medical information.

The baseline DB and ML-based extraction techniques are assessed considering various dictionaries/datasets sizes, word matching strategies and different feature combinations. The distant supervision is a viable method to overcome the shortage of manually annotated monolingual data and can be successfully applied to automatically and productively generate large sets of the annotated multilingual seed terms. The proposed term-based information extraction opens perspectives for multi- and cross-lingual QA application design. The concepts categories populated with terms in multiple languages enable cross-lingual mappings. If the language is available on Wiktionary, the relational connections can be used as well.

Our future work will pursue multiple goals. To improve the quality, a larger annotated corpus for German will be collected. Larger data sets will also allow to train machine learning classifiers on noisy labelled data. Different search and retrieval methods will be explored, i.e. based on machine translation, cross-lingual language models and multilingual embeddings. In particular, we are interested in training new neural networks in multi- and cross-lingual term extraction and definition retrieval settings. We also plan to invest into the adaptation and simplification of the retrieved definitions where the complex medical terms will be translated into common terms. This can be achieved in a dictionary-based setting augmenting seed terms collections, but also defining the task as a machine translation one.

References

Enrique Alfonseca and Suresh Manandhar. 2002. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st international conference on general WordNet, Mysore, India*. pages 34–43.

Jari Björne, Suwisa Kaewphan, and Tapio Salakoski. 2013. Uturku: drug named entity recognition and drug-drug interaction extraction using svm classification and domain knowledge. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. volume 2, pages 651–659.

G Bonacruz Kazzi and Carolyn Cooper. 2003. Barriers to the use of interpreters in emergency room paediatric consultations. *Journal of paediatrics and child health* 39(4):259–263.

Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, pages 1999–2008.

Laura Chiticariu, Yunyao Li, and Frederick R Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the 2013 conference on empirical methods in natural language processing*. pages 827–832.

Shui-Lung Chuang and Lee-Feng Chien. 2003. Automatic query taxonomy generation for information retrieval applications. *Online Information Review* 27(4):243–255.

Adam L Cohen, Frederick Rivara, Edgar K Marcuse, Heather McPhillips, Robert Davis, et al. 2005. Are language barriers associated with serious medical events in hospitalized pediatric patients? In *peds*. volume 2005:0521, page 116.

Julia Dembowski, Michael Wiegand, and Dietrich Klakow. 2017. Language independent named entity recognition using distant supervision. In *Proceedings of Language and Technology Conference (LTC)*.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics* 47:1–10.

Richard Evans and Stafford Street. 2003. A framework for named entity recognition in the open domain. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP* 260(267-274):110.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine* 31(3):59–79.

Peter Gärdenfors. 2004. *Conceptual spaces: The geometry of thought*. MIT press.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. volume 1.

Harsha Gurulingappa, Roman Klinger, Martin Hofmann-Apitius, and Juliane Fluck. 2010. An empirical evaluation of resources for the identification of diseases and adverse effects in biomedical literature. In *2nd Workshop on Building and evaluating resources for biomedical text mining (7th edition of the Language Resources and Evaluation Conference)*.

Hilda Hardy, Nobuyuki Shimizu, Tomek Strzalkowski, Liu Ting, Xinyang Zhang, and G Bowden Wise. 2002. Cross-document summarization by concept classification. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 121–128.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl_1):i180–i182.

Soo-Min Kim, Dae-Ho Baek, Sang-Beom Kim, and Hae-Chang Rim. 2001. Question answering considering semantic categories and co-occurrence density. In *AUTHOR Voorhees, Ellen M., Ed.; Harman, Donna K., Ed. TITLE The Text REtrieval Conference (TREC-9)(9th, Gaithersburg, Maryland, November 13-16, 2000). NIST Special Publication. INSTITUTION National Inst. of Standards and Technology, Gaithersburg, MD.; Advanced Research Projects Agency (DOD), Washington, DC.*. Citeseer, page 261.

Mack Lipkin, Richard M Frankel, Howard B Beckman, Rita Charon, and Oliver Fein. 1995. Performing the interview. In *The medical interview*, Springer, pages 65–82.

Alexa T McCray, Anita Burgun, and Olivier Bodenreider. 2001. Aggregating umls semantic types for reducing conceptual complexity. *Studies in health technology and informatics* 84(0 1):216.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12(2):153–157.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*. Association for Computational Linguistics, pages 1003–1011.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Tomoko Ohta, Sampo Pyysalo, Jun'ichi Tsujii, and Sophia Ananiadou. 2012. Open-domain anatomical entity mention detection. In *Proceedings of the workshop on detecting structure in scholarly discourse*. Association for Computational Linguistics, pages 27–36.

Kathryn Pitkin Derose, Benjamin W Bahney, Nicole Lurie, and José J Escarce. 2009. Immigrants and health care access, quality, and cost. *Medical Care Research and Review* 66(4):355–408.

Sunil Kumar Sahu and Ashish Anand. 2017. Unified neural architecture for drug, disease and clinical entity recognition. *arXiv preprint arXiv:1708.03447* .

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association* 17(5):507–513.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. In *LREC*. Lisbon, Portugal, pages 1977–1980.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*. Association for Computational Linguistics, pages 104–107.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. 2003. Effective adaptation of a hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*. Association for Computational Linguistics, pages 49–56.

Rohini Srihari and Wei Li. 1999. Information extraction supported question answering. Technical report, CYMFONY NET INC WILLIAMSVILLE NY.

Lorraine Tanabe and W John Wilbur. 2002. Tagging gene and protein names in biomedical text. *Bioinformatics* 18(8):1124–1132.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2003. Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13*. Association for Computational Linguistics, pages 41–48.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* .

Christopher D Wickens, Sallie E Gordon, Yili Liu, et al. 1998. *An introduction to human factors engineering*. Longman New York.

Robert Wilensky, David N Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. 1988. The berkeley unix consultant project. *Computational Linguistics* 14(4):35–84.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics* 46(6):1088–1098.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 473–480.

A Survey of the Perceived Text Adaptation Needs of Adults with Autism

Victoria Yaneva, Constantin Orăsan, Le An Ha and Natalia Ponomareva
Research Institute in Information and Language Processing,
University of Wolverhampton, UK
{v.yaneva, c.orasan, ha.l.a, n.ponomareva}@wlv.ac.uk

Abstract

NLP approaches to automatic text adaptation often rely on user-need guidelines which are generic and do not account for the differences between various types of target groups. One such group are adults with high-functioning autism, who are usually able to read long sentences and comprehend difficult words but whose comprehension may be impeded by other linguistic constructions. This is especially challenging for real-world user-generated texts such as product reviews, which cannot be controlled editorially and are thus in a stronger need of automatic adaptation. To address this problem, we present a mixed-methods survey conducted with 24 adult web-users diagnosed with autism and an age-matched control group of 33 neurotypical participants. The aim of the survey is to identify whether the group with autism experiences any barriers when reading online reviews, what these potential barriers are, and what NLP methods would be best suited to improve the accessibility of online reviews for people with autism. The group with autism consistently reported significantly greater difficulties with understanding online product reviews compared to the control group and identified issues related to text length, poor topic organisation, identifying the intention of the author, trustworthiness, and the use of irony, sarcasm and exaggeration.

1 Introduction

The aim of automatic text adaptation (also known as automatic text simplification) is to make the meaning of texts more accessible to specific target groups such as language learners or people with cognitive disabilities. To achieve this, the development of automatic systems is driven by guidelines that describe the needs of the target group, but very often these guidelines are generic. For example, one of the most authoritative sources of such guidelines for people with cognitive disabilities, the *European Guidelines for the Production of Easy-to-Read Information* (Freyhoff et al., 1998), lists requirements that fit the profile of people with moderate to severe comprehension deficits, but not those of more highly able individuals. As a result, the majority of text simplification strategies aim to reduce sentence and word complexity, while approaches to other aspects of text adaptation (e.g. clarifying the opinion of the author or strengthening the text organization) receive less attention.

In this paper we address this issue by providing insight into the needs of a specific subgroup, people with high-functioning autism, who are known to be able to read and comprehend complex texts, but may struggle with specific aspects of their comprehension. For example, readers with high-functioning autism are usually able to cope with the meaning of complex words but are known to struggle with non-literal language or with combining the meaning of individual text components into a meaningful whole (see Section 2).

These difficulties are particularly challenging when interacting with texts that were not controlled editorially and may create barriers for people with autism to interact with the web, make informed decisions, and being an active part of the economy. One such type of text is the user feedback on goods and services, which is increasingly used to guide decision making in many aspects of life, from travel, entertainment, and shopping to education, social care, and policy making (Eynon and Margetts, 2007). This feedback usually comprises numerical ratings and written reviews submitted by users. A survey of online shoppers in the UK showed that product reviews have the greatest influence on purchasing decisions, greater than that of expert reviews or advice from friends (Fretwell et al., 2013). Unlike other types of Web content, whose accessibility can be controlled editorially, product reviews in user feedback are produced spontaneously by a large, dynamic, and heterogeneous population of writers. Unsurprisingly, this feedback, which represents a large proportion of content available on the Web, is of varying levels of accessibility.

The paper presents a mixed-methods survey conducted with 24 adult web-users diagnosed with autism and an age-matched control group of 33 neurotypical participants. The aim of the survey was to identify whether the group with autism: i) experienced any barriers when reading online reviews, ii) what these potential barriers were, and iii) what automatic methods would be best suited to improve the accessibility of online reviews for people with autism. To the best of our knowledge, this is the first time when the perception of online product reviews has been investigated in terms of its accessibility for people with autism.

The next section presents information on the reading difficulties of people with autism.

2 Background

2.1 Autism Spectrum Disorder and Reading

Autism Spectrum Disorder (ASD) is a developmental disorder with neural origin characterised by impairment in communication and social interaction (American Psychiatric Association, 2013) and is known to affect about 1 in 100 people in the UK (Brugha et al., 2011). Language comprehension difficulties in autism cover phenomena such as *difficulties in syntax processing of long sentences* (Whyte et al., 2014), *resolving ambiguity in meaning* (Happe, F., and Frith, U, 2006), and *identifying pronoun referents* (O'Connor and Klein, 2004), as well as having difficulties in *figurative language comprehension* and *making pragmatic inferences* (MacKay and Shaw, 2004). These difficulties, together with the specific cognitive profile of individuals with autism (e.g., differences in the Theory of Mind (Baron-Cohen, 2000)) may lead to secondary issues such as challenges with identifying author intent and subtler nuances of meaning. In addition, web users with autism have been consistently shown to have different information searching strategies when processing web pages (Eraslan et al., 2017; Yaneva et al., 2018; Eraslan et al., 2019; Yaneva et al., 2019), which relate to differences in visual attention. As a result of these difficulties, information contained in online user feedback can be less accessible for people with autism.

2.2 Automatic Text Adaptation for Adults with Autism

In terms of systems aimed at making text more accessible for autistic individuals who are fairly able, the *OpenBook* tool[1] is the most comprehensive existing system to date. The tool provides semi-automatic conversion of text documents by reducing syntactic complexity and disambiguating meaning by resolving pronominal reference, performing word sense disambiguation and detecting conventional metaphors (Evans et al., 2014; Orăsan et al., 2018), with some initial efforts towards concept substitutions for images (Barbu et al., 2015). As part of the research project, the tool was evaluated together with end-users with ASD who were shown to find the adapted texts more accessible than the originals. Nevertheless, a major impediment for the automatic evaluation of such systems is the limited amount of user-evaluated data. To the best of our knowledge, the only available resources containing a limited amount of such data are the ASD corpus (Yaneva et al., 2016a; Yaneva, 2016), followed by a corpus of easy-to-read documents that were specifically developed for people with cognitive disabilities (Yaneva et al., 2016b) [2]. Constrained by these limitations, some approaches propose to automatically evaluate text simplification systems for people with autism in terms the change in readability of the generated sentences (Evans et al., 2014; Štajner and Saggion, 2013), the incorporation of user-evaluated data into larger corpora (Yaneva et al., 2017), or the use of corpora containing texts for children and language-learners (Štajner et al., 2014). Therefore, very little is known about the perceptions of adults with high-functioning autism on the usefulness of specific simplification strategies.

In the following sections we present a survey on the perceptions of adults with high-functioning autism on the accessibility of user reviews.

3 Data Collection

This section presents the way the survey responses were collected.

[1] http://www.openbooktool.net/

[2] Note, however, that the latter is not targeted at readers with high-functioning autism

Question	Possible Answers
1. Do you read online reviews to determine whether a product or service is good or bad?	a) Every time b) Very often c) Sometimes d) Rarely e) Never
2. How many product reviews do you read before coming to a decision?	a) None b) Less than 10 c) Between 10 and 20 d) More than 20 e) Most of the available reviews
3. In general, do you find understanding product reviews:	a) Very easy b) Easy c) Medium d) Difficult e) Very difficult
4. In general, how do you find understanding whether the author approves or disapproves of the described product?	a) Very easy b) Easy c) Medium d) Difficult e) difficult
5. Have you ever felt confused about the meaning of an online review because the author used irony or sarcasm?	a) Very often b) Often c) Sometimes d) Rarely e) Never
6. Do you think that having a summary of the main points of all reviews would be:	a) Very helpful b) Helpful c) Moderately helpful d) Not very helpful e) Not helpful at all
7. Do you feel that there are certain barriers for you with regards to understanding product reviews?	a) Most of the time b) Often c) Sometimes d) Rarely e) Never
8. Do you have any recommendations about improving online reviews? Please type your answer in the box below.	(open ended question)

Table 1: List of the survey questions and their possible answers

3.1 Survey Structure

The questions and their possible answers are presented in Table 1. The survey was designed in such a way as to collect information on four main subjects. The first subject was whether or not the participants had any interest in and/or experience with reading online reviews, as well as whether the two groups read a similar amount of surveys before reaching a decision (Q1 and Q2). The second subject was whether or not the participants felt that they experienced any barriers when reading online reviews. This was assessed through two separate questions positioned at different places in the survey. The first question was formulated as a question about rating their experience with understanding the reviews (Q3), while the second one directly asked whether they experienced any barriers (Q7). Collecting responses relevant to this subject using two separate types of questions allowed assessing the consistency of the answers between the two. The third subject consisted of structured questions about specific barriers that were both: i) suggested by the literature as potential obstacles for this target population and ii) had corresponding NLP applications developed for these domains (e.g. opinion mining, figurative language identification and text summarization) (Q4, Q5 and Q6). Finally, the last subject was the recommendations that participants had on improving the accessibility of online reviews (Q8), which simultaneously revealed other frustrations that they had which we not accounted for in the structured questions. This was an open-ended question, the responses to which were coded into categories during the analysis stage.

3.2 Participants

A total of 57 participants took part in the survey, of whom 24 had a formal clinical diagnosis of autism and 33 were neurotypical control-group participants. All participants from both groups were native speakers of English, with the exception of 1 ASD participant who was native in Romanian and Hungarian but fluent in English. We screened the participants for other conditions affecting reading such as dyslexia and aphasia. None of the control-group participants had any of these conditions and 2 of the ASD participants had been diagnosed with dyslexia. The mean age in years of the ASD group was m = 40.08 (SD = 14.09). Number of years spent in formal education were m = 17.4 (SD = 3.26). 18 out of the 24 participants responded to the question “When did you receive your diagnosis?”. A total of 7 cases were diagnosed before 2013 following the diagnostic criteria outlined in the DSM-IV. The remaining 11 cases were diagnosed after 2013 following the diagnostic criteria in the DSM-5. The mean age in years of the Control group was m = 40.38 (SD = 10.89) and number of years spent in formal education were m = 16.39 (SD = 3.11).

3.3 Recruitment Channels

The majority of the participants with autism were recruited through a UK charity organisation (N = 17). Another 3 participants were recruited through the Student Enabling Centre at the University of Wolverhampton and the remaining 4 participants were recruited through peer-support groups for people with autism on Facebook. All control-group participants were recruited through snowball sampling.

3.4 Survey Administration

All participants completed an online version of the survey. All control-group participants and four ASD-participants were sent a survey link through Survey Monkey[3]. The rest of the ASD participants (N = 20) took part in a larger online reading comprehension experiment and completed the survey as an attachment to that experiment. The questions and their presentation was identical in both platforms.

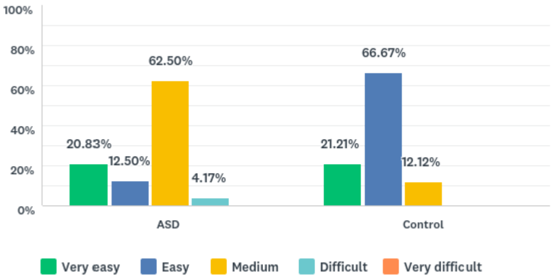
First, all participants read an information sheet and ticked “Yes” to a question asking for their informed consent to take part in the research. After that the participants were asked for their age in years, number of years spent in formal education and whether or not they had been diagnosed with any of the following: Autism Spectrum Disorder, Dyslexia, Aphasia. If they ticked “Yes” to any of these they were required to state the year in which they had received their formal diagnosis. Another answer options was “No”, in which case the participant was assigned to the Control group. The next question assessed whether or not the participant was a native speaker of English. If not, they were required to state their level of fluency in English and their mother tongue. Once information about the demographic characteristics of the participants was collected, they proceeded to answering 7 multiple-choice questions and one open-ended question related to online product reviews.

4 Results

This section presents the main results from each question of the survey.

Q1: Do you read online reviews to determine whether a product or service is good or bad? All participants who took part in the survey had experience with reading online reviews for products and services. An equal number of ASD-group participants chose the answer options *Sometimes* (33.33%, N = 8) and *Very often* (33.33%, N = 8). Another 7 participants chose the option *Every time* (29.17%, N = 7) and only 1 participant chose the option *Rarely* (4.17%, N = 1). More than half of the control-group participants said they read online reviews *Very often* (54.55%, N = 18) or *Every time* (15.15%, N = 5). Nine control participants said they read reviews *Sometimes* (27.27%, N = 9) and one participant selected the option *Rarely* (3.03%, N = 1).

Figure 1: In general, do you find understanding product reviews:



Q2: How many product reviews do you read before coming to a decision? The majority of the participants from both groups indicated that they read less than 10 reviews before coming to a decision. The answer distribution for the ASD group was 65.22% (N = 15) for the option *Less than 10*, 21.74% (N = 5) for the option *Between 10 and 20*, and 8.7% (N = 2) for the option *More than 20*. One participant had selected the answer *Most of the available reviews* (4.35%, N = 1). For the Control group, 69.7% (N = 23) of the participants chose *Less than 10*, 24.24% (N = 8) chose *Between 10 and 20*, 3.03% (N = 1) chose *Most of the available reviews*, and, surprisingly, 3.03% (N = 1) chose *None*. The answer of this last participant contradicts the results from the previous question where no participant chose the option *Never*.

Q3:In general, do you find understanding product reviews ... There was a statistically significant association between the perceived level of understanding and the group type, where the participants with ASD reported greater difficulty with understanding product reviews ($\chi^2(3) = 21.25$, $p < 0.0001$). The answer distributions are presented in Figure 1.

Q4: In general, how do you find understanding whether the author approves or disapproves of the described product? Similar to the previous question, there was a statistically significant association between the perceived understanding of the author’s opinion and the group type, where the participants with ASD reported greater difficulty with understanding what the opinion was ($\chi^2(3) = 11.94$, $p = 0.008$). The answer distribution for the two groups is presented in Figure 2.

[3] https://www.surveymonkey.com

Figure 2: In general, how do you find understanding whether the author approves or disapproves of the described product?

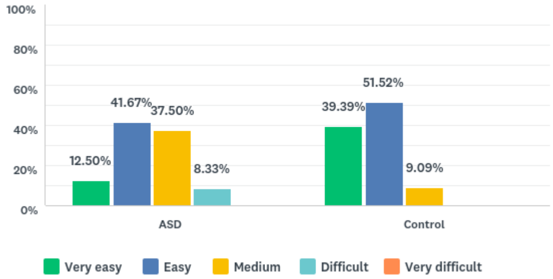
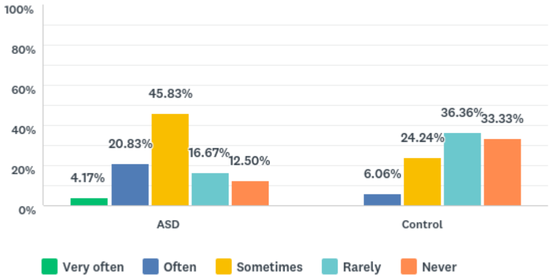


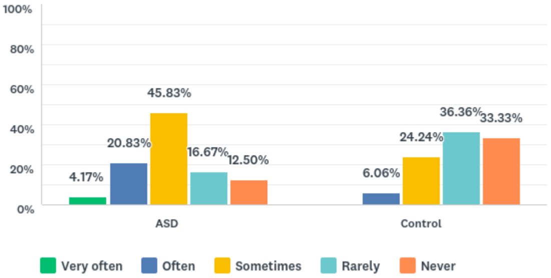
Figure 3: Have you ever felt confused about the meaning of an online review because the author used irony or sarcasm?



Q5: Have you ever felt confused about the meaning of an online review because the author used irony or sarcasm? The answers to this question were very diverse, however, there was a statistically significant association between the group type and the selected answers ($\chi^2(4) = 10.16$, *p*= 0.038). While 33.33% (N = 11) of the control group said that they had never felt confused by irony or sarcasm in online reviews, this corresponded to only 12.5% (N = 3) of the ASD group. The rest of the answer distributions are presented in Figure 3.

Q6: Do you think that having a summary of the main points of all reviews would be ... Large proportions of both groups[4] reported that they find the idea of summary of the main points of all reviews either *Very helpful* (36.36%, N = 8 of the ASD group and 45.45%, N = 15 of the Control group) or *Helpful* (45.45%, N = 10 of the ASD group and 39.39%, N = 13 of the Control group). *Moderately helpful* and *Not very helpful* were selected by an equal number of people from both groups, namely 9.09%, N = 2 of the ASD group and 6.06%, N =2 of the Controls. Finally, one participant from the Control group selected the option *Not helpful at all* (3.03%, N =1). There were no significant differences between the opinions of the two groups ($\chi^2(4) = 1.37$, *p*= 0.848).

Figure 4: Do you feel that there are certain barriers for you with regards to understanding product reviews?



Q7: Do you feel that there are certain barriers for you with regards to understanding product reviews? Analysis of the responses to this question revealed that the participants with autism significantly more often felt that there are barriers to their comprehension of online reviews[5] ($\chi^2(3) = 12.92$, *p*= 0.005). The percentages for each answer option are presented in Figure 4.

Q8: Do you have any recommendations about improving online reviews? The answers to this open-ended question were manually coded. The responses of the ASD participants were grouped into the two main categories below.

Category 1: Issues related to language and presentation.

- **Text length:** Long reviews were identified as the most confusing ones by the participants with autism and they would often give up on them because of information overload, e.g. “I don’t like those really long reviews as I can’t take it in”.

- **Organisation:** Another demand was for the information to be better organised: “I would rather have numbers and star ratings to support any language.”; “Subheadings for what to review; i.e. cost, appearance, functionality etc.”; “Bullet points or a summary of the review at the top of the review would be brilliant”, and “Having the positive and negatives in a table would be really helpful for me”.

[4] Two participants from the ASD group did not give an answer to this question

[5] One person with autism did not give an answer to this question.

Category 2: Issues related to interpretation

- **Trustworthiness:** Another general issue was anxiety over not being able to decide which reviews are truthful and which ones might be biased or should not be taken seriously. "I would also like to know who the reviewers are and their bias! I find it difficult to trust unknown sources."

- **Focus on facts instead of emotion:** There was a clear preference for facts to avoid confusion: "Stating facts rather than how it made you feel".

- **Exaggeration:** Although this was mentioned by only one participant, they explained at length that reviews containing exaggeration and jokes were very confusing for them.

The responses of the control group were mostly related to better organisation and the inclusion of summaries and subheadings.

5 Discussion

The results from the survey revealed two important points: i) that the ASD group does perceive reviews as being more challenging to comprehend, confirming the need for adaptation efforts in this domain and ii) the specific aspects in which they find the reviews challenging, together with recommendations for their improvement.

First, participants with and without autism alike had a similar disposition towards reading product reviews and the amount of reviews they read before coming to a decision (less than 10). However, the group with autism consistently perceived online product reviews as being more difficult to understand compared to the control group (Q3). Furthermore, significantly more people from the ASD group felt that there were certain barriers to understanding product reviews compared to the control-group participants (Q7), which was a control question that assessed similar information as Q3 (general understanding of product reviews). Formulating this query in two different ways gave consistent results about the perceived difficulty of online reviews for people with autism, which was significantly higher compared to controls.

With regards to specific aspects of the reviews that were potentially challenging, there were significant between-group differences in terms of the understanding of the author's opinion of the product (Q4) and the use of irony and sarcasm (Q5). This suggests that adaptation strategies related to *sentiment analysis* or *opinion mining*, together with *figurative language identification* would be suitable for this domain and target population. Both groups felt strongly in favour of having a summary of the main points of all reviews (Q6), indicating that *text summarisation* would also be helpful for improving accessibility.

The open-ended question revealed even more aspects that need to be improved, including the length of the text, the lack of consistent structure and the use of exaggeration (Q8). Again, summarisation and *topic modeling* could help improve the structure of the reviews and potentially present them as a populated table of characteristics that has a consistent structure. An interesting addition was the issue with trusting the reviews. While relevant to all, this may be particularly challenging for individuals on the spectrum due to overall comprehension issues. Therefore, another application that would be particularly helpful for this group of web users would be the *detection of fake reviews*. While less applicable to a broad type of texts, detecting fake reviews or posts is also relevant to improving the accessibility of social media by making it a safer space.

It is important to note that these results reflect the *perceived* experiences of the participants rather than their actual processing of product reviews. It is therefore possible that the results from evaluation studies of specific text adaptation strategies may point out to different outcomes. Nevertheless, the perceived experiences of the target group should always be taken into consideration when developing technical solutions and gaining insight into what these are is the first step towards making the web more accessible.

6 Conclusion

We conducted a survey with 24 participants with high-functioning autism and 33 neurotypical control participants on their experiences with reading online product reviews. The results showed that both groups were interested in reading reviews before making a purchasing decision but that the ASD group perceived comprehending the reviews to be significantly more challenging. Appropriate strategies for making the reviews more accessible were clarifying the opinion of the author, identifying any figurative language and summarising the main points of the review, together with enhancing the way the information is structured, as well as flagging reviews that are not trustworthy.

References

American Psychiatric Association. 2013. Diagnostic and Statistical Manual of Mental Disorders (5th ed.).

Eduard Barbu, M Teresa Martín-Valdivia, Eugenio Martínez-Cámara, and L Alfonso Ureña-López. 2015. Language technologies applied to document simplification for helping autistic people. *Expert Systems with Applications*, 42(12):5076–5086.

Simon Baron-Cohen. 2000. Theory of mind and autism: A fifteen year review. *Understanding other minds: Perspectives from developmental cognitive neuroscience*, 2:3–20.

Traolach S Brugha, Sally McManus, John Bankart, Fiona Scott, Susan Purdon, Jane Smith, Paul Bebbington, Rachel Jenkins, and Howard Meltzer. 2011. Epidemiology of autism spectrum disorders in adults in the community in england. *Archives of general psychiatry*, 68(5):459–465.

Sukru Eraslan, Victoria Yaneva, Yeliz Yesilada, and Simon Harper. 2017. Do web users with autism experience barriers when searching for information within web pages? In *Proceedings of the 14th Web for All Conference on The Future of Accessible Work*, page 20. ACM.

Sukru Eraslan, Victoria Yaneva, Yeliz Yesilada, and Simon Harper. 2019. Web users with autism: eye tracking evidence for differences. *Behaviour & Information Technology*, 38(7):678–700.

Richard Evans, Constantin Orasan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism. Association for Computational Linguistics.

Rebecca Eynon and Helen Margetts. 2007. Organisational solutions for overcoming barriers to egovernment.

L. Fretwell, J. Stine, H. Sethi, and A. Noronha. 2013. 'catch and keep' digital shoppers how to deliver retail their way. CISCO Internet Business Solutions Group.

Geert Freyhoff, Gerhard Hess, Linda Kerr, Bror Tronbacke, and Kathy Van Der Veken. 1998. Make it simple.

Happe, F., and Frith, U. 2006. he weak coherence account: Detail focused cognitive style in autism spectrum disorder. *Journal of Autism and Developmental Disorders*, 36:5–25.

Gilbert MacKay and Adrienne Shaw. 2004. A comparative study of figurative language in children with autistic spectrum disorders. *Child Language Teaching and Therapy*, 20(13).

I.M. O'Connor and P.D. Klein. 2004. Exploration of Strategies for Facilitating the Reading Comprehension of High-Functioning Students with Autism Spectrum Disorders. *Journal of autism and developmental disorders*, 34(2).

Constantin Orăsan, Richard Evans, and Ruslan Mitkov. 2018. Intelligent text processing to help readers with autism. In *Intelligent Natural Language Processing: Trends and Applications*, pages 713–740. Springer.

Sanja Štajner, Richard Evans, and Iustin Dornescu. 2014. Assessing conformance of manually simplified corpora with user requirements: the case of autistic readers. In *Proceedings of the Workshop on Automatic Text Simplification-Methods and Applications in the Multilingual Society (ATS-MA 2014)*, pages 53–63.

Sanja Štajner and Horacio Saggion. 2013. Readability indices for automatic evaluation of text simplification systems: A feasibility study for spanish. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 374–382.

Elisabeth M Whyte, Keith E Nelson, and K Suzanne Scherf. 2014. Idiom, syntax, and advanced theory of mind abilities in children with autism spectrum disorders. *Journal of Speech, Language, and Hearing Research*, 57(1):120–130.

Victoria Yaneva. 2016. *Assessing text and web accessibility for people with autism spectrum disorder*. Ph.D. thesis.

Victoria Yaneva, Le Ha, Sukru Eraslan, and Yeliz Yesilada. 2019. Adults with high-functioning autism process web pages with similar accuracy but higher cognitive effort compared to controls.

Victoria Yaneva, Le An Ha, Sukru Eraslan, Yeliz Yesilada, and Ruslan Mitkov. 2018. Detecting autism based on eye-tracking data from web searching tasks. In *Proceedings of the Internet of Accessible Things*, page 16. ACM.

Victoria Yaneva, Constantin Orasan, Richard Evans, and Omid Rohanian. 2017. Combining multiple corpora for readability assessment for people with cognitive disabilities. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 121–132.

Victoria Yaneva, Irina Temnikova, and Ruslan Mitkov. 2016a. A corpus of text data and gaze fixations from autistic and non-autistic adults. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 480–487.

Victoria Yaneva, Irina Temnikova, and Ruslan Mitkov. 2016b. Evaluating the readability of text simplification output for readers with cognitive disabilities. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 293–299.

An Open, Extendible, and Fast Turkish Morphological Analyzer

Olcay Taner Yıldız[1], Begüm Avar[2], Gökhan Ercan[1]
[1] Department of Computer Engineering, Işık University, İstanbul, Turkey
[2] Department of Linguistics, Boğaziçi University, İstanbul, Turkey
olcaytaner@isikun.edu.tr, begum.avar@boun.edu.tr
gokhan.ercan@isik.edu.tr

Abstract

In this paper, we present a two-level morphological analyzer for Turkish which consists of five main components: finite state transducer, rule engine for suffixation, lexicon, trie data structure, and LRU cache. We use Java language to implement finite state machine logic and rule engine, Xml language to describe the finite state transducer rules of the Turkish language, which makes the morphological analyzer both easily extendible and easily applicable to other languages. Empowered with a comprehensive lexicon of 54,000 bare-forms including 19,000 proper nouns, our morphological analyzer is amongst the most reliable analyzers produced so far. The analyzer is compared with Turkish morphological analyzers in the literature. By using LRU cache and a trie data structure, the system can analyze 100,000 words per second, which enables users to analyze huge corpora in a few hours.

1 Introduction

Morphological analysis is one of the key components of computational language processing, especially in morphologically rich languages. After some preprocessing stages such as stripping nontextual parts from the corpus and sentence segmentation, the first and foremost stage in computational analysis of the text is morphological analysis, that is dividing the words into constituent morphemes.

In this study, we deal with the morphology of Turkish, which is a textbook example for an agglutinative language. In Turkish, a word in its surface form contains 3 to 4 morphemes on the average (Sak, 2011), and these morphemes can have a semantic and/or syntactic content. Not only a surface form can have a multimorpheme structure in Turkish, but also has multi morphological analyses.

In this paper, we will present a new morphological analyzer, which is (i) open: The latest version of source codes, the lexicon, and the morphotactic rule engine are all available on the Internet[1], (ii) extendible: One of the disadvantages of other morphological analyzers is that their lexicons are fixed or unmodifiable, which prevents to add new bare-forms to the morphological analyzer. In our morphological analyzer, the lexicon is in text form and is easily modifiable, (iii) fast: Morphological analysis is one of the core components of any NLP process. It must be very fast to handle huge corpora. Compared to other morphological analyzers, our analyzer is capable of analyzing hundreds of thousands words per second, which makes it one of the fastest Turkish morphological analyzers available.

2 Turkish Morphology

In linguistics, the term *morphology* refers to the study of the internal structure of words. Each word is assumed to consist of one or more *morphemes*, which can be defined as the smallest linguistic unit having a particular meaning or grammatical function. One can come across morphologically simplex words, i.e. *roots*, as well as morphologically complex ones, such as compounds or affixed forms.

(1) Batı-lı-laş-tır-ıl-ama-yan-lar-dan-mış-ız
west-With-Make-Caus-Pass-Neg.Abil-Nom-Pl-Abl-Evid-A3Pl
'It appears that we are among the ones that cannot be westernized.'

[1] https://github.com/olcaytaner/TurkishMorphologicalAnalysis

The morphemes that constitute a word combine in a (more or less) strict order. Most morphologically complex words are in the ”ROOT-SUFFIX1-SUFFIX2-. . .” structure. Affixes have two types: (i) derivational affixes, which change the meaning and sometimes also the grammatical category of the base they are attached to, and (ii) inflectional affixes serving particular grammatical functions. In general, derivational suffixes precede inflectional ones. The order of derivational suffixes is reflected on the meaning of the derived form. For instance, consider the combination of the noun göz ‘eye’ with two derivational suffixes -lIK and -CI: Even though the same three morphemes are used, the meaning of a word like göz-cü-lük ‘scouting’ is clearly different from that of göz-lük-çü ‘optician’.

Owing to its morphological properties, in Turkish, the problem of parsing and disambiguation constitutes a major challenge, as not only content words in their base forms but also functional morphemes may be the source of ambiguity. While certain affixes have clear functions/meanings, there exist others, for which the meaning can only be determined within a context. In fact, only a few derivational suffixes (which are productively used for word-formation) are semantically transparent and some derived forms are no longer considered to be compositional in that their meaning cannot be predicted from the morphemes they contain.

2.1 Allomorphy

Linguists speak of underlying representations (UR) of morphemes, which is the representation of a form in the mental lexicon, and surface representations of morphs, which concerns their exact pronunciation. While some morphemes may be realized in a single way, others have several allomorphs, i.e. phonetic variants. There are, in general, two major kinds of allomorphy with respect to their source: (i) phonologically conditioned, and (ii) morphologically or lexically conditioned.

2.1.1 Phonologically-Conditioned Allomorphy

Turkish has a rich inventory of phonological rules affecting the pronunciation of morphemes and dictating what a well-formed word in Turkish may look (or rather sound) like. There are restrictions concerning the distribution of vowel segments (V), consonants segments (Cs) or their co-occurrence. For instance, the nature of consonant clusters is highly restricted. Due to the rules dictating the so-called Vowel-Alternation, an epenthetic high vowel is inserted to break up a disallowed sequence of consonants. Morphology must respect the rules of phonology, and therefore, for instance, an impermissible cluster cannot be formed through morphological operations – an epenthetic vowel comes to rescue as in (Table 1(b)), as opposed to (Table 1(a)).

a. kedi ‘cat’ + 1st person Poss = kedim ‘my cat’
b. ev ‘house’ + 1st person Poss = evim ‘my house’

Table 1: Example cases showing how morphology must respect the rules of phonology.

UR	Bare-form	Accu.	Plural
a. akl ‘mind’	a.kıl	ak.l-ı	a.kıl-lar
b. sırr ‘secret’	sır	sır.r-ı	sır-lar

Table 2: Example words whose underlying representation ends in an impermissible cluster or a geminate.

Bare-form	Accu.	Dative	Plural
a. kas ‘muscle’	kas-ı	kas-a	kas-lar
b. kasa ‘safe’	kasa-yı	kasa-ya	kasa-lar

Table 3: Example Turkish words which do not contain two successive vowels.

There are some words in Turkish whose underlying representation ends in an impermissible cluster or a geminate. In the former case, vowel epenthesis takes place (Table 2(a)) whereas words of the latter kind undergo degemination (Table 2(b)), unless they are followed by a vowel which results in the resyllabification of the second consonant in the cluster (‘.’ indicates syllable boundary in the examples below).

There are also restrictions on neighboring vowels. Typically, Turkish words do not contain two successive vowels, except for some loanwords. Therefore, if a vowel-initial suffix is attached to a vowel-final word, a consonant (typically ‘y’) emerges to avoid a VV sequence, as demonstrated in (Table 3(b)).

A further phonological operation in Turkish is Vowel Harmony, which requires any vowel to agree in backness and any high vowel to agree in rounding with the preceding vowel. Respecting the rules dictating harmony, suffixes in Turkish have various allomorphs. Turkish vowel harmony

Bare-form	Accusative	Plural
a. at ‘horse’	at-ı	at-lar
b. et ‘meat’	et-i	et-ler
c. saat ‘hour, clock’	saat-i	saat-ler

Table 4: Example Turkish words which obey/does not obey vowel harmony while taking suffixes.

Bare-form	Past
kal ‘stay’	kal-dı ‘She/he/it stayed’

Table 5: Example Turkish cases with consonant harmony.

UR	Bare-form	Accu.	Plural
a. kitab ‘book’	kitap	kitab-ı	kitap-lar
b. top ‘ball’	top	top-u	top-lar

Table 6: Example Turkish words with word-final devoicing.

operates regularly on suffixes with a very few exceptions consisting of non-alternating suffixes, the most productive of which being the progressive suffix –(I)yor (e.g. when attached to git ‘to go’, we get gidiyor ‘s/he goes’, instead of *gidiyör or *gidiyir) and some words which take a front-vowel suffix even though their last vowel is back (Table 4)).

Other than Vowel Harmony, there is also Consonant Harmony in Turkish according to which oral stops and affricates agree in voicing with the preceding segment. Hence, a suffix, such as the past tense morpheme –DI, has up to 8 allomorphs when both consonant and vowel harmonies are applicable (Table 5)).

Another process relating to the voicing properties of consonants is called ‘word-final devoicing’, according to which word-final obstruents must be voiceless. Words, or rather morphemes, that are affected by this process have a voiced obstruent in the final position in their UR (as in Table 6(a)), which gets devoiced unless a vowel-initial suffix follows.

A further alternation targets the word- or rather morpheme-final ‘k’s. Dubbed ‘k-alternation’ in the literature, this process results in the replacement of morpheme-final ‘k’s with ‘ğ’ (which phonologically means a lengthening of the vowel preceding it) when they are followed by a vowel-initial suffix. K-alternation may affect roots (as in köpek ‘dog’ + ACC = köpeği instead of *köpeki) as well as suffixes (as in göz ‘eye’ + lIK + ACC forming gözlüğü instead of *gözlükü).

a. dur ‘stop, stand’ → dur-ur	bil ‘know’ → bil-ir
b. kur ‘set up’ → kur-ar	sil ‘wipe’ → sil-er

Table 7: Examples of unpredictable allomorphy.

a. it ‘push’ → it-tir	bit ‘end’ → bit-ir
b. bak ‘look’ → bak-tır	ak ‘flow, leak’ → ak-ıt

Table 8: Examples of unpredictable allomorphy for the causative suffix.

The list of phonological processes presented in this section covers some of the most frequently occurring alternations, yet it is not exhaustive. For one, there are also processes that do not have any reflection on orthography, such as alternations on vowel length. There are several others which have a narrower distribution, such as vowel reduction occurring in verbs ending in a vowel.

2.1.2 Morphologically or Lexically Conditioned Allomorphy

The crucial difference between phonologically conditioned allomorphy from other types of allomorphy is that in the former case, the alternations are predictable and apply regularly, while in the latter case the phonological features, by themselves, are not sufficient to define the environment in which the change takes place. Among the suffixes having several allomorphs, which cannot be accounted for by phonological premises only, is the aorist. While some of its forms are phonologically conditioned and thus predictable, others (Table 7(a) vs. Table 7(b)) are unpredictable from the phonological shape of the base they are attached to. A similar allomorphy is found for the causative suffix, as demonstrated in Table 8.

2.2 Inflectional Categories

Inflectional markers encode grammatical information and are category-selective. For instance, (i) Number (Singular vs. Plural); (ii) Case (nominative, accusative, dative, locative, ablative, genitive, comitative); and (iii) Possessive are encoded by inflectional suffixes (or the lack of them) on nominal stems.

Inflectional suffixes attached to verbal stems encode (i) Tense/Aspect/Modality (such as Past, Future, Aorist, Progressive, Evidential, Optative, Conditional, Ability/Possibility, Obligative etc.); (ii) Agreement (person & number); (iii) Voice (Passive, causative, reflexive, reciprocal); and (iv) Polarity (Affirmative vs. Negative).

Due to spatial restrictions, only a brief overview of morphological processes in Turkish is presented in this paper. For further information on the linguistic structure of Turkish in general, and on Turkish morphology in particular, the reader is referred to (Lewis, 1967), (Kornfilt, 1997), (Underhill, 1976), (Göksel and Kerslake, 2005), and (Erguvanlı, 2015).

3 Related Work

3.1 Available Resources

There exist several morphological analyzer resources in the Turkish NLP literature. In this section, we aim at analyzing currently available resources in terms of their usage, technology, structure, availability, and extendibility. Table 9 shows a comparison table of such resources along with our utility. Publicly unavailable resources such as widely known KIMMO-based (Karttunen et al., 1983) analyzer (Oflazer, 1994) were decidedly left out of scope of this study.

Sak et al. (2008) released the Finite-State Morphological Parser (SakMP) which uses AT&T FSM parser (Mohri, 1997). Although it is not publicly available, authors provide compiled Linux library (*.so file) upon requests. As it does not depend on any external components on runtime, researchers can call the services through the command line or Python scripting without making any installations on Linux systems. Since it is delivered as a single compiled file, its lexicon, suffixation rules and the transducer is not extendible for researchers.

TRMorph (Çöltekin, 2010) is another morphological analyzer implementation which is built upon an existing FST engine. Its latest version[2] (2.0 pre-release) uses Foma FST compiler (Hulden, 2009) which is basically a C compiler converting regular expressions to finite automata and transducers. TRMorph's lexicon files are in a raw text file format which makes them easily updatable for the researcher. It has a special regular expression based syntax (through *.xfst files) that enable researchers to update suffixation rules in compile time. Once the output file (*.fst) compiled for the platform, it can be queried with the help of Foma executables through the command line or Python scripts. The author has also introduced a web service integration with the WebLicht environment (Hinrichs et al., 2010) which allows serving TRMorph's functionality through the web interfaces (Çöltekin, 2015).

Similarly, ITU Turkish NLP Web Service (ITUWS) (Eryiğit, 2014) offers a publicly available[3] NLP user interface[4] and a web service for Turkish language which covers common tools such as tokenizer, morphological analyzer/disambiguator and dependency parser. ITUWS requires an access token on http requests in order authenticate researchers to web services. According to their paper, ITUWS wraps the morphological analyzer tool ITUMORPH (Şahin, 2013)(Şahin et al., 2013) which depends on another external tool HFST (Lindén et al., 2009) for its FST implementation. Since ITUWS is a web service-based resource, it is not suitable for tasks that require millions of analyses. Another downside of the web service-based model is the researcher could not modify any of its components such as lexicon, suffixes, and suffixation rules.

Lastly, Zemberek[5] is a popular open-source NLP framework which includes tools for Turkish such as morphological analyzer/disambiguator, tokenizer, and spell checker. It has been using as a spelling checking extension for LibreOffice[6] and the Turkish national Linux Distribution Pardus.[7] Although the project is still actively developed and maintained, its original paper is quite outdated (Akın and Akın, 2007). In documentation pages, authors note that the latest version of the library almost written from the scratch. While the original goal of the project was to abstract language specific components form the parser to support all Turkic languages (e.g., Turkmen, Azeri, Uzbek), its current focus seems on the Turkish language only. Unlike the generally accepted approach in the literature, Zemberek does not use a FST for morphological parsing. Whereas it allows developers to easily modify the lexicon through text files and the API, updating the suffixation and morphotactic rules require recompilation because of they are represented in the core Java code.

4 Core Components

Our morphological analyzer consists of five main components, namely, a lexicon, a finite state trans-

[2] https://github.com/coltekin/TRmorph

[3] http://tools.nlp.itu.edu.tr/

[4] http://tools.nlp.itu.edu.tr/MorphAnalyzer

[5] https://github.com/ahmetaa/zemberek-nlp

[6] https://extensions.libreoffice.org/extensions/zemberek-turkce-yazim-denetleyicisi

[7] https://www.pardus.org.tr/

Resource	Ours	SakMP	TRMorph	ITUWS	Zemberek
Paper	this paper	Sak et al. (2008)	Çöltekin (2010)	Eryiğit (2014)	Akın and Akın (2007)
Availability	open-source	by request	open-source	free, by request	open-source
Form	Java library	binary	Foma impl.	web service	Java library
Compile-time	Java runtime	AT&T FSM	Foma, C proc.	invisible	Java runtime
Runtime Dep.	Java runtime	no dep.	Foma, Unix tools	any REST	Java runtime
Operating Sys.	Win, OSX, Linux	Linux	Win, OSX, Linux	Any OS	Win, OSX, Linux
FST Impl.	custom Java	AT&T FSM	Foma	HFST	no FST
Lexicon	text file	embedded	text file	invisible	text file
Suffix Rules	xml file	embedded	regex, C, lexc	invisible	Java code

Table 9: Comparison of available morphological analyzer resources.

ducer, a rule engine for suffixation, a trie data structure, and a least recently used (LRU) cache.

4.1 Lexicon

For the purposes of the present study, we will assume all idiosyncratic information to be encoded in the lexicon. While phonologically conditioned allomorphy will be dealt with by the transducer, other types of allomorphy (including the ones discussed in Section 2.1.2), all exceptional forms to otherwise regular processes, as well as words formed through derivation (except for the few transparently compositional derivational suffixes) are considered to be included in the lexicon.

Table 10 shows 10 example words taken from our lexicon, where the lexicon is sorted alphabetically. Each line in the lexicon consists of the bare-form and a set of attributes separated by white space.

4.1.1 Bare-Forms

The bare-forms in the lexicon consists of nouns, adjectives, verbs, adverbs, shortcuts, etc. Each bare-form appears the same in the lexicon except verbs. Since the bare-forms of the verbs in Turkish do not have the infinitive affix 'mAk', our lexicon includes all verbs without the infinitive affix. For instance, in Table 10, verbs 'abanmak' and 'abartmak' appear as 'aban' and 'abart' respectively.

Since our morphological analyzer must support all types of texts, the bare-forms with diacritics are included in two forms, with and without diacritics. For example, noun 'rüzgâr' appear both as 'rüzgâr' and 'rüzgar'.

Special markers are included as bare-forms such as <doc>, <s>, etc.

Some compound words are included in their affixed form. For instance, 'acemlalesi' appears as it is, but not as 'acemlale'.

Foreign words, especially proper noun foreign words, are included, so that the system can easily recognize them as proper nouns. In Table 10, the words 'abbott', 'abbigail' are example foreign proper nouns. Including foreign proper nouns, there are 19,000 proper nouns in our lexicon.

From derivational suffixes, we only include words which has taken -lI, -sIz, -CI, -lIk, and -CIlIk derivational affixes. In Table 10, the bare-forms 'abacı', 'abdallık', 'abdestli' and 'abdestlilik', are included, since they have taken one or more derivational affixes listed above.

abacı CL_ISIM	aban CL_FIIL F5PR
abart CL_FIIL F5PR	abbott IS_OA
abbigail IS_OA	abdallık CL_ISIM IS_SD
abdestli IS_ADJ	abdestlilik CL_ISIM IS_SD
rüzgar CL_ISIM	rüzgâr CL_FIIL CL_ISIM

Table 10: 10 example words from our lexicon.

4.1.2 Attributes

Each bare-form has a set of attributes give in Table 11. For instance, in Table 10, 'abacı' is a noun, therefore, it includes CL_ISIM attribute. Similarly, 'abdestli' is an adjective, which includes IS_ADJ attribute. If the bare-form has homonyms with different part of speech tags, all corresponding attributes are included.

4.2 Finite State Transducer

Given a possible bare-form, depending on the possible part of speech(es) of that bare-form, the finite state transducer (FST) starts with one or more initial state. FST is responsible from state transitions, where at each transition FST (i) changes from one state to another state, (ii) appends a suffix to the current surface form to generate a new surface form, (iii) produces an output, which is the current morphological analysis of the current surface form. After a set of transitions, the current surface form will be equal to the word, for which morphological analysis sought, and if also the current state is a final state, FST will output current morphological analysis as a possible mor-

Name	Purpose
CL_ISIM, CL_FIIL, ...	Part of speech tag(s)
IS_OA	Proper noun
IS_DUP	Part of a duplicate form
IS_KIS	Abbreviation, which does not obey vowel harmony while taking suffixes.
IS_UU, IS_UUU	Does not obey vowel harmony while taking suffixes.
IS_BILEŞ	A portmanteau word in affixed form, such as 'adamotu'
IS_B_SI	A portmanteau word ending with 'sı', such as 'acemlalesi'
IS_CA	Already in a plural form, therefore can not take plural suffixes such as 'ler' or 'lar'.
IS_ST	The second consonant undergoes a resyllabification.
IS_UD, IS_UDD, F_UD	Includes vowel epenthesis.
IS_KG	Ends with a 'k', and when it is followed by a vowel-initial suffix, the final 'k' is replaced with a 'g'.
IS_SD, IS_SDD, F_SD	Final consonant gets devoiced during vowel-initial suffixation.
F_GUD, F_GUDO	The verb bare-form includes vowel reduction.
F1P1, F1P1-NO-REF, ...	A verb, and depending on this attribute, the verb can (or can not) take causative suffix, factitive suffix, passive suffix etc.

Table 11: Attributes of the bare-forms

phological analysis. Depending on the number of initial states, the number of possible paths FST has sought, FST can output one or more possible morphological analyses.

In our morphological analyzer, FST is encoded in an xml file. Table 12 shows three example states from our Turkish FST xml file. <state> tag shows the properties of a state including; **name** of the state, if the state is a **start** state, if the state is a **final** state, the **pos** (part of speech) of the state if the state is a start state.

<to> tag shows the properties of state transitions from the current state including; **name** of the next state, **output** of the transducer while doing this transition, if the transition changes the part of the speech, **pos** of the produced surface form.

<with> tag has a text showing the suffix to append to the current surface form in this current state. **null** transitions are shown with '0' suffix. Similar to the <to> tag, <with> tag may have **output** of the transducer while doing this transition; if the transition changes the part of the speech, **pos** of the produced surface form.

In Table 12, first state is the *ConjunctionRoot* state, which shows the initial state for conjunctions. Since in Turkish conjunctions can not take suffixes, it is also a final state with no additional transitions. Second state is the *VerbalRoot(F4PW)* state, which shows the initial state for a specific class of verbs. These verbs can take a reciprocal suffix 'Hs', a causative suffix 't', a passive suffix 'n', and null passive suffix. Third state is the *NominalRootPlural* state, which shows the initial state for root nouns already in plural form. Since they are already in plural form, the morphological outputs always start with 'A3PL' and depending on the possesive suffix, the person of the possessive is determined.

```
<state name="ConjunctionRoot" start="yes"
 final ="yes" pos="CONJ">
</state>
<state name="VerbalRoot(F4PW)" start="yes"
 final ="no" pos="VERB">
    <to name="Reciprocal" output="RECIP">
        <with>Hs</with>
    </to>
    <to name="PassiveHn">
        <with>0</with>
    </to>
</state>
<state name="NominalRootPlural" start="yes"
 final ="no" pos="NOUN">
    <to name="Possessive">
        <with output="A3PL+PNON">0</with>
        <with output="A3PL+P1SG">Hm</with>
        <with output="A3PL+P2SG">Hn</with>
        <with output="A3PL+P1PL">HmHz</with>
        <with output="A3PL+P2PL">HnHz</with>
    </to>
</state>
```

Table 12: Example states from Turkish FST.

4.3 Morphotactic Rule Engine

Given the FST and a possible transition, the rule engine's job is to apply morphotactical rules to append the suffix (in the transition) to the current surface form to produce the suffixed surface form.

When the suffixes are appended to the bare-form, for the categorical exceptional cases, rule engine uses the attributes of the bare-forms given in Section 4.1.2. So, for example, if the bare-form is 'saat', and since it has an attribute IS_UU, when the accusative suffix is appended to that word, we get the surface form 'saati' (Table 4c)).

The most important function of morphotactic rule engine is to solve allomorphic cases, that is phonetic realization of metamorphemes such as 'Hs'. There are a total of four allomorphs defined in our FST. The allomorphs, the metamorphemes which use those allomorphs, and their possible phonetic realizations are given in Table 13).

There are also well known exceptions in the application of morphotactical rules, for example, when the pronouns 'bu', 'şu', 'o' get the suf-

A.morph	M.morpheme	Real.
D	DA, DAn, DH, DHk	d, t
A	Ar, CA, cAsHnA, DA	a, e
C	CA, CH, CHk	c, ç
H	cAsHnA, CH, CHk, DH	ı, i, u, ü

Table 13: Allomorphs defined in our FST and some of the metamorphemes and their realizations.

fix ‘ylA’, the surface form is not ‘buyla’, ‘şuyla’, ‘oyla’; but ‘bununla’, ‘şununla’, ‘onunla’. Another example is, when the pronoun ‘ben’ gets the suffix ‘ya’, the surface form is not ‘bene’ but ‘bana’. The engine must also handle these kind of irregularities.

4.4 Trie Data Structure

One of the most important tasks of a morphological analyzer is to guess possible bare-forms given a surface form. The speed of the analyzer mainly depends on the number of initial bare-forms it starts with. When extraneous bare-forms are present, FST deals with unnecessary morphotactics and / or phonetic realizations, which decreases the speed of the analyzer significantly.

The naive approach of taking the k ($1 \leq k \leq$ length of the surface form) leftmost characters as possible bare-forms does not work well, since there are many irregularities. For instance, if the word ‘ahenk’ takes an accusative suffix, ‘k’ is replaced with ‘g’, resulting in the word ‘ahengi’. In this case, we can not get the bare-form ‘ahenk’ by taking any leftmost characters of the word ‘ahengi’.

To overcome these irregularities and also to accelerate the search for the bare-forms, we use a trie data structure in our morphological analyzer, and store all words in our lexicon in that data structure. For the regular words, we only store that word in our trie, whereas for irregular words we store both the original form and some prefix of that word. Let s^k represent the leftmost k characters of a string s, $s[m]$ represent the m'th character of a string s, and l represent the length of string s. The irregular cases occur, when the bare-form has one of the following attributes.

In these cases, we insert token t into the trie. Figure 1 shows the above cases on a trie data structure. After inserting all of the lexicon into the trie, we are ready for searching the candidate bare-forms of a given surface form. We just traverse the trie and select matched words in the trie as candidate bare-forms.

Attribute	t	Example
IS_BILEŞ	s^{l-1}	ademot
IS_B_SI	s^{l-2}	acemlale
IS_UD, IS_UDD, F_UD	$s^{l-2} + s[l] + s[l-1]$	aklı
IS_KG	s^{l-1} + ‘g’	aheng
IS_SD, IS_SDD, F_SD	s^{l-1} + (‘b’ $\mid$ ‘c’ $\mid$ ‘d’ $\mid$ ‘ğ’)	açlığ
F_GUD, F_GUDO	s^{l-1}	açıkl

Table 14: Tokens to be inserted for the selected attributes.

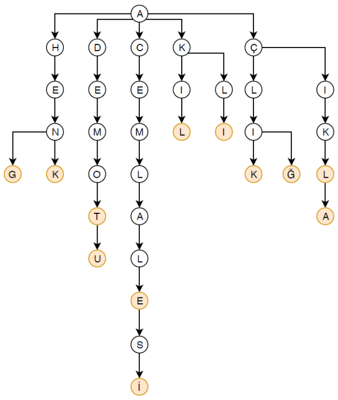


Figure 1: An example trie in our analyzer.

4.5 LRU Cache

The speed of a morphological analyzer is usually calculated on large corpora. These corpora contains millions (sometimes billions) of surface forms and as expected include many surface forms repeatedly. Since the morphological analyses of a surface form does not depend on the neighboring words, one can safely assume that, once we have extracted the morphological analyses of a word, we do not need to reextract those analyses. We can just look up the analyses of that surface form from a cache.

The idea of caching items for fast retrieval goes back nearly to the beginning of the computer science. We also use that idea and use a LRU cache for storing morphological analyses of surface forms. Before analyzing a surface form, we first look up to the cache, and if there is an hit, we just take the analyses from the cache. If there is a miss, we analyze the surface form and put the morphological analyses of that surface form in the LRU cache. As can be expected, the speed of the caching mechanism surely depends on the size of the cache. In our experiments, our cache contains up to 100K (sometimes 1M) surface forms.

5 Evaluation

5.1 Functional Evaluation

The main motivation behind our functional experiments is to test our morphological analyzer's parsing capability against known suffixation cases based on our own lexicon entries and their attributes described in the section 4.1.2. To achieve this, first, we synthetically generated test cases for 14 attributes by applying suffixes to bare-forms using our FST as a word generator. Then reversely, we tried to analyze the generated words using the same engine. After fixing some incorrectly generated cases manually, we ended up with 28,900 generated test cases in total. Lastly, we ran generated test cases on all available morphological analyzers. Table 15 shows the *passed case counts* of 14 group of test cases for each morphological analyzer. Our analyzer successfully parsed generated tests cases with 99.36% accuracy.

Test oracles for test cases slightly differ based on their group of attributes. For instance, IS_OA group test cases check whether the parser successfully yields any analysis which contains the given proper noun. This is the most simple group of test cases where its success depends heavily on the existence of such proper nouns in lexicons. Another example from group IS_UU has the test oracle ”$saat \mid saatler$” meaning that at least one analysis (e.g., ”saat+NOUN+A3PL+PNON+NOM”) of the second string ‘saatler’ should contain the first string ‘saat’ as a bare-form. All analyzers have marked as ’passed’ in this specific test case. One last special test oracle example is the IS_BI_SI attribute. This time the test oracle marks the case as passed even if the last two characters of the parsed bare-form does not match the given bare-form. This special oracle resolves the falsely failed cases caused by the incompatible bare-form structures of different analyzers as in the example of ”$geceyarısı \mid geceyarıları$” where the parsed bare-form of the second string can be accepted for both ‘geceyarı’ or ‘geceyarısı’. In this study, our test oracles do not check for any special tags (i.e., POS, MTAG) or suffixes while marking the test case as passed or not. We leave the test cases with more advanced oracles for future work.

Groups	Cases	Ours	Zemb.	TRMo.	SakMP	ITUWS
F_GUD	1,367	1,366	1,324	337	1,261	1,333
IS_BILES	1,186	1,084	547	109	260	1,186
IS_UD	176	175	151	151	132	116
IS_ST	38	37	32	25	24	14
IS_UU	347	324	265	231	243	309
IS_KG	26	26	24	24	22	25
IS_CA	415	415	348	296	326	415
F_SD	90	90	90	62	76	82
IS_UUU	10	10	7	7	6	8
IS_BI_SI	201	199	77	3	38	200
F_UD	12	12	10	6	2	2
F_GUDO	2	2	2	2	2	2
Subtotal	3,870	3,740	2,877	1,253	2,392	3,692
	100%	**96.64%**	74.34%	32.38%	61.81%	95.4%
IS_SD	5,970	5,917	2,591	2,025	3,085	-
IS_OA	19,060	19,054	14,649	15,005	12,646	-
	100%	**99.78%**	68.88%	68.04%	62.85%	-
Overall	28,900	28,716	20,117	18,283	18,123	-
	100%	**99.36%**	69.61%	63.26%	62.71%	-

Table 15: The number of passed test cases on functional evaluations, grouped by attributes.

	Ours		**TRMorph**		**Zemberek**	
Corpus (words)	Dur.	Par.	Dur.	Par.	Dur.	Par.
Milliyet (810K)	22 sec	1.2M	39 sec	-	17 sec	1.6M
Gazete (19M)	219 sec	37M	950 sec	-	330 sec	47M
BounC. (433M)	24 min	839M	161 min	-	72 min	1.1B

Table 16: Durations (Dur.) and parse counts (Par.)

5.2 Performance Evaluation

For the performance comparison, we used three corpora in different sizes: Milliyet (Hakkani-Tür et al., 2002), BounCorpus (Sak et al., 2008), and our corpus Gazete. All datasets are constructed from daily news websites. We excluded ITUWS and SakMP from the experiments. Since their usage models and platform (web service-based and Linux) is different, it would not yield comparable results with others. Table 16 shows final durations and number of analyses/parses of our performance tests. Number of returning parses vary depending on the authors' design choices for morphological analysis process. Parse counts are not related with the execution performances.

Our performance experiments show that our analyzer can analyze a big corpus with 37.2 million sentences in 24 minutes. Compared to the other analyzers, our tool's built-in cache mechanism leverages the memory efficiently which leads to the reduction of the analysis times.

6 Conclusion

In this paper, we have presented common challenges of morphological analysis task for Turkish caused by the rich morphology of the language. Then we extensively explained the internal structure of our open-source morphological analyzer toolkit which is designed to deal with such challenges. Finally, we analyzed the currently available morphological analyzer resources in the Turkish literature and reported the functional and performance-wise comparisons we have made.

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for Turkic languages. *Structure* 10:1–5.

Çagrı Çöltekin. 2010. A freely available morphological analyzer for Turkish. In *LREC*. volume 2, pages 19–28.

Çagrı Çöltekin. 2015. Turkish nlp web services in the weblicht environment. In *Proceedings of the CLARIN Annual Conference*.

Muhammet Şahin, Umut Sulubacak, and Gülsen Eryigit. 2013. Redefinition of Turkish morphology using flag diacritics. In *Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013), Phuket, Thailand, October*.

E.T. Erguvanlı. 2015. *The Phonology and Morphology of Turkish*. Boğaziçi University Press, İstanbul, Turkey.

Gülşen Eryiğit. 2014. ITU Turkish nlp web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 1–4.

A. Göksel and C. Kerslake. 2005. *Turkish: A Comprehensive Grammar*. Routledge, New York, USA.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities* 36(4):381–410.

Marie Hinrichs, Thomas Zastrow, and Erhard W Hinrichs. 2010. Weblicht: Web-based lrt services in a distributed escience infrastructure. In *LREC*.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 29–32.

Lauri Karttunen et al. 1983. Kimmo: a general morphological processor. In *Texas Linguistic Forum*. volume 22, pages 163–186.

J. Kornfilt. 1997. *Turkish*. Routledge, London, UK.

G. Lewis. 1967. *Turkish Grammar*. Clarendon, Oxford, UK.

Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. 2009. Hfst tools for morphology–an efficient open-source package for construction of morphological analyzers. In *International Workshop on Systems and Frameworks for Computational Morphology*. Springer, pages 28–47.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics* 23(2):269–311.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and linguistic computing* 9(2):137–148.

Muhammet Şahin. 2013. *Itumorph: Türkçe İçin Daha Geniş Kapsamlı Ve Başarılı Bir Biçimbilimsel Çözümleyici*. Master's thesis, Fen Bilimleri Enstitüsü.

Hasim Sak. 2011. *Integrating morphology into automatic speech recognition: morpholexical and discriminative language models for Turkish*. Ph.D. thesis, PhD thesis, Boğaziçi University, Istanbul.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *International Conference on Natural Language Processing*. Springer, pages 417–427.

R. Underhill. 1976. *Turkish Grammar*. Cambridge University Press, Oxford, UK.

Self-Attention Networks for Intent Detection

Sevinj Yolchuyeva, Géza Németh, Bálint Gyires-Tóth

Department of Telecommunications and Media Informatics,
Budapest University of Technology and Economics, Budapest, Hungary
{syolchuyeva, nemeth, toth.b}@tmit.bme.hu

Abstract

Self-attention networks (SAN) have shown promising performance in various Natural Language Processing (NLP) scenarios, especially in machine translation. One of the main points of SANs is the strength of capturing long-range and multi-scale dependencies from the data. In this paper, we present a novel intent detection system which is based on a self-attention network and a Bi-LSTM. Our approach shows improvement by using a transformer model and deep averaging network-based universal sentence encoder compared to previous solutions. We evaluate the system on Snips, Smart Speaker, Smart Lights, and ATIS datasets by different evaluation metrics. The performance of the proposed model is compared with LSTM with the same datasets.

1 Introduction and Related Work

Spoken dialogue systems are agents that are intended to help users to access information efficiently by speech interactions (Liu, et al., 2006). In doing so, spoken dialogue systems categorize most of the major fields of spoken language technology, including speech recognition and speech synthesis, language processing, and dialogue system (McTear, 2002). There are different areas of research in the field of spoken dialogue systems. Spoken language understanding (SLU) is one of the essential components of spoken dialogue systems, and it aims to form a semantic frame that captures the semantics of user utterances or queries. Intent detection is one of the main tasks of SLU system. It can be treated as a semantic utterance classification task; since the dialogue system is created to answer a limited range of questions, there is a predefined finite set of intents (Balodis and Deksne, 2019). This task focuses on classifying the user's intent and extracting semantic concepts as constraints for natural language. For example, the utterance "Switch off the garage lights" is related to switching the light off, as shown in Table 1.

Table 1. Example of utterance and a corresponding intent label.

Utterance	**Intent**
Switch off the garage lights.	SwitchLightOff
Get the room brighter, please.	IncreaseBrightness
Skip this song and go on to the next one.	NextSong

Intent detection has been an ongoing field of research in SLU, and similar to most NLP tasks, there are two main approaches to identify the intent of an utterance: rule-based and statistical methods (Hashemi, et al., 2016). The rule-based systems use predefined rules to match new utterances to their intents, and these rules need to be carefully engineered by human experts. Thus, the advancement of these systems requires a huge amount of human effort.

Statistical models, like conditional random field (CRF) and Support Vector Machines, were investigated for this task (Mendoza and Zamora, 2009; Chen et al., 2018). Another important task of SLU is slot filling, which can be formulated as a sequence labelling task. The combination of intent detection and slot filling models was investigated (Mendoza and Zamora, 2009; Kim, 2016).

Furthermore, neural network-based models have also been investigated by (Liu, 2017). Convolutional neural networks (CNN) were applied for classifying intents in (Hashemi, et al., 2016). The combination of CNN and the triangular CRF model (TriCRF) was proposed for the intent labels and the slot filling in (Kim, et al. 2016). During

Proceedings of Recent Advances in Natural Language Processing, pages 1373–1379,
Varna, Bulgaria, Sep 2–4 2019.

training, the features are learned through CNN layers and shared by the intent detection and slot filling tasks. With this approach, for intent detection, the error on the ATIS dataset was 5.91%, and F1-score was 95.42% for slot filling.

In recent years, neural network-based solutions and word embeddings have gained growing popularity for intent detection (Balodis and Deksne, 2019; Kim, et al. 2016). The enriching word embeddings with semantic lexicons can be helpful intent detection, and it is combined with bidirectional LSTM in (Kim, et al., 2016).

The encoder-decoder neural architectures have achieved remarkable success in various tasks (e.g., speech recognition, text-to-speech synthesis and machine translation). This type of networks has also been enhanced with attention mechanism (Xu, et al., 2015; Luong, et al., 2015). Those models have also been used for intent detection and other SLU tasks (Liu and Lane, 2016; Schumann and Angkititrakul, 2018). The combination of attention-based encoder-decoder architecture and alignment-based methods was studied in (Liu and Lane, 2016) for joint intent detection and slot filling.

Self-attention networks (SANs) have shown outstanding performance in various NLP tasks, such as machine translation (Vaswani et al. 2017), and sentiment analysis (Letarte, et al., 2018) stance classification (Xu, et al., 2018; Raheja and Tetreault 2019). It is a special attention mechanism for selecting specific parts of an input sequence by relating its elements at different positions (Vaswani et al. 2017). With a well-designed architecture, SANs are capable of multi-scale modelling. Inspired by (Xu, et al., 2018), we propose the Self-Attention Network (SAN) architecture for intent detection. In our approach, the self-attention is applied to utterances (input), and it is combined with Bi-LSTM (or LSTM). For evaluation, we used Natural Language Understanding benchmark dataset (Snips) (Goo, et al., 2018), Smart Speaker and Smart Lights dataset (Saade, et al., 2018), and ATIS (Hemphill, et al., 1990). We show the effectiveness of this approach in different experimental settings. The application of pre-trained Word2vec (Mikolov, et al., 2013), and FastText (Bojanowski, et al., 2017) embeddings also helps to get competitive results. The remaining part of the paper is organized as follows. In Section 2, we introduce word embedding methods. Section 3 presents the proposed approach. In Section 4, we describe the datasets, which were used in this work and discuss the experimental setup and the results.

2 Word Embedding

Word embeddings map the words to vectors of real numbers. This approach has been widely used as the inputs to neural network-based models for NLP tasks. Word embedding models can be trained with several different tools, such as Word2vec (skip-gram and continuous bag-of-words (CBOW)) (Mikolov, et al., 2013), GloVe (Pennington, et al., 2014), FastText (Bojanowski, et al., 2017). Continuous Bag-of-Words (CBOW) and Continuous Skip-gram models are both powerful techniques for creating word vectors. FastText is one of the recent advances in word embedding algorithms. The main contribution of FastText is to introduce the idea of modular embeddings, which computes a vector for sub-word components, usually n-grams, instead of computing an embedded vector per word. These n-grams are later combined by a simple composition function to compute the final word embeddings. In pre-trained word embedding models, the word embedding tool is trained on large corpora of texts in the given language and highly useful in different NLP tasks. One of the latest embedding methods is Universal Sentence Encoder models (Cer, et al., 2018), which is a form of transfer learning. In (Cer, et al., 2018), it was introduced two encoding models. One of them is based on a Transformer model (TM) and the other one is based on Deep Averaging Network (DAN). They are pre-trained on a large corpus and can be used in a variety of tasks (sentimental analysis, classification, etc.). Both models take a word, sentence or a paragraph as input and generate a 512-dimensional output vector. The transformer-based encoder model targets high accuracy at the cost of greater model complexity and resource consumption (Cer, et al., 2018). But DAN targets performance efficient inference with slightly reduced accuracy.

In this work, we used 300-dimension Word2vec and FastText word embeddings, which were pre-trained on the English Wikipedia corpus. We also investigated TM and DAN based universal encoder models, and each embedding is combined LSTM and Bi-LSTM.

3 Proposed Model

In this section, we introduce two models for intent detection:

1. SAN and LSTM (SAN + LSTM)
2. SAN and Bi-LSTM (SAN + Bi-LSTM)

Both proposed models encode each word to its embedding first. We carried out experiments with different embeddings, as discussed in Section 4.3. As the next step, the contextual information in the input sentences (utterances) is encoded. In the first model, LSTM, in the second one, a Bi-LSTM was used to capture the left and right contexts of each word in the input. In the second model, Bi-LSTM combines two unidirectional LSTM layers that process the input from left-to-right and right-to-left, respectively. Both models are followed by the SAN (see Figure 1), which is based on an attention mechanism for selecting specific parts of a sequence by relating its elements at different positions (Vaswani, et al., 2017). In our work, we only perform input-input attention with self-attention. By using the self-attention, the semantics of the entire utterance can be extracted, and it can be helpful for the better classified. To score attention weight vectors we applied the method of (Xu, et al., 2018).

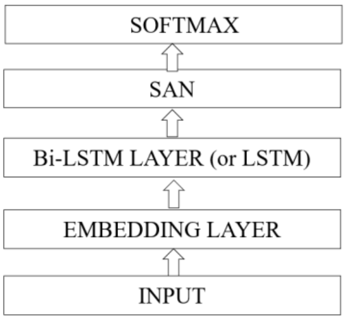


Figure 1. The architecture of the proposed model.

The goal of training is to minimize the loss function. For this purpose, we use multi-class cross-entropy loss,

$$J = -\sum_i \sum_j y_j^i log\hat{y}_j^i + \gamma\|\theta\|^2 \qquad (1)$$

where i is the index of utterance and j is the index of the intent label. γ is the L_2 regularization coefficient and θ is the parameter set. y_j^i is the ground-truth label indicator for i-th utterance, and $\hat{y}_j^i$ is the predicted probability output of i-th utterance. At the output of the network, softmax function was used to predict probabilities.

4 Experiments

4.1. Dataset

We used Natural Language Understanding benchmark dataset (Snips) (Goo, et al., 2018), Spoken Language Understanding research datasets (Saade, et al., 2018) and Airline Travel Information System (ATIS) dataset (Hemphill, et al., 1990). Snips is a balanced dataset and collected from the Snips personal voice assistant; the number of samples for each intent is approximately the same. The training set contains 13,084 utterances, the test and validation (development set) set consist of 700 - 700 utterances. Vocabulary size is 11,241 and intent types are 7, as shown in Table 2.

Table 2. The intent labels and the number of utterances in each label in Snips.

Type of intent	**Number**
PlayMusic	1914
GetWeather	1896
BookRestaurant	1881
RateBook	1876
SearchScreeningEvent	1851
SearchCreativeWork	1847
AddToPlaylist	1818

Smart Lights has 6 intents allowing to turn on or off the light or change its brightness or colour, as shown in Table 3. It has a vocabulary size of approximately 400 words. Smart Speaker dataset has 9 intents and vocabulary size is approximately 1,270. The number of utterances in each intent label is presented in Table 4. In these two datasets, we have split the data into 90 % training and 10% test sets. The validation dataset consists of 10% proportion of the training set.

Table 3. The intent labels and the number of utterances in each label in Smart Lights.

Type of intent	**Number**
Decrease Brightness	296
Increase Brightness	296
Set Light Brightness	296
Set Light Color	306
Switch Light Off	299
Switch Light On	278

Table 4. The intent labels and the number of utterances in each label in Smart Speaker.

Type of intent	Number
GetInfos	199
NextSong	200
PlayMusic	1508
PreviousSong	199
ResumeMusic	200
SpeakerInterrupt	172
VolumeDown	215
VolumeSet	100
VolumeUp	260

ATIS contains audio recordings of people making flight reservations. The training set contains 4,478 utterances, the test set contains 893 utterances; 500 utterances were used as development set. The intent types in ATIS are unbalanced. For example, the intent atis_flight equals about 73.8 % of the training data, while the number of some intents were less than 10.

4.2 Training setup

Data preprocessing may include data normalization, tokenization, lower-casing, removal of punctuation, grammar correction, feature extraction etc., by depending on the task and given dataset. We have done tokenization, have removed punctuation and have converted the numbers to words for all investigated datasets. The word embeddings are initialized with the pretrained 300-dimension Word2Vec or FastText word vectors and these are fixed during training. We also investigated TA and DAN Universal Encoder model-based utterance vectors. The number of units in LSTM and Bi-LSTM is 64.

The L2-regularization coefficient λ in the loss is 0.01.

ADAM (Kingma and Ba, 2015) is used as the optimizer, with a learning rate of 0.001, and with the baseline values of β1, β2 and ε (0.9, 0.999 and 1e-08, respectively). The batch size is 16, the number of epochs is 25.

4.3 Evaluation and Results

We evaluated the performance of the models by accuracy, precision, recall, F1-score. The results are presented in Table 5.

By micro and macro averaged, overall F1-scores were computed, and their average was used (Sokolova and Lapalme, 2009). In Table 5, the first column describes the proposed models, and other columns show the overall accuracy and F1-score for each dataset.

For all datasets, SAN + Bi-LSTM consequently have shown better results than SAN + LSTM, as expected. For Snips, the accuracy of FastText + SAN +Bi-LSTM and TM +SAN +Bi-LSTM is almost the same. The result of Word2Vec + SAN + Bi-LSTM, FastText + SAN + Bi-LSTM, and TM + SAN + Bi-LSTM is almost the same for Smart Speaker. The lowest accuracy score for Smart Lights was produced by DAN + SAN + LSTM, which is 90.2%. The highest accuracy score for ATIS was produced by TM + SAN + Bi-LSTM, which is 96.81. This result is comparable with (Goo, et al., 2018; Hakkani-Tür, et al., 2016). We observed that TM based universal encoder can help to improve accuracy.

Furthermore, Figure 2 and Figure 3 show the confusion matrix of Smart Lights and Snips test dataset by using DAN and TM universal encoder vectors with SAN + Bi-LSTM. SAN + Bi-LSTM based DAN correctly classified 30 intents labels out of 31 for SetLightBrightness and 29 tokens out of 30 for SwitchLightOff, which is the same in SAN + Bi-LSTM based TM.

Table 5. Result of proposed models

Model	Snips		Smart Lights		Smart Speaker		ATIS	
	Acc(%)	F1-s.	Acc (%)	F1-s.	Acc(%)	F1-s.	Acc(%)	F1-s.
Word2Vec + SAN + LSTM	94.2	0.94	91.8	0.90	94.9	0.94	93.93	0.92
FastText + SAN + LSTM	94.6	0.94	92.1	0.92	95.1	0.95	94.51	0.94
DAN + SAN + LSTM	94.1	0.94	90.2	0.90	91.7	0.90	93.56	0.93
TM + SAN + LSTM	94.2	0.94	93.6	0.93	94.2	0.93	94.81	0.94
Word2Vec+SAN+Bi-LSTM	95.6	0.96	93.8	0.94	97.7	0.98	94.49	0.93
FastText + SAN + Bi-LSTM	96.1	0.96	93.4	0.92	97.7	0.97	95.77	0.94
DAN + SAN + Bi-LSTM	94.2	0.94	93.2	0.93	94.7	0.95	94.91	0.93
TM + SAN + Bi-LSTM	96.5	0.97	96.6	0.97	97.7	0.98	96.81	0.95

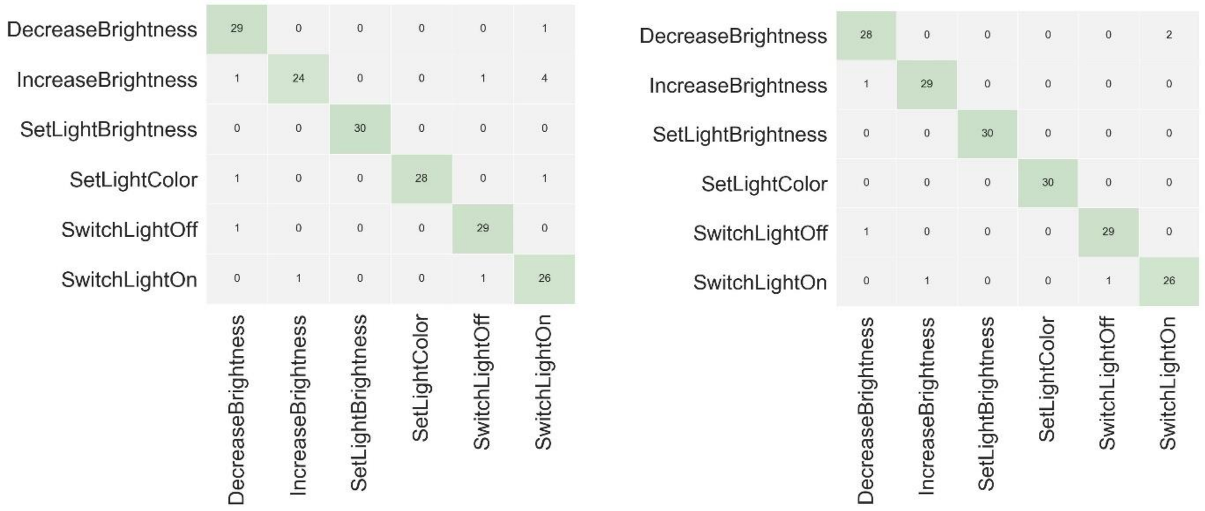


Figure 2. Confusion matrix of Smart Light test dataset by using DAN and TM Universal encoder vectors with SAN + Bi-LSTM.

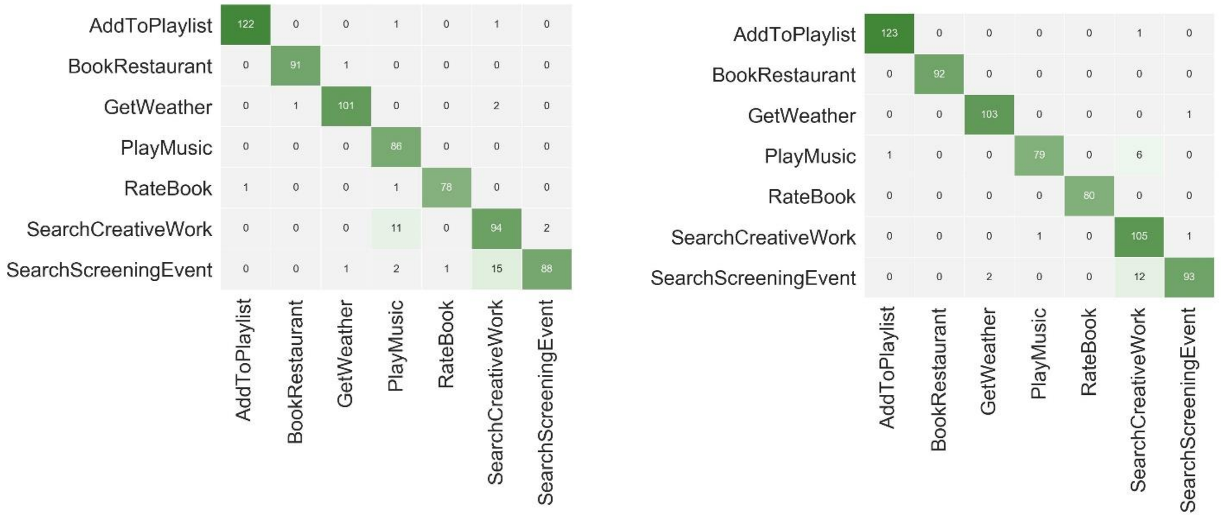


Figure 3. Confusion matrix of Snips test dataset by using DAN and TM Universal encoder vectors with SAN + Bi-LSTM.

The intent of IncreaseBrightness was predicted correctly in case of 24 out of 30, while 4 intent labels were misclassified to the SwitchLightOn by SAN + Bi-LSTM based DAN.

For Snips, SAN + Bi-LSTM based DAN correctly classified 88 intent labels out of 105 SearchScreeningEvent, while the TM-based approach classified 93 intent labels correctly. The AddToPlaylist, GetWeather, RateBook labels achieved almost the same accuracy from both models. SearchCreativeWork intent labels were better predicted by TM based SAN + Bi-LSTM.

As reasons for the misclassification are that some words can belong to both intent classes, depending on the context, and the size of training data is not large enough.

Conclusion

In this paper, the combination of SAN and Bi-LSTM for intent detection were proposed. 300-dimensional Word2Vec and FastText embeddings pretrained on English Wikipedia were used as word representations. Utterance vectors of DAN and TM based Universal sentence encoders were investigated. The results were evaluated with the help of accuracy and confusion matrices. Experiments were also carried out with SAN + LSTM, however, the accuracy was worse than with SAN + Bi-LSTM.

Generally, comparison of these models shows that SAN + Bi-LSTM with TM embeddings performs better than other models on all the investigated datasets. In the future, we would like to carry out more comprehensive analysis and investigate other attention mechanisms such as directional self-attention and bi-directional block self-attention (Shen, et al., 2018) for this task.

Acknowledgements

The research presented in this paper has been supported by János Bolyai Research Scholarship of the Hungarian Academy of Sciences, by Doctoral Research Scholarship of Ministry of Human Resources in the scope of New National Excellence Program, by the BME-Artificial Intelligence FIKP grant of Ministry of Human Resources (BME FIKP-MI/SC) and by the AI4EU project. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Alaa Saade, Alice Coucke, Alexandre Caulier, Joseph Dureau, Adrien Ball, Théodore Bluche, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet. 2018. Spoken Language Understanding on the Edge. CoRR, abs/1810.12735.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin 2017. *Attention Is All You Need.* 31st Conference on Neural Information Processing Systems.

Baosong Yang, Jian Li, Derek Wong, Lidia S. Chao, Xing Wang, Zhaopeng Tu. 2019. Context-Aware Self-Attention Networks. Thirty-Third AAAI Conference on Artificial Intelligence.

Bing Liu, Ian Lane. 2016. Attention-based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. Proceedings of the 17th Annual Meeting of the International Speech Communication Association, 685-689.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. DARPA speech and natural language workshop, 96–101.

Chang Xu, Cecile Paris, Surya Nepal, Ross Sparks. 2018. Cross-Target Stance Classification with Self-Attention Networks. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 778-783.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-Gated Modeling for Joint Slot Filling and Intent Prediction. Proceedings of Annual Conference North American Chapter of the Association for Computational Linguistics, 753-757.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, Philip S. Yu. 2018. Zero-shot User Intent Detection via Capsule Neural Networks. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 3090–3099.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, Ray Kurzweil. 2018. Universal Sentence Encoder. CoRR, abs/1803.11175.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, Ye-Yi Wang. 2016. Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM. In Proceedings of the 17th Annual Meeting of the International Speech Communication Association, 715-719.

Gaël Letarte, Frédérik Paradis, Philippe Giguère, François Laviolette. 2018. Importance of Self-Attention for Sentiment Analysis. Proceedings of the 2018 EMNLP Workshop Blackbox NLP: Analyzing and Interpreting Neural Networks for NLP, 267–275.

Homa B. Hashemi, Amir Asiaee, Reiner Kraft. 2016. Query Intent Detection using Convolutional Neural Networks. In International Conference on Web Search and Data Mining, Workshop on Query Under-standing.

Jeffrey Pennington, Richard Socher, Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 1532–1543.

Jianyi Liu, Jinghua Wang, Cong Wang. 2006. Spoken Language Understanding in Dialog Systems for Olympic Game Information. 2006 4th IEEE International Conference on Industrial Informatics, 1042-1045.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, Ye-Yi Wang .2016. Intent detection using semantically enriched word embeddings. 2016 IEEE Spoken Language Technology Workshop (SLT), 414-419.

Kaspars Balodis and Daiga Deksne. 2019. FastText-Based Intent Detection for Inflected Languages, Information, 10(5), 161.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard Szemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, 2048-2057.

Marcelo Mendoza, Juan Zamora. 2009. Identifying the Intent of a User Query Using Support Vector Machines. SPIRE 2009. Lecture Notes in Computer Science, vol 5721.

Marina Sokolova, and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. Information Processing and Management 45, 427–437.

Michael McTear. 2002. Spoken dialogue technology: enabling the conversational user interface. ACM Computer Survey., 34, 90-169.

Minh-Thang Luong, Hieu Pham, Christopher D. Manning 2015. *Effective Approaches to Attention-based Neural Machine Translation*. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 1412-1421.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics (TACL), 5, 135-146.

Raphael Schumann, Pongtep Angkititrakul. 2018. Incorporating ASR Errors with Attention-based, Jointly Trained RNN for Intent Detection and Slot Filling. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),685-689.

Tao Shen, Jing Jiang,Tianyi Zhou, Shirui Pan, Guodong Long, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. The Thirty-Second AAAI Conference on Artificial Intelligence, 5446-5455.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018. Bi-Directional Block Self-Attention for Fast and Memory-Efficient Sequence Modeling. CoRR, abs/1804.00857.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR), 1–12.

Vipul Raheja, Joel Tetreault. 2019. Dialogue Act Classification with Context-Aware Self-Attention. 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Yang Liu, Kun Han, Zhao Tan, Yun Lei. 2017. *Using Context Information for Dialog Act Classification in DNN Framework.* Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2170-2178.

Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, Xiaofei He. 2018. *Dialogue Act Recognition via CRF-Attentive Structured Network.* 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, 225-234.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, Yoshua Bengio. 2017. *A Structured Self-Attentive Sentence Embedding.* 5th International Conference on Learning Representations.

Turkish Tweet Classification with Transformer Encoder

Atıf Emre Yüksel[*]
Boğaziçi University
İstanbul, Turkey

Yaşar Alim Türkmen[*]
Boğaziçi University
İstanbul, Turkey

Arzucan Özgür
Boğaziçi University
İstanbul, Turkey

{ atif.yuksel, yasar.turkmen, arzucan.ozgur }@boun.edu.tr

Ayşe Berna Altınel
Marmara University
İstanbul, Turkey
berna.altinel@marmara.edu.tr

Abstract

Short-text classification is a challenging task, due to the sparsity and high dimensionality of the feature space. In this study, we aim to analyze and classify Turkish tweets based on their topics. Social media jargon and the agglutinative structure of the Turkish language makes this classification task even harder. As far as we know, this is the first study that uses a Transformer Encoder for short text classification in Turkish. The model is trained in a weakly supervised manner, where the training data set has been labeled automatically. Our results on the test set, which has been manually labeled, show that performing morphological analysis improves the classification performance of the traditional machine learning algorithms Random Forest, Naive Bayes, and Support Vector Machines. Still, the proposed approach achieves an F-score of 89.3% outperforming those algorithms by at least 5 points.

1 Introduction

Short-text usage is increasing day by day and we encounter short-text messages on many social media platforms in different forms such as tweets, Facebook status posts, or microblog entries. Twitter is one of the most widely used platforms, where a huge amount of short-texts are produced. More than 500 million tweets are posted on a typical day (Aslam, 2018). People use Twitter in order to produce and reach information faster about the topic they are interested in. Therefore, tweet classification becomes an important task to improve tweet filtering and tweet recommendation.

Traditional machine learning algorithms such as K-Nearest Neighbor, Support Vector Machines (SVM), and Naive Bayes have been widely used for text classification tasks and accuracy levels of over 80% have been reported in the literature for various data sets (Kadhim, 2019). However, obtaining a similar level of success for short-text classification is difficult, since short-texts contain smaller number of words compared to lengthy texts, which makes classifying them effectively a challenging task (Taksa et al., 2007). While a number of studies have been conducted for short-text classification, most of them have addressed the task of English tweet classification (Batool et al., 2013; Selvaperumal & Suruliandi, 2014).

In this paper, we tackle the task of Turkish tweet classification. The grammatical and syntactic features of the Turkish language pose additional challenges for short-text classification. The agglutinative nature of Turkish results in a high number of different word surface forms, since a root word can take many different derivational and inflectional affixes. This leads to the data sparseness problem. We propose a Transformer-Encoder based model for Turkish topic-based tweet classification and compare it with the traditional machine learning algorithms Naive Bayes, SVM, and Random Forest. The results show that morphological analysis enhances the performance of the traditional classification algorithms. However, the Transformer-Encoder model achieves the best F-score performance, even without any morphological analysis. Another contribution of this study is the constructed Turkish tweet data set on nine different topics. The tweet IDs and the corresponding topics are made available for future studies. [1]

The rest of this paper is organized as follows: in Section 2, we give a review of the related work

[*] Equal contribution

[1] The dataset can be obtained by e-mailing the authors.

Proceedings of Recent Advances in Natural Language Processing, pages 1380–1387,
Varna, Bulgaria, Sep 2–4 2019.

on short-text classification, especially for Turkish. In Section 3, the data set creation and the steps of tweet preprocessing are explained in detail. Section 4 describes the proposed Transformer Encoder model and its usage. The experimental results and error analysis are presented in Section 5. Finally, Section 6 includes the conclusion and future works.

2 Related Work

Traditional text classification methods based on the BoW (Bag of words) model (Harris, 1954) suffer from high dimensionality and sparse feature sets, particularly in short-texts. Other limitations of BoW based models are that semantic features are not captured, the positions of the words in the text are not considered, and the words are assumed to be independent from each other.

In order to overcome the weaknesses of BoW, Latent Dirichlet Allocation (LDA) (Blei et al., 2003), where the terms are mapped to distributional representations based on latent topics within documents, has been used for building classifiers that deal with short and sparse text (Phan et al., 2008; Song et al., 2014).

One of the most commonly used algorithms for short text classification is Naive Bayes, which is based on word occurrence and class priors. For example, Kiritchenko & Matwin (2011) used this algorithm on an email dataset and Kim et al. (2006) offered powerful techniques for improving text classification by Naive Bayes. Sriram et al. (2010) extracted different features from short-texts and used these with Naive Bayes to classify them.

Support Vector Machine (SVM) is another commonly used algorithm in short text classification studies. Pointing to the weaknesses of the BoW approach, different kernels have been developed for SVM such as semantic kernels that use TF-IDF (Salton & Buckley, 1988) and its variants that apply different term weighting functions on the term incidence matrix. Wang & Manning (2012) showed that Naive Bayes obtains higher scores than SVM for short-text sentiment classification tasks and the combination of SVM and Naive Bayes outperforms SVM and Naive Bayes for some of the datasets that they used. More relevant to our study, Lee et al. (2011) used SVM for text-based classification of trending topics under 18 classes and reached 61.76% accuracy. They obtained 65.36% accuracy with Multinomial Naive Bayes.

The advances of deep learning in NLP led to the use of Artificial Neural Network (ANN) based models in recent studies. Convolutional Neural Networks (CNNs) and Dynamic CNNs have been applied to text classificatioon and promising results have been obtained (Kim, 2014; Kalchbrenner et al., 2014). In a recent study, Le & Mikolov (2014) successfully added sequential information by using Recurrent and Convolutional Neural Networks for sequential short-text classification.

In this study, we address the task of Turkish short text classification. Turkish is an agglutinative language, which may result in the same word to map to different features when it takes different inflectional affixes or suffixes. Possessive pronouns, tenses, and auxiliary verbs can be encoded as affixes of the words. For this reason, a word may have many different forms and this poses challenges for classical term weighting methods to construct strong relations between the text and its topic. Most prior work on Turkish short text classification is on sentiment analysis. Demirci (2014) used Naive Bayes, SVM and k Nearest Neighbor (kNN) for emotion analysis on Turkish tweets and obtained accuracy levels of up to 69.9%. Similarly, Yelmen et al. (2018) showed that SVM reaches 80% accuracy for sentiment classification of Turkish tweets for GSM operators, whereas an Artificial Neural Network (ANN) results in slightly lower performance. Türkmenoğlu & Tantuğ (2014) compared lexicon based sentiment analysis and machine learning based sentiment analysis on tweet and movie review datasets and concluded that the machine learning based algorithms SVM, Naive Bayes, and decision trees achieve better scores. Yıldırım et al. (2014) combined lexicon and machine learning based methods to improve sentiment analysis of Turkish tweets.

Unlike prior studies on Turkish short text classification that address the task of sentiment analysis, we tackle the task of topic-based tweet classification and create a Turkish tweet data set for nine different topics. We propose using Transformer Encoder architecture for Turkish short-text classification. Transformer Encoder is a recently proposed model that offers a better understanding of the language structure by protecting the semantic values and meanings of word sequences (Vaswani et al., 2017). Furthermore, the positions of the terms in text are taken into account and the terms are not assumed to be independent from each other by using a contextual word embedding representation.

3 Dataset

In this section, we briefly explain how we created the dataset and the main steps of data preprocessing, as well as the importance of lemmatization on Turkish short-text classification.

3.1 Dataset Creation

The dataset includes 164,549 Turkish tweets, written by 74 different users until February 2019. These tweets are retrieved from the users' profiles, who are known experts in their areas and mostly write tweets on the subjects of their expertise. The tweets of each user are automatically labelled with the topic of expertise of the user. The dataset contains nine topics, namely politics, economics & investment, health, technology & informatics, history, literature & film, sports, education & personal growth, and magazine. The selection of topics was made in a similar way the news sites categorise their content.

We observed that some of the tweets of a user could be mislabeled, since a user may also tweet about topics different than his/her area of expertise, which results in noise. In order to ensure that most of the tweets are related to their assigned topics, we selected a random sample of tweets and manually checked the percentage of the correctly labelled ones. The percentage of the correctly labelled tweets was around 80%. That is, the training dataset contains around 20% noise (i.e., incorrectly labeled tweets). We randomly selected 10% of the dataset as a test set. In order to report reliable results, we manually verified and corrected the labels of the test set.

3.2 Data Preprocessing

Preprocessing Turkish sentences is quite different from the English ones, since Turkish is an agglutinative language and we may encounter words in many different forms. In addition, tweets come with their own difficulties when they are used in natural language processing. They contain hashtags, mentions, emojis, and links which make tokenization difficult. To overcome those challenges we follow three main steps for preprocessing as described in the following subsections.

3.2.1 Data Cleaning

Users use some adhoc pattern such as hashtags, mentions, the link of website they want to refer to, and emojis in tweets.In our work, we are only interested in lexical terms in tweets. Hence, we drop the hashtags, mentions, links, emojis, numbers, and punctuations. On the other hand, hashtags can be quite useful when deciding the topic of a tweet. There are some studies to split hashtags (Çelebi & Özgür, 2018) into meaningful words in English. However, it is left as an open future work for Turkish hashtags.

Some tweets contain only response to a tweet like "greetings", "thank you", "okay", or "congratulations" for good news etc. When we examine the tweets which have less than 4 words, nearly all of these tweets are examples of such tweets. These tweets are removed from the dataset, since they are not about any specific topic.

In addition, when we examined the tweets with retweet and like statistics that are significantly different from the others, we discovered that those tweets are mostly retweets of campaigns or calling for help. For this reason, we filtered this type of tweets from the dataset.

3.2.2 Language Identification

In Twitter, users sometimes write and retweet tweets in languages different from their native language. When we analysed our dataset, we observed that the languages used by users are Turkish, English, Arabic, French, and German. For filtering non-Turkish tweets, we used the language identification model in (Lui & Baldwin, 2012).

After the dataset cleaning and language-based filtering steps, the training set contains 119,778 tweets and the test set contains 3050 tweets.

3.2.3 Lemmatization

The goal of lemmatization is to find the lemmas of the words by removing the prefixes, suffixes, and other types of affixes based on morphological analysis. This process is harder for the Turkish language, since it has more types of affixes than English.

In the Naive Bayes model, the frequency of a word is prominent for scoring the importance of the word for each class. Therefore, finding the correct lemmas of the words is a useful transformation to classify tweets correctly. Lemmatization is also important for the SVM classifier, in order to compute the similarity between the instances in the training set, as well as the support vectors and the test instances in the classification step more accurately. Yıldırım et al. (2014) showed the positive impact of morphological analysis for the sentiment analysis of Turkish tweets. Therefore, in order to increase the success of the Naive Bayes and SVM models, we use a lemmatization model (Sak et al., 2008) which is trained by nearly one million Turkish sentences. An example tweet and its lemmatized version are shown in Figure 1. The effects of lemmatization on the success of Turkish tweet classification are presented in Section 5.

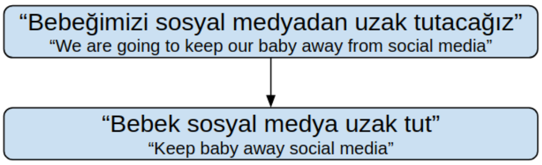

"Bebeğimizi sosyal medyadan uzak tutacağız"
"We are going to keep our baby away from social media"
"Bebek sosyal medya uzak tut"
"Keep baby away social media"

Figure 1: A sample tweet from the dataset and its lemmatized from.

4 Model Description

In this section, the main components of the proposed model are presented. First, the general structure of the input embeddings is explained. Then, the architecture of the Transformer Encoder model is described in detail.

4.1 Input Embeddings

One of the most widely used and robust word embedding models, word2vec, was proposed by Mikolov et al. (2013). Besides word embeddings, sentence embeddings (Logeswaran & Lee, 2018) and paragraph embeddings (Le & Mikolov, 2014) were studied in order to represent larger text snippets correctly and classify documents accordingly. Aiming to construct the word embeddings based on the context of each word, Peters et al. (2018) offered a new word embedding representation model named as ELMo. ELMo tries to keep the contextual features in word embedding representations so that the polysemy and homonymy problems are alleviated.

In our study, pretrained contextual word embedding representations of Turkish words (Devlin et al., 2019) are used. They are obtained by using the bidirectional approach with masked language model (Taylor, 1953) in training. Hence, the representation of each word contains the information of the left and right words in their own context. Moreover, in order to account for the word order in text, positional embeddings (Vaswani et al., 2017) are used and combined with the word embeddings. When using the input embeddings, they are fed into the Transformer Encoder by matching each word in the input tweet with the longest token in the vocabulary of the pretrained contextual word embeddings. An example from the dataset is shown in Figure 2.

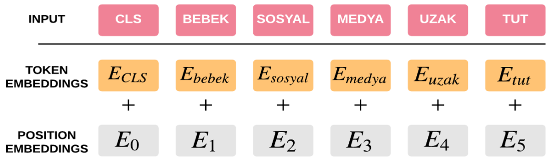


Figure 2: Input embeddings representation.

4.2 Transformer Encoder

This model is based on the Transformer architecture (Vaswani et al., 2017), which contains self-attention layers. What we aim to find in this architecture is reaching the importance of every word in tweets by training the query and key matrices of the attention layers. Hence, the attention score is calculated for every word in a tweet in order to determine its usefulness for each class. Lots of different attention patterns are obtained with multi-head self attention layers, which enable the model to decide which attention score is significant among the pairs of the terms in the tweet.

In general, the transformer architecture is constructed by combining the layers of multi self-attention heads, Dropout, Layer Normalization, and 2 fully-connected layers, respectively. A pictorial representation of this Transformer architecture is shown as a grey block in Figure 3.

In the general architecture of the Transformer Encoder, Layer Normalization and Dropout layers are applied to the contextual word embeddings, into which positional information is added before it enters into the Transformer. In the Transformer, multi-head self attention layers are used as we mentioned above. After the Transformer obtains the attention score matrices containing the score of every word with different attention patterns, these inter-layer features are connected into 2 fully-connected layers in order to reach the combination of different attention scores of the terms. Then, the class prediction of the tweet is found after passing the pooler, dropout, and the last fully-

Model	Accuracy	Precision	Recall	F1-Score
Random Forest (Baseline)	65.0	64.9	64.5	64.5
Random Forest + Lemmatization	71.1	70.9	70.9	70.7
Random Forest + Lemmatization + TF-IDF	69.9	69.8	69.6	69.6
Naive Bayes	82.7	82.4	82.4	82.6
Naive Bayes + Lemmatization	85.0	84.8	84.7	84.9
Naive Bayes + Lemmatization + TF-IDF	85.4	85.3	85.4	85.2
SVM	78.6	78.2	78.3	78.1
SVM + Lemmatization	80.3	80.3	80.2	80.1
SVM + Lemmatization + TF-IDF	84.4	84.2	84.3	84.2
Transformer Encoder + Input Embeddings	**89.4**	**89.3**	**89.4**	**89.3**

Table 1: Comparison of the models' scores over the manually labeled test set.

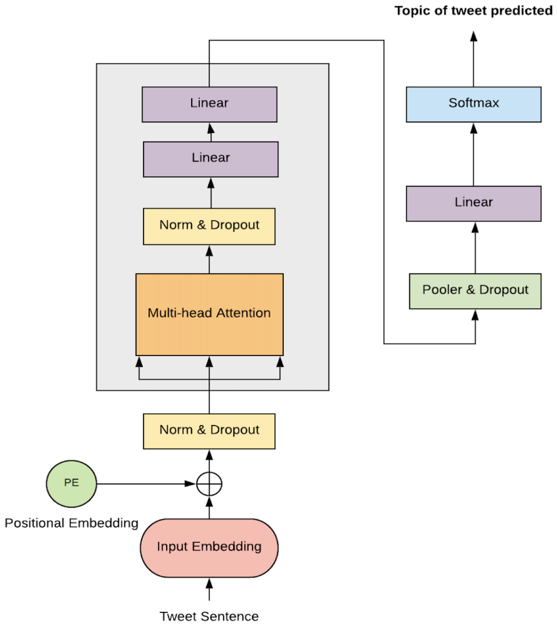


Figure 3: The architecture of the Transformer Encoder model.

connected layer. The class probabilities are calculated by using the Softmax classifier. The general architecture of the model is shown in Figure 3.

5 Experiments

We compared the performance of the Transformer Encoder model with three different baseline models, which are widely used in the area of short-text classification: Naive Bayes, SVM and Random Forest. The parameters of the models are tuned using cross-validation over the training set and the performance results are reported over the manually labeled test set (Table 1).

In the training phase of the Transformer Encoder, 10 epochs are run on the dataset. Batch size of training data is selected as 8 due to computational constraints. Maximum sequence length is 128, learning rate is equal to 2e-5, and the number of attention heads is equal to 12 in our configuration.

For Naive Bayes, SVM and Random Forest the most frequent 15000 words are selected as the features to represent the data. We also experimented with using the most frequent 10000, 20000, and 25000 words, however, the results in the cross-validation experiments were lower. So, we report the results with 15000 words over the test set. We also investigated the effect of lemmatization and TF-IDF weighting on the baseline models.

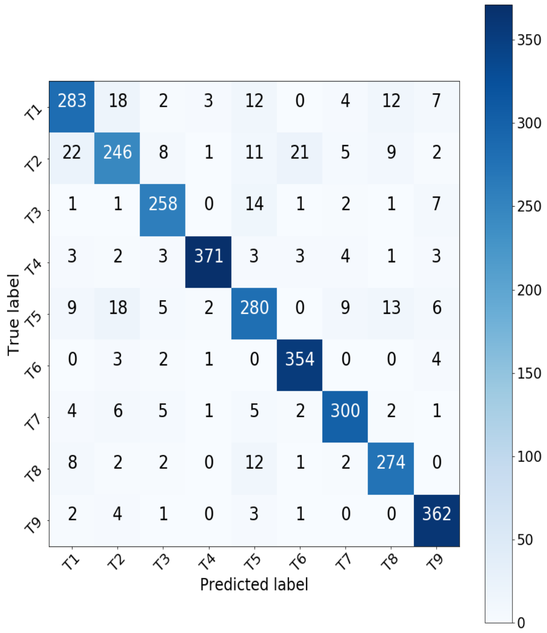
5.1 Comparison Results

The results of the compared models over the test set are shown in Table 1. Naive Bayes, Random Forest, and SVM obtain better scores when morphological analysis is performed.

However, in spite of these improvements, the proposed Transformer Encoder model achieves the highest scores in all metrics for tweet classification even though it does not use the morphological analysis. These results point out that using pretrained word embeddings, which are trained on large datasets, with a Transformer Encoder architecture is more effective than using morphological analysis and TF-IDF based term weighting with the traditional machine learning algorithms Random Forest, Naive Bayes, and SVM for Turkish short text classification.

5.2 Error Analysis

Figure 4 shows the confusion matrix for the Transformer Encoder model over the test set.



	T1	T2	T3	T4	T5	T6	T7	T8	T9
T1	283	18	2	3	12	0	4	12	7
T2	22	246	8	1	11	21	5	9	2
T3	1	1	258	0	14	1	2	1	7
T4	3	2	3	371	3	3	4	1	3
T5	9	18	5	2	280	0	9	13	6
T6	0	3	2	1	0	354	0	0	4
T7	4	6	5	1	5	2	300	2	1
T8	8	2	2	0	12	1	2	274	0
T9	2	4	1	0	3	1	0	0	362

Figure 4: Confusion matrix for the Transformer Encoder. The names of the topics (from T1 to T9) are literature & film, education & personal growth, economy & investment, magazine, politics, health, sport, history, and technology & informatics, respectively.

It is observed that most of the errors are caused by the proximity of the topics like politics and history or literature & film and education & personal growth. The main reason is that there are many common words which are frequently used in close topics. In many cases, it is hard to differentiate a tweet about recent history from a tweet about politics. Similarly, an expression from a novel can be similar to an expression written by an educator. For example[1], the topic of the tweet "*Bachmann is so right, the real death of a man is not from diseases, but from what another man does to him.*" is predicted by the Transformer Encoder model as education & personal growth, whereas its correct label is literature & film. As another misclassified example the topic of the tweet "*The Lausanne Treaty debate should be discussed in public.*" is predicted as politics, whereas this tweet is in the scope of the recent history.

6 Conclusion and Future Works

We proposed using a Transformer Encoder architecture with contextual word embeddings and positional embeddings for Turkish short text classification. In addition, we compiled a dataset of Turkish tweets for topic-based classification. The training set has been labeled automatically using a weakly supervised approach, whereas the test set has been manually labeled for reliable evaluation. Our results demonstrate that the Transformer Encoder model outperforms the widely used machine learning models for topic-based Turkish tweet classification. Also, this study shows the importance of morphological analysis for Turkish short-text classification with the widely used machine learning algorithms SVM, Naive Bayes, and Random Forest. It is interesting to note that, the Transformer Encoder model is able to outperform these algorithms, even though it does not involve a morphological analysis step.

One of the areas of usage of this study is guiding Twitter users and making easier for them to find correct accounts to follow. Twitter is one of the most used social media platforms in the world in order to get news and learn about what people think (Riquelme & Gonzlez-Cantergiani, 2016). Users may prefer Twitter for getting breaking news or reading about a social event. Therefore, it is necessary to find the correct people to follow and be aware of the popular and informative tweets. By using our model, Turkish tweets can be categorized and the most relevant ones can be suggested to users according to their interests, which would lead to saving time and manual effort.

References

Aslam, S. (2018). Twitter by the numbers: Stats, demographics fun facts. `https://www.omnicoreagency.com/twitter-statistics`. Accessed: 2019-01-06.

Batool, R., Khattak, A. M., Maqbool, J., & Lee, S. (2013). Precise tweet classification and sentiment analysis. In *2013 IEEE/ACIS 12th International*

[1] The original tweets are in Turkish. Their English translations are provided here for easier comprehension.

Conference on Computer and Information Science (ICIS), (pp. 461–466).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*, 993–1022.

Çelebi, A. & Özgür, A. (2018). Segmenting hashtags and analyzing their grammatical structure. *Journal of the Association for Information Science and Technology*, *69*.

Demirci, S. (2014). Emotion analysis on Turkish tweets. Master's thesis, Middle East Technical University.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (pp. 4171–4186)., Minneapolis, Minnesota. Association for Computational Linguistics.

Harris, Z. S. (1954). Distributional structure. *WORD*, *10*(2-3), 146–162.

Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, *52*(1), 273–292.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (pp. 655–665)., Baltimore, Maryland. Association for Computational Linguistics.

Kim, S. B., Han, K. S., Rim, H. C., & Myaeng, S. H. (2006). Some effective techniques for Naive Bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, *18*(11), 1457–1466.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (pp. 1746–1751)., Doha, Qatar. Association for Computational Linguistics.

Kiritchenko, S. & Matwin, S. (2011). Email classification with co-training. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '11, (pp. 301–312). IBM Corp.

Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, ICML'14, (pp. II–1188–II–1196). JMLR.org.

Lee, K., Palsetia, D., Narayanan, R., Patwary, M., Agrawal, A., & Choudhary, A. (2011). Twitter trending topic classification. *2011 11th IEEE International Conference on Data Mining Workshops*.

Logeswaran, L. & Lee, H. (2018). An efficient framework for learning sentence representations. *CoRR*, *abs/1803.02893*.

Lui, M. & Baldwin, T. (2012). Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, (pp. 25–30)., Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances In Neural Information Processing Systems*, (pp. 3111–3119).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Phan, X., Nguyen, L., & Horiguchi, S. (2008). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, (pp. 91–100)., New York, NY, USA. ACM.

Riquelme, F. & Gonzlez-Cantergiani, P. (2016). Measuring user influence on Twitter: A survey. *Information Processing & Management*, *52*(5), 949–975.

Sak, H., Güngör, T., & Saraçlar, M. (2008). Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In Nordström, B. & Ranta, A. (Eds.), *Advances in Natural Language Processing*, (pp. 417–427)., Berlin, Heidelberg. Springer Berlin Heidelberg.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513 – 523.

Selvaperumal, P. & Suruliandi, A. (2014). A short message classification algorithm for tweet classification. In *2014 International Conference on Recent Trends in Information Technology*, (pp. 1–3).

Song, G., Yunming, Y., Xiaolin, D., Huang, X., & Shifu, B. (2014). Short text classification: A survey. *Journal of Multimedia*, *9*(5), 635.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (2010). Short text classification in Twitter to improve information filtering. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Taksa, I., Zelikovitz, S., & Spink, A. (2007). Using web search logs to identify query classification terms. *International Journal of Web Information Systems*, *3*, 315–327.

Taylor, W. L. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Bulletin*.

Türkmenoğlu, C. & Tantuğ, A. C. (2014). Sentiment analysis in Turkish media. In *Proceedings of Workshop on Issues of Sentiment Discovery and Opinion Mining, International Conference on Machine Learning (ICML)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances In Neural Information Processing Systems*, (pp. 5998–6008).

Wang, S. & Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, (pp. 90–94)., Stroudsburg, PA, USA. Association for Computational Linguistics.

Yelmen, İ., Zontul, M., Kaynar, O., & Sönmez, F. (2018). A novel hybrid approach for sentiment classification of Turkish tweets for GSM operators. *International Journal of Circuits, Systems and Signal Processing*.

Yıldırım, E., Çetin, F. S., Eryiğit, G., & Temel, T. (2014). The impact of NLP on turkish sentiment analysis. *Turkiye Bilisim Vakfi Bilgisayar Bilimleri ve Muhendisligi Dergisi*, *7*, 43 – 51.

Multilingual Dynamic Topic Model

Elaine Zosa and **Mark Granroth-Wilding**
Department of Computer Science
University of Helsinki
Helsinki, Finland
firstname.lastname@helsinki.fi

1 Abstract

Dynamic topic models (DTMs) capture the evolution of topics and trends in time series data. Current DTMs are applicable only to monolingual datasets. In this paper we present the multilingual dynamic topic model (ML-DTM), a novel topic model that combines DTM with an existing multilingual topic modeling method to capture cross-lingual topics that evolve across time. We present results of this model on a parallel German-English corpus of news articles and a comparable corpus of Finnish and Swedish news articles. We demonstrate the capability of ML-DTM to track significant events related to a topic and show that it finds distinct topics and performs as well as existing multilingual topic models in aligning cross-lingual topics.

2 Introduction

Dynamic topic models (DTMs, Blei and Lafferty, 2006) capture themes or topics discussed in a set of time-stamped documents and how the words related to these topics change in prominence over time. Other topic models have been proposed that aim to model time series data (Wang and McCallum, 2006; Wei et al., 2007; Hall et al., 2008). These models can be used to explore historical document collections to study historical trends, language changes (Frermann and Lapata, 2016) and track the emergence and evolution of certain subjects (Hall et al., 2008; Yang et al., 2011).

With the internet becoming more multilingual it is increasingly important to build cross-lingual tools to bridge different linguistic groups online. Fortunately, large multilingual datasets such as Wikipedia, the Europarl parallel corpus (Koehn, 2005) and other datasets assembled from crawling the web (Van Gael and Zhu, 2007) are also becoming widely available to researchers. This has led to the development of several multilingual topic models to infer topics from multilingual datasets. Examples include the polylingual topic model (PLTM, Mimno et al., 2009), multilingual topic model for unaligned text (MuTo, Boyd-Graber and Blei, 2009), and JointLDA (Jagarlamudi and Daumé, 2010). What is currently lacking are topic models for multilingual time-stamped data that can model historical and linguistic changes in a specific context. Digitalization efforts in libraries and archives, such as the Europeana collections[1], have made available online historical document collections from different European countries. Collections such as these are valuable resources for comparing historical trends in different countries. However, scholars and other interested parties may not possess the linguistic skills necessary to explore such data and would benefit from tools to automatically discover connections across linguistic boundaries.

In this paper, we present the multilingual dynamic topic model (ML-DTM), a novel topic model that captures dynamic topics from broadly topically aligned multilingual datasets. We extend a DTM inference method by Bhadury et al. (2016) to train this model.

In the following sections, we give a broad review of related work, discuss existing *dynamic* and *multilingual* topic models in more detail, and then give a description of our proposed combined model. We then demonstrate usage of this model on a parallel dataset and a comparable dataset of news articles and present our results. We show that this novel topic model learns aligned bilingual topics as demonstrated by the cosine similarities of learned vector representations of named entities. Table 1 summarizes the notations used in this paper. Code is available at: `https://github.com/ezosa/multilingual_dtm`.

[1] https://www.europeana.eu

Symbol	Description
α	parameter for θ
β	hyperparameter for ϕ
ψ	hyperparameter for θ
θ	distribution of topics over a document
ϕ	distribution of words over a topic
D	set of documents
W_d	words in document d
N_d	number of words in document d, or $\|W_d\|$
Z_d	topic assignments of words in document d
K	number of topics
T	number of time slices
L	number of languages in the dataset
V	words in a vocabulary for language

Table 1: Summary of notations.

3 Related Work

Topic models capture themes inherent in document collections through the co-occurence patterns of the words in documents. Latent Dirichlet Allocation (LDA, Blei et al., 2003) is a popular method for inferring these themes or topics. It is generative document model where a document is described by a mixture of different topics and each topic is a probability distribution over the words in the vocabulary. In a document collection we can only observe the *words* in a document. Therefore, training a model involves inferring these latent variables through approximate inference methods.

In the case of documents with timestamps covering some time interval, such as news articles, we might want to capture *dynamic* co-occurence patterns that evolve through time. Dynamic Topic Model (DTM, Blei and Lafferty, 2006) divides time into discrete slices and chains parameters from each slice in order to infer topics that are aligned across time. DTM gives us a set of topic-term distributions that evolve from one time slice to the next. There are also other topic models for time-series data such as the Continuous Dynamic Topic Model (cDTM, Wang et al., 2008), a version of DTM that does not explicitly discretize time intervals. Dynamic Mixture Model (DMM, Wei et al., 2007) captures the evolution of documents across time and Topics over Time (TOT, Wang and McCallum, 2006) is a method that models the prominence of topics over time.

A limitation of LDA, as well as these dynamic models, is that it is not applicable to multilingual data. LDA captures co-occurences of words in documents and words from different languages would rarely, if ever, occur in the same document regardless of their semantics, as demonstrated by experiments on the Europarl corpus (Jagarlamudi and Daumé, 2010; Boyd-Graber and Blei, 2009). Multilingual topic models are developed to capture cross-lingual topics from multilingual datasets.

Polylingual Topic Model (PLTM, Mimno et al., 2009) is a multilingual topic model that extends LDA for an aligned multilingual corpus. Instead of running topic inference on individual documents as in LDA, PLTM infers topics for *tuples* of documents, where each document in the tuple is in a different language. PLTM assumes that the documents of a tuple discuss the same subject broadly and therefore share the same document-topic distribution.

Other topic models for multilingual data include Multilingual Topic Model for Unaligned Text (MuTo, Boyd-Graber and Blei, 2009) and JointLDA (Jagarlamudi and Daumé, 2010). MuTo attempts to match words between languages in the corpus and samples topic assignments for these matchings. JointLDA is a multilingual model that does not require an aligned corpus but requires a bilingual dictionary and uses concepts, instead of words, to infer topics where concepts can be entries in the bilingual dictionary.

In this work we will focus on DTM and PLTM because we want to capture topic evolution in multilingual settings without using additional lexical resources such as dictionaries.

3.1 Dynamic Topic Model

LDA uses Dirichlet and multinomial distributions for inferring both topic-term distributions ϕ and document-topic distributions θ. The conjugacy of these distributions allow ϕ and θ to be integrated out leaving us only with the posterior distribution for topic-term assignments Z, which we can sample through Gibbs sampling (Griffiths and Steyvers, 2004). Inference in DTM, however, is

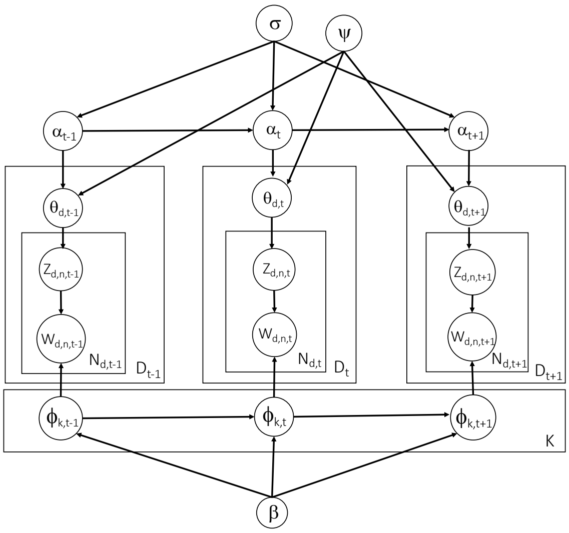


Figure 1: DTM for three time slices as shown in Bhadury et al. (2016).

more complicated due to the non-conjugacy of the distributions used in the model. Blei and Lafferty (2006) use variational Kalman filtering for topic inference, which does not scale well for a large number of topics and documents and large numbers of time slices (Bhadury et al., 2016; Wang et al., 2008). Bhadury et al. (2016) developed a method for inferring the posterior distributions of DTM with Gibbs sampling. In their method, the parameters α, θ, ϕ and Z are re-sampled during every iteration of the sampler.

The document-topic proportions θ, sampled for each document in each time slice, and the topic-term distributions ϕ, sampled for each topic in each time slice, are updated using Stochastic Gradient Langevin Dynamics (SGLD, Welling and Teh, 2011) which is based on Stochastic Gradient Descent (SGD). Figure 1 shows the plate diagram for DTM from Bhadury et al. (2016).

3.2 Polylingual Topic Model

The polylingual topic model (PLTM, Mimno et al., 2009) is an extension of LDA that infers topics from an aligned multilingual corpus composed of document tuples. Tuples are composed of documents in different languages that are thematically aligned, meaning that they discuss the subject in broadly similar ways. For instance, a news article in German and another article in English that report on the same event can compose a tuple.

Inference on PLTM can be done via Gibbs sampling where the topic assignment of each term $z^l_{d,n}$ is resampled during every iteration. Following Vulić et al. (2015), we provide the update formulae for the bilingual case for brevity. The update formulae for documents in languages x and y are:

$$P(z^x_{d,n}=k|z^x,z^y,w^x,w^y,\alpha,\beta)\propto \frac{m^x_{d,k}-1+m^y_{d,k}+\alpha}{\sum_{i=1}^K m^x_{d,i}-1+\sum_{i=1}^K m^y_{d,i}+K\alpha}\cdot \frac{v^x_{k,w_{d,n}}-1+\beta}{\sum_{i=1}^{|V^x|}v^x_{k,w_{d,i}}-1+|V^x|\beta} \qquad (1)$$

$$P(z^y_{d,n}=k|z^y,z^x,w^y,w^x,\alpha,\beta)\propto \frac{m^y_{d,k}-1+m^x_{d,k}+\alpha}{\sum_{i=1}^K m^y_{d,i}-1+\sum_{i=1}^K m^x_{d,i}+K\alpha}\cdot \frac{v^y_{k,w_{d,n}}-1+\beta}{\sum_{i=1}^{|V^y|}v^y_{k,w_{d,i}}-1+|V^y|\beta} \qquad (2)$$

where $m^x_{d,k}$ is the number of times topic k has been assigned to a word in document d written in language x and $v^x_{k,w_{d,n}}$ is the number of times word $w_{d,n}$, that is, the word at position n in document d, has been assigned to topic k. $|V^x|$ is the vocabulary size of language x. The first part of these formulae links the two languages together and is language-independent while the second part is language-specific.

Figure 2 shows the graphical representation of PLTM for l languages.

4 Multilingual Dynamic Topic Model

Here we combine the above *dynamic* and *polylingual* models to produce a *Multilingual Dynamic Topic Model* (ML-DTM). Figure 3 shows the diagram of ML-DTM for two languages and three time slices. Although we show only the bilingual case here for brevity, the model is applicable for any number of languages.

The inference method of Bhadury et al. (2016) was originally motivated by the need to speed up DTM inference for very large datsets. We apply it here to the combined ML-DTM model. We propose the following posterior conditional distribution for $\theta_{x,t}$ where x is a tuple index in the dataset:

$$p(\theta_{x,t}|\alpha_t,Z_{x,t})\propto \mathcal{N}(\theta_{x,t}|\alpha_t,\psi^2 I)\times \prod_{l=1}^{L}\prod_{n=1}^{N_{d_l,t}} Mult(Z_{d_l,n,t}|\pi(\theta_{x,t})) \qquad (3)$$

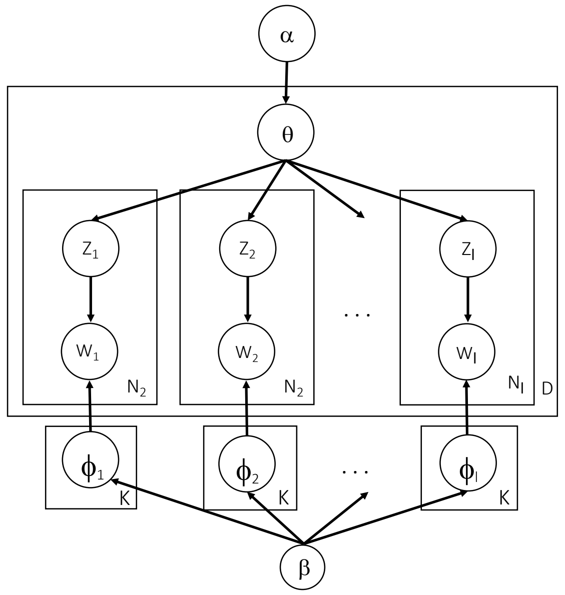


Figure 2: Polylingual topic model for l languages of Mimno et al. (2009).

Following Bhadury et al. (2016), the update equation to evaluate the gradient of $\theta^k_{x,t}$ becomes:

$$\nabla_{\theta^k_{x,t}} \log p(\theta_{x,t}|\alpha_t, Z_{x,t}) = \frac{-1}{\psi^2}(\theta^k_{x,t} - \alpha^k_t) + \sum_{l=1}^{L} C^k_{d_l,t} - \left(N_{d_l,t} \times \frac{\exp(\theta^k_{x,t})}{\sum_j \exp(\theta^j_{x,t})}\right) \quad (4)$$

where $Z_{x,t}$ are the topic assignments for the words in the documents in tuple x at time slice t; $C^k_{d_l,t}$ is the number of times topic k has been assigned to a word in document d_l at time t; and $N_{d_l,t}$ is the length of document d_l at time t.

Instead of evaluating $\theta_{d,t}$ for a single document as in monolingual DTM, we compute $\theta_{x,t}$ for a document *tuple*. The second term in (4) links the languages together by summing up the counts of each document in the tuple.

The equation for evaluating the gradient of the topic-term distributions $\phi_{k,t}$ is the same as in the original paper except that we compute separate distributions for each language since every language has a different vocabulary. This means that for each time slice, instead of updating K different ϕs (one for each topic), we will need to update $K \cdot L$ ϕs. Table 2 shows the dimensions of the parameters to be estimated.

Finally, the topic assignment $Z_{d_l,n,t}$ is sampled as in the original paper:

$$P(Z_{d_l,n,t} = k|\theta_{x,t}, \phi^{w_l}_{k,t}) \propto exp(\theta^k_{x,t})exp(\phi^{w_l}_{k,t}) \quad (5)$$

where w_l is a word from the vocabulary of language l.

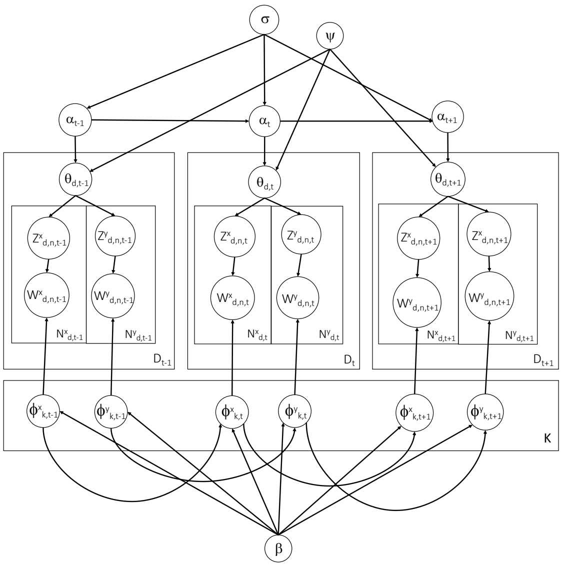


Figure 3: ML-DTM for two languages and three time slices.

Parameter	Dimension
α	$K \times T$
θ	$D^t \times K \times T$
ϕ	$\lvert V^l \rvert \times L \times K \times T$

Table 2: Dimensions of the sampled parameters in the multilingual dynamic topic model (ML-DTM). D^t is the number of document tuples in a dataset.

5 Evaluation

5.1 Datasets

We ran experiments on ML-DTM with two kinds of data: a parallel dataset and a thematically-comparable one.

The DE-NEWS parallel dataset consists of German news articles from August 1996 to January 2000 with English translations done by human volunteers[2]. This dataset covers 42 months with an average of 200 articles per month. Since this is a parallel corpus there is no need to align the articles.

[2] http://homepages.inf.ed.ac.uk/pkoehn/publications/de-news/

For the comparable dataset, we use the YLE news dataset which consists of Finnish and Swedish articles from the Finnish broadcaster YLE, covering news in Finland from January 2012 to December 2018[3]. The Finnish and Swedish articles are written separately and are not direct translations of each other. We use existing methods for aligning comparable news articles (Utiyama and Isahara, 2003; Vu et al., 2009). Specifically, we create an aligned corpus by pairing a Finnish article with a Swedish article published within a two-day window and sharing three or more named entities. We want to have a one-to-one alignment in our dataset such that no article is duplicated, so we pair a Finnish article with the first Swedish article encountered in the dataset that fits the above criteria and remove the paired articles from the unaligned dataset. The unaligned dataset has a total of 604,297 Finnish articles and 228,473 Swedish articles and the final aligned dataset consists of 123,818 articles covering 84 months. A script for aligning articles using the method described is provided in the Github project associated with this work.

We tokenized, lemmatized (using WordNetLemmatizer for German and English and LAS (Mäkelä, 2016) for Finnish and Swedish) and removed stopwords for these two datasets and then used the 5,000 most frequent words of each language as the vocabulary for that language.

5.2 Cross-Lingual Alignment

We compare the cross-lingual alignment of topics of ML-DTM and PLTM by evaluating the similarity of the learned vector representations of named entities (NEs) that appear in both languages of the same dataset. This method is suggested by Vulić et al. (2015) on the basis that NEs tend to be spelled in the same way in different languages and can be expected to have a similar association with topics across languages. The K-dimensional vector of a NE w for language s is thus:

$$vec(w_s) = [P(z_1|w_s), P(z_2|w_s), ..., P(z_K|w_s)] \tag{6}$$

Under an assumption of a uniform prior over topics, this vector can be computed as:

$$\begin{aligned} P(z_k|w_s) &\propto \frac{P(w_s|z_k)}{P(w_s)} \\ &= \frac{\phi_{l,z_k,w_s}}{Norm_{\phi_{s,\cdot,w_s}}} \end{aligned} \tag{7}$$

$$Norm_{\phi_{s,\cdot,w_s}} = \sum_{k=1}^{K} \phi_{s,z_k,w_s} \tag{8}$$

$$vec(w_s) = \frac{[\phi_{l,z_1,w_s}, \phi_{l,z_2,w_s}, ..., \phi_{l,z_K,w_s}]}{Norm_{\phi_{s,\cdot,w_s}}} \tag{9}$$

We then take the cosine similarities between the L different vector representations of the NE (for both datasets, $L = 2$).

We evaluate the cosine similarities of NEs that occur five or more times in each time slice. To make the comparison between PLTM and ML-DTM, we train one ML-DTM model on three time slices for 10 topics and three separate PLTM models for each time slice, also capturing 10 topics. We set $\alpha = 1.0$ and $\beta = 0.08$ for PLTM and $\alpha = 0.5$ and $\beta = 0.5$ for ML-DTM for both datasets, which achieved the best results of a small range of values tried. We did not, for now, perform more extensive optimisation of hyperparameters.

5.3 Topic Diversity

We also measure the *diversity* of the topics ML-DTM finds by computing the Jensen-Shannon (JS) divergence of every topic pair for each time slice for each language and averaging the divergences. Wang and McCallum (2006) used this method, though with KL divergence. It is desirable for the model to find topics that are as distinct as possible from each other.

We compare the diversity of the topics found by ML-DTM, trained as in the previous section, with the topics found by DTM. To make this comparison we train separate DTM models for each language in our two datasets, giving us four different models and compare the divergences of the topics found by these models with their ML-DTM counterparts. We use the Gensim implementation of DTM[4] where we set the chain variance to 0.1 and leave other parameters to be inferred during training. We train both ML-DTM and DTM on 10 time slices for 10 topics.

[3] https://www.kielipankki.fi/corpora/

[4] https://radimrehurek.com/gensim/models/ldaseqmodel.html

Time slice	# of NEs	PLTM	ML-DTM
Aug 1996	53	**0.880**	0.692
Sept 1996	65	0.876	**0.908**
Oct 1996	64	0.840	**0.885**

Table 3: Average cosine similarity of topic vectors for NEs over three time slices in DE-NEWS.

Time slice	# of NEs	PLTM	ML-DTM
Jan 2012	79	0.800	**0.896**
Feb 2012	71	**0.810**	0.796
Mar 2012	72	0.722	**0.745**

Table 4: Average cosine similarity of the vectors of NEs for three time slices in the YLE dataset.

Time slice	DTM English	ML-DTM English
Aug 1996	0.372	**0.655**
Sep 1996	0.368	**0.660**
Oct 1996	0.366	**0.657**
Nov 1996	0.365	**0.664**
Dec 1996	0.363	**0.650**
	DTM German	ML-DTM German
Aug 1996	0.315	**0.661**
Sep 1996	0.312	**0.670**
Oct 1996	0.310	**0.665**
Nov 1996	0.308	**0.638**
Dec 1996	0.306	**0.666**

Table 5: Topic diversity comparison between DTM and ML-DTM: average JS divergences of each topic pair for five months of the DE-NEWS dataset for English and German.

6 Results and Discussion

Tables 3 and 4 show the average cosine similarity between NEs for each language in the DE-NEWS and YLE datasets, respectively. In the DE-NEWS data (Table 3), PLTM outperforms ML-DTM in the first time slice but ML-DTM performs better on the succeeding time slices. This is an encouraging result, considering that the parameters of ML-DTM at time slice t are estimated from adjacent time slices, adding a large degree of complexity to the model, whereas PLTM estimates parameters based on the current time slice only (PLTM has no concept of time).

For the YLE dataset (Table 4), ML-DTM shows an improvement in the first time and third slices and comparable performance in the second. The comparable nature of this dataset makes aligning NEs a more challenging task for both models. One way to improve performance on this task might be to use stricter criteria in aligning the dataset, such as pairing articles only if they were published on the same day or if they share more named entities.

We compare topic diversity of the topics found by DTM and ML-DTM. Tables 5 and 6 show the average JS divergence of every topic pair for five time slices in the DE-NEWS and YLE datasets, respectively. ML-DTM consistently learns more diverse topics than DTM for both datasets.

In Figure 4, we show the evolution of one topic found by ML-DTM trained on DE-NEWS. We show the top words of a topic about labor unions for the first eight months of the dataset. The English and German words are not exact translations of each other but we see similar or related words and NEs in each time slice. For instance, in August 1996 'employer' and 'arbeitgeber' both appear, as does 'einzelhandel' and 'retail'. In Sept 1996, 'kohl' is the top term for both languages (referring to former German chancellor Helmut Kohl). There are cases where German terms have no direct translation in English but an equivalent concept appears in the English topic. This is the case with 'lohnfortzahlung' (sick-leave pay) where the terms 'sick' and 'pay' appear on the English side; and 'steuerreform' (tax reform) where 'reform' appears on the English side as well.

A named entity, 'thyssen', appears in March 1997 in both languages but not in other months. This is because of an event that happened around mid-March where the German steel company Thyssen was being bought by competitor Krupp-Hoesch (also a top term in the German topic) prompting concerns about job losses[5].

Figure 5 shows the first six months of a topic about political news from the YLE dataset. The first two months has terms related to presidential elections. This refers to the Finnish presidential election in 2012, where rounds of voting took place in January and February 2012[6]. These time slices also mention the two candidates in the runoff election, Sauli Niinistö and

[5] https://www.nytimes.com/1997/03/19/business/krupp-hoesch-confirms-bid-of-8-billion-for-thyssen.html

[6] https://en.wikipedia.org/wiki/2012_Finnish_presidential_election

Time slice	DTM Finnish	ML-DTM Finnish
Jan 2012	0.332	**0.445**
Feb 2012	0.324	**0.465**
Mar 2012	0.322	**0.470**
Apr 2012	0.353	**0.498**
May 2012	0.357	**0.495**
	DTM Swedish	ML-DTM Swedish
Jan 2012	0.365	**0.480**
Feb 2012	0.360	**0.491**
Mar 2012	0.354	**0.497**
Apr 2012	0.388	**0.535**
May 2012	0.393	**0.537**

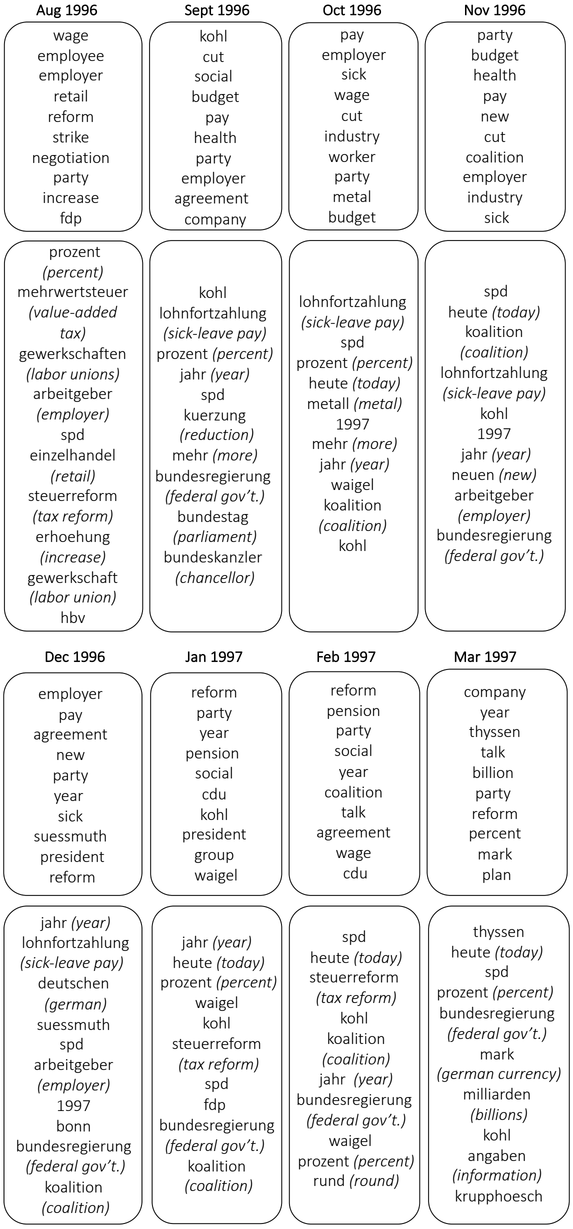
Table 6: Topic diversity comparison between DTM and ML-DTM: average JS divergences of each topic pair for five months of the YLE dataset for Finnish and Swedish.

Pekka Haavisto. Sauli Niinistö eventually won the election which explains why the next time slices ceases to mention Pekka Haavisto while 'niinistö' is still a prominent term. After March 2012, the topic stops talking about presidential elections and moves on to other political news. This gives us an insight into how the model can track significant events, such as high-profile elections, related to a topic. Another example is May 2012, where Greece ('kreikka' in Finnish, 'grekland' in Swedish) suddenly becomes a prominent term for both languages due to the Greek legislative elections which took place on 6 May 2012. The term 'syyria'/'syrien' appears in May and June, corresponding to the beginning of the Syrian Civil War.

Figure 6 shows the posterior probabilities of some terms related to the presidential elections ('niinistö'), Greece ('kreikka' or 'grekland') and Syria ('syyria' or 'syrien') in the political news topic for both languages. We see the rise and fall of the prominence of the terms according to their relevance in the news.

7 Conclusions and Future Work

In this paper we present a novel topic model, the *multilingual dynamic topic model* (ML-DTM), that combines dynamic topic modeling (DTM) and polylingual topic modeling (PLTM) to infer dynamic topics from aligned multilingual data. ML-DTM uses an extension of the DTM inference method of Bhadury et al. (2016) to aligned multilingual data.



Aug 1996	Sept 1996	Oct 1996	Nov 1996
wage employee employer retail reform strike negotiation party increase fdp	kohl cut social budget pay health party employer agreement company	pay employer sick wage cut industry worker party metal budget	party budget health pay new cut coalition employer industry sick
prozent *(percent)* mehrwertsteuer *(value-added tax)* gewerkschaften *(labor unions)* arbeitgeber *(employer)* spd einzelhandel *(retail)* steuerreform *(tax reform)* erhoehung *(increase)* gewerkschaft *(labor union)* hbv	kohl lohnfortzahlung *(sick-leave pay)* prozent *(percent)* jahr *(year)* spd kuerzung *(reduction)* mehr *(more)* bundesregierung *(federal gov't.)* bundestag *(parliament)* bundeskanzler *(chancellor)*	lohnfortzahlung *(sick-leave pay)* spd prozent *(percent)* heute *(today)* metall *(metal)* 1997 mehr *(more)* jahr *(year)* waigel koalition *(coalition)* kohl	spd heute *(today)* koalition *(coalition)* lohnfortzahlung *(sick-leave pay)* kohl 1997 jahr *(year)* neuen *(new)* arbeitgeber *(employer)* bundesregierung *(federal gov't.)*

Dec 1996	Jan 1997	Feb 1997	Mar 1997
employer pay agreement new party year sick suessmuth president reform	reform party year pension social cdu kohl president group waigel	reform pension party social year coalition talk agreement wage cdu	company year thyssen talk billion party reform percent mark plan
jahr *(year)* lohnfortzahlung *(sick-leave pay)* deutschen *(german)* suessmuth spd arbeitgeber *(employer)* 1997 bonn bundesregierung *(federal gov't.)* koalition *(coalition)*	jahr *(year)* heute *(today)* prozent *(percent)* waigel kohl steuerreform *(tax reform)* spd fdp bundesregierung *(federal gov't.)* koalition *(coalition)*	spd heute *(today)* steuerreform *(tax reform)* kohl koalition *(coalition)* jahr *(year)* bundesregierung *(federal gov't.)* waigel prozent *(percent)* rund *(round)*	thyssen heute *(today)* spd prozent *(percent)* bundesregierung *(federal gov't.)* mark *(german currency)* milliarden *(billions)* kohl angaben *(information)* krupphoesch

Figure 4: Top words of a topic concerning news about labor unions from the DE-NEWS dataset for English (top) and German (bottom) from Aug 1996 to March 1997. English translations of the German words excluding named entities are enclosed in parentheses.

We ran experiments on ML-DTM with parallel and comparable datasets. We compare cross-lingual topic alignment of PLTM and ML-DTM by evaluating the cosine similarities of topic vectors corresponding to named entity terms across languages for corresponding time slices. ML-DTM achieves similar performance to PLTM on DE-NEWS and the comparable dataset (YLE). We also demonstrate the ability of ML-DTM to detect

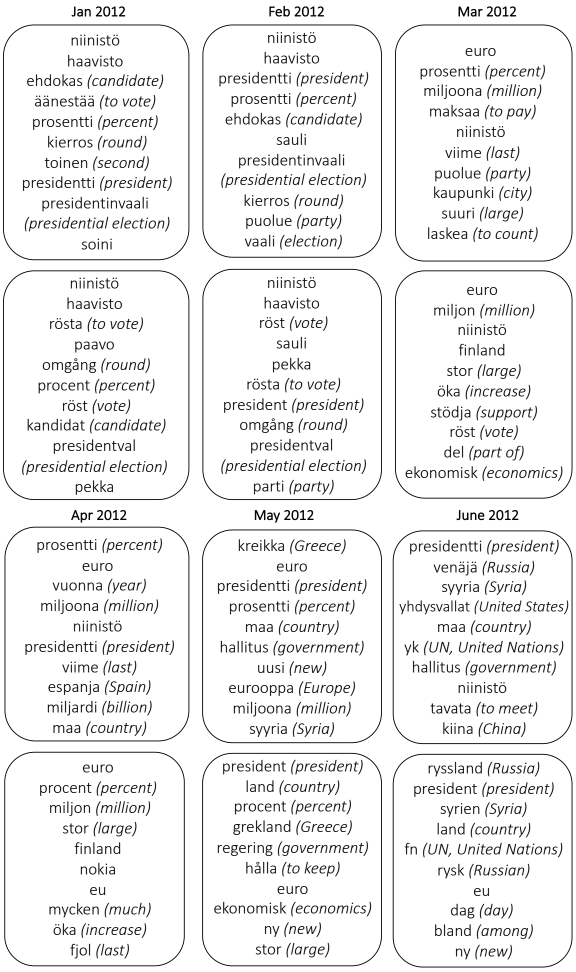

Jan 2012
Feb 2012
Mar 2012
niinistö
haavisto
ehdokas (candidate)
äänestää (to vote)
prosentti (percent)
kierros (round)
toinen (second)
presidentti (president)
presidentinvaali
(presidential election)
soini
niinistö
haavisto
presidentti (president)
prosentti (percent)
ehdokas (candidate)
sauli
presidentinvaali
(presidential election)
kierros (round)
puolue (party)
vaali (election)
euro
prosentti (percent)
miljoona (million)
maksaa (to pay)
niinistö
viime (last)
puolue (party)
kaupunki (city)
suuri (large)
laskea (to count)
niinistö
haavisto
rösta (to vote)
paavo
omgång (round)
procent (percent)
röst (vote)
kandidat (candidate)
presidentval
(presidential election)
pekka
niinistö
haavisto
röst (vote)
sauli
pekka
rösta (to vote)
president (president)
omgång (round)
presidentval
(presidential election)
parti (party)
euro
miljon (million)
niinistö
finland
stor (large)
öka (increase)
stödja (support)
röst (vote)
del (part of)
ekonomisk (economics)
Apr 2012
May 2012
June 2012
prosentti (percent)
euro
vuonna (year)
miljoona (million)
niinistö
presidentti (president)
viime (last)
espanja (Spain)
miljardi (billion)
maa (country)
kreikka (Greece)
euro
presidentti (president)
prosentti (percent)
maa (country)
hallitus (government)
uusi (new)
eurooppa (Europe)
miljoona (million)
syyria (Syria)
presidentti (president)
venäjä (Russia)
syyria (Syria)
yhdysvallat (United States)
maa (country)
yk (UN, United Nations)
hallitus (government)
niinistö
tavata (to meet)
kiina (China)
euro
procent (percent)
miljon (million)
stor (large)
finland
nokia
eu
mycken (much)
öka (increase)
fjol (last)
president (president)
land (country)
procent (percent)
grekland (Greece)
regering (government)
hålla (to keep)
euro
ekonomisk (economics)
ny (new)
stor (large)
ryssland (Russia)
president (president)
syrien (Syria)
land (country)
fn (UN, United Nations)
rysk (Russian)
eu
dag (day)
bland (among)
ny (new)

Figure 5: Top words of a topic on political news in Finland from the YLE dataset for Finnish (top) and Swedish (bottom) from Jan to June 2012. English translations of the words excluding named entities are enclosed in parentheses.

significant events regarding a topic through sudden changes in the prominent terms of the topic. This same method can also detect approximately when the event emerged and when it ended.

In a further experiment, we compared ML-DTM to the monolingual DTM, showing that ML-DTM achieves a consistently higher topic diversity within a single language.

We plan to run further experiments with ML-DTM using noisy datasets, such as historical news data where OCR errors might affect upstream tasks such as tokenization and lemmatization. We also plan to use named-entity recognition to improve our model such that named entities are treated as distinct items in the model's vocabulary, allowing us to track mentions of an entity across time slices and languages.

Historical news data covering a longer time span (several decades or more) would also enable us to study the changes in the use of words in a language and compare these changes with other languages. Historical news data from different regions would enable us to compare the way certain historical events were discussed in these places.

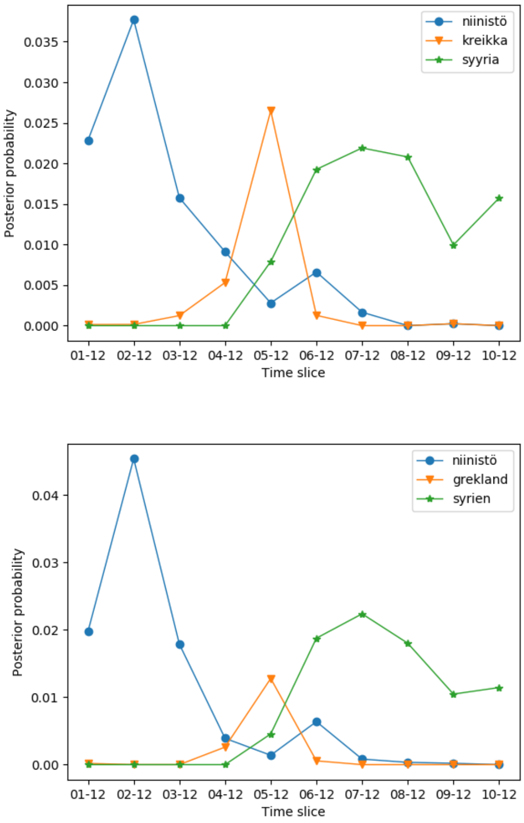


Figure 6: Posterior probabilities of salient terms in Finnish (top) and Swedish (bottom) related to events in the political news topic captured by ML-DTM from the YLE dataset.

Acknowledgements

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grants 770299 (NewsEye) and 825153 (EMBEDDIA).

References

Arnab Bhadury, Jianfei Chen, Jun Zhu, and Shixia Liu. 2016. Scaling up dynamic topic models. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 381–390.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd interna-*

tional conference on Machine learning. ACM, pages 113–120.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Jordan Boyd-Graber and David M Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pages 75–82.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics* 4:31–45.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences* 101(suppl 1):5228–5235.

David Hall, Daniel Jurafsky, and Christopher D Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 363–371.

Jagadeesh Jagarlamudi and Hal Daumé. 2010. Extracting multilingual topics from unaligned comparable corpora. In *European Conference on Information Retrieval*. Springer, pages 444–456.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Eetu Mäkelä. 2016. Las: an integrated language analysis tool for multiple languages. *The Journal of Open Source Software* 1.

David Mimno, Hanna M Wallach, Jason Naradowsky, David A Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, pages 880–889.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 72–79.

Jurgen Van Gael and Xiaojin Zhu. 2007. Correlation clustering for crosslingual link detection. In *IJCAI*. pages 1744–1749.

Thuy Vu, Ai Ti Aw, and Min Zhang. 2009. Feature-based method for document alignment in comparable news corpora. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 843–851.

Ivan Vulić, Wim De Smet, Jie Tang, and Marie-Francine Moens. 2015. Probabilistic topic modeling in multilingual settings: An overview of its methodology and applications. *Information Processing & Management* 51(1):111–147.

Chong Wang, David Blei, and David Heckerman. 2008. Continuous time dynamic topic models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, UAI'08, pages 579–586. http://dl.acm.org/citation.cfm?id=3023476.3023545.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 424–433.

Xing Wei, Jimeng Sun, and Xuerui Wang. 2007. Dynamic mixture models for multiple time-series. In *Ijcai*. volume 7, pages 2909–2914.

Max Welling and Yee W Teh. 2011. Bayesian learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 681–688.

Tze-I Yang, Andrew Torget, and Rada Mihalcea. 2011. Topic modeling on historical newspapers. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*. pages 96–104.

A Wide-Coverage Context-Free Grammar for Icelandic and an Accompanying Parsing System

Vilhjálmur Þorsteinsson, Hulda Óladóttir
Miðeind ehf.
Reykjavik
Iceland
vt@extrada.com, holado@gmail.com

Hrafn Loftsson
Department of Computer Science
Reykjavik University
Iceland
hrafn@ru.is

Abstract

We present an open-source, wide-coverage context-free grammar (CFG) for Icelandic, and an accompanying parsing system. The grammar has over 5,600 nonterminals, 4,600 terminals and 19,000 productions in fully expanded form, with feature agreement constraints for case, gender, number and person. The parsing system consists of an enhanced Earley-based parser and a mechanism to select best-scoring parse trees from shared packed parse forests. Our parsing system is able to parse about 90% of all sentences in articles published on the main Icelandic news websites. Preliminary evaluation with *evalb* shows an F-measure of 71.90% on parsed sentences. Our system demonstrates that parsing a morphologically rich language using a wide-coverage CFG can be practical.

1 Introduction

A CFG consists of a set of production rules that recursively describe how the strings of the underlying language can be derived. A parser for a natural language CFG checks whether a sentence can be derived by the CFG, and if so, assigns a syntactic structure (one or more parse trees) to the sentence. Well-known general parsing algorithms for CFGs include the (bottom-up) CYK algorithm (Younger, 1967) and the (top-down) Earley algorithm (Earley, 1970).

Various textbooks on Natural Language Processing (NLP) contain toy CFGs (for English, in most cases, e.g. (Jurafsky and Martin, 2009; Ljunglöf and Wirén, 2010)), but very few papers in recent literature describe hand-crafted natural language CFGs and parsers for them (we discuss a couple of them in Section 2). The reason is that the development of wide-coverage hand-crafted CFGs has been viewed as a challenging and time-consuming task (Briscoe et al., 1987; Hindle, 1989; Kinyon and Prolo, 2002), and that common phenomena, like agreement and inflection, have been considered complicated to describe using a CFG (Ljunglöf and Wirén, 2010). The processor and memory requirements of fully general parsing algorithms that can handle ambiguous grammars, being of worst-case cubic order (Younger, 1967; Scott, 2008), have also been seen as prohibitive. Beyond parsing, hand-crafted CFGs can be used *inter alia* to check and correct grammar in text (see Section 3).

Several treebanks have been developed for various languages during the past 25 years or so, e.g. (Marcus et al., 1993; Brants et al., 2004; Rögnvaldsson et al., 2012). Thus, instead of developing a CFG for a given language and a parser for the grammar, the more common approach has been to induce a probabilistic CFG (PCFG) from a treebank, and to train a probabilistic parser on the PCFG (Cahill, 2008). However, creating a high-quality treebank is a labor-intensive task, and, indeed, in many aspects similar to the process of manually crafting a grammar (Xia, 2001). Moreover, attempts to apply probabilistic parsing to morphologically rich languages (MRLs) have in many cases yielded unsatisfactory results, as complex word structure and flexible word order cause data sparseness in the probabilistic model (Tsarfaty et al., 2013).

In this paper, we present the development of an open-source, wide-coverage CFG for Icelandic, an MRL in the Germanic language family. We also present a parsing system, based on an enhanced Earley parser, that performs Part-of-Speech (PoS) tagging and parsing in a combined algorithm, and is able to cope with the large CFG and high levels of ambiguity associated with an MRL. It uses heuristics to return a single best scoring tree from the (often very large) packed parse forest for a sentence.

This paper is organized as follows: We discuss related work in Section 2 and the motivation for our work in Section 3. Tokenization and Out-of-Vocabulary words are discussed in Sections 4 and 5, respectively. We describe the development of the CFG in Section 6 and the parsing system in Section 7. The system is demonstrated in Section 8 and an evaluation is presented in Section 9. Finally, we conclude in Section 10.

2 Related Work

As discussed in Section 1, it is difficult to find published work from recent years describing the development of wide-coverage natural language CFGs and parsers for them. Here, we mention a couple of papers that we are aware of.

A CFG for English, developed over several years, is part of the GATE (General Architecture for Text Engineering) system, developed at the University of Sheffield (Gaizauskas et al., 2005). The CFG was specifically developed to complement a general purpose bottom-up chart parser called SUPPLE for feature-based CFGs, i.e. a CFG augmented with features in order to enforce agreement.

Abbas (2016) describes an extended Earley algorithm for parsing Urdu, an MRL. The parser uses a CFG extracted from a small treebank of 1,400 sentences (in comparison, the Penn Treebank (Marcus et al., 1993) contains 40,000 sentences). The small size of the treebank is, presumably, the main reason for not training a probabilistic parser. Moreover, the author mentions that it is hard to achieve good parsing results with probabilistic parsers for an MRL, without explicit encoding of linguistic information.

Before the work presented in this paper, no full parser existed for Icelandic. On the other hand, a shallow parser, *IceParser*, has been developed for the language (Loftsson and Rögnvaldsson, 2007). IceParser is an incremental finite-state parser, which annotates both constituent structure and syntactic functions, given PoS tagged input. The annotation scheme was designed in the project.

One Icelandic treebank, *The Icelandic Parsed Historical Corpus (IcePaHC)*, has been developed (Rögnvaldsson et al., 2012). It consists of one million words of historical texts, from the late 12^{th} century to the present, with about 10% being modern Icelandic. The annotation scheme follows the Penn Treebank style. IcePaHC has been used to train a probabilistic parser in a project aimed at improving the parsing accuracy of a related language, Faroese (Ingason et al., 2014).

3 Motivation

An initial primary goal of our project was to be able to extract information from text on Icelandic websites, such as associating person names with titles, and named entities with their definitions, irrespective of the grammatical forms that such associations may take in the text. As the project evolved, a secondary goal became to leverage the CFG and the parser to check and correct grammar, by adding specially annotated grammar rules and parser configurations for common error constructs. Finally, as our corpus of automatically parsed sentence trees grew to an order of millions, it became the basis of a follow-on project, beyond the scope of this paper, to train a deep neural network to parse Icelandic text.

4 Tokenization and Annotation

One of the catalysts for the project was the availability of the *Database of Modern Icelandic Inflection* (DMII) (Bjarnadóttir, 2012)[1]. DMII is a database that maps over 6 million lexical entries, 2.8 million unique word forms, to lemmas as well as PoS tags (word categories and morphological features).

The first phase of our project involved writing a tokenizer that uses data from the DMII to annotate each word token of a sentence with the set of its possible lemmas and corresponding PoS tag profile (possible PoS tags). In Icelandic, even common words such as *á* are highly ambiguous; *á* can mean *own* (verb), *female sheep* (noun), *river* (noun), or *on* (preposition), and can also occur as an adverb and an exclamation – our system associates it with 14 different PoS tags. This again means that the token for *á*, and indeed most word tokens, can match several different terminals in the CFG.

The tokenizer greedily recognizes certain multi-token spans, such as dates and adverbial multi-

[1] The DMII is available for download at http://bin.arnastofnun.is/dmii/.

word idioms, and coalesces them into single compound tokens. This makes the parsing stage more efficient and reduces ambiguity.

5 Out-of-Vocabulary Words

Compounding in Icelandic is very productive, and thus out-of-vocabulary (OOV) words, i.e. word forms that are not present in the DMII, are frequently encountered. We handle this by implementing a de-compounding algorithm, on top of a *Directed Acyclic Word Graph* comprising all word forms in the DMII, that maps each OOV word to a minimal sequence of prefixes followed by the longest possible suffix. This suffix is then used as a proxy for the compound word when looking up its PoS tags in the DMII.[2]

If the de-compounding algorithm is unable to make sense of a word, e.g. a foreign word, spelling error, or previously unseen named entity, its token is assumed to represent a neutral gender, singular noun that matches noun terminals in any of the four cases.

6 Context-Free Grammar

Once we had designed the mechanism for token-to-terminal matching (further described in Section 6.1), we proceeded to write out the nonterminals and productions of our CFG in an iterative fashion. Initially, a kernel of core nonterminals and productions for basic sentence structures and simple forms of noun, verb, prepositional and adverbial phrases was created.[3] This small CFG was then used to parse a starting reference set of typical texts selected from news articles. In each iteration, the most significant gaps in the CFG were identified and a round of improvements was applied. The set of reference texts was gradually expanded as the CFG coverage increased. This cycle is still ongoing, albeit of course with diminishing returns.

A web-based graphical user interface (GUI) was developed at an early stage in the project (see Section 8). The GUI provides an overview of news articles where sentences that the system fails to parse are clearly identified, and allows inspection of parse trees in graphical format. Having a visual way to identify problem areas and assess the system's performance facilitated the iterative cycle.

Our CFG for Icelandic is defined in a text file, presently about 5,800 lines including comments, written in a superset of Enhanced Backus-Naur Format (EBNF). This superset is our own implementation which facilitates automatic expansion of production rules (as discussed in Section 6.2). Apart from the CFG itself, the file contains annotation pragmas, such as priority scores for nonterminals (further discussed in Section 7.2). The total effort to date on the construction of the CFG is on the order of 2–3 man years.

6.1 Terminals

Our CFG allows three types of terminals:

- **Literal terminals**: Terminals of the form `"text"` match tokens with that text only (case-insensitive). This terminal form is, for example, used for various types of conjunctions.

- **Lemma terminals**: Terminals of the form `'lemma:category'_var1_var2...` match tokens having at least one PoS tag matching the indicated word category with the given lemma, optionally also agreeing with the specified variants (see Section 6.2). As an example, `'hafa:vb'_sg` (*hafa* being the infinitive of the verb *to have*) matches any singular form (`_sg`) of the verb, including *hef* (*[I] have*), *hafðir* (*[you] had*) and *hefur* (*[she] has*). This terminal form is, for example, used for auxiliary verbs in complex verb constructions.

- **Lookup terminals**: Terminals of the form `category_var1_var2...` match tokens having at least one PoS tag with the indicated word category that agrees with all of the specified variants, if any. As an example, `no_neut_sg_nom` matches any word token that has a PoS tag that identifies it as a noun (`no`), neutral (`_neut`), singular (`_sg`), nominative case (`_nom`). A word token such as *veður* (meaning *weather*) would match the terminal `no_neut_sg_nom`, but the same token also has a PoS tag indicating a verb (meaning *(he) wades*) and would thus also match the terminal `vb_sg_p3` that specifies a singular, third person verb.

[2] An example is *menntamálaráðherra*, (the minister of matters of education), composed of *mennta-mála-ráð-herra*, i.e. three prefixes and the suffix noun *herra*, from which the inflection of the compound word is derived.

[3] The general outlines of such a kernel would be similar for most languages in the Germanic family.

Feature	Variant	Values
Gender	`/gender`	`_masc, _fem, _neut`
Number	`/number`	`_sg, _pl`
Case	`/case`	`_nom, _acc, _dat, _gen`
Person	`/person`	`_p1, _p2, _p3`

Table 1: The major variants used in our CFG for Icelandic.

6.2 Variants and Feature Agreement

Describing nontrivial, potentially long-distance feature agreement in CFGs has been considered a challenge. In our grammar, we use automatic expansion of macro-like constructs, called *variants*, for this purpose. Variants are defined for the morphological features for which agreement is required within the productions of a nonterminal. They can be applied to both nonterminals and terminals. Table 1 shows the major variants used in our CFG for Icelandic.

Terminals with variants can only match word tokens that have at least one PoS tag that matches all of the variants. If a terminal has, e.g., an accusative case variant (`_acc`), it can only match word tokens that have a PoS tag indicating the accusative in the DMII.

As a simplified (and anglicized) example of how feature agreement is specified in the CFG, consider the following fragment:

```
NounPhrase ->
  Determiner/case/number/gender?
  Noun/case/number/gender
Determiner/case/number/gender ->
  det/case/number/gender
Noun/case/number/gender ->
  no/case/number/gender
```

Here, the nonterminal `NounPhrase` is defined as an optional `Determiner` having case, number and gender variants (`/case/number/gender`), followed by a mandatory `Noun` nonterminal, again with case, number and gender variants that agree in each expansion with the ones in the `Determiner`. The production for `Determiner` consists of a single terminal, `det/case/number/gender`, which matches word tokens having a PoS tag with the `det` category, inflected in accordance with the variants. The same applies to the production for `Noun`, which contains a single `no` terminal matching any noun that agrees with the features specified by the variants.

When the CFG is parsed, the variants are expanded as macros having each of their feature values in turn. The fragment above is expanded from three production rules to a total of 3 rules * 4 cases * 2 numbers * 3 genders = 72 expanded rules in the grammar. They include:

```
NounPhrase ->
    Determiner_nom_sg_masc? Noun_nom_sg_masc
  | Determiner_nom_sg_fem? Noun_nom_sg_fem
  | Determiner_nom_sg_neut? Noun_nom_sg_neut
...[21 generated productions omitted]...
Determiner_nom_sg_masc -> det_nom_sg_masc
Determiner_nom_sg_fem -> det_nom_sg_fem
Determiner_nom_sg_neut -> det_nom_sg_neut
...[21 generated productions omitted]...
Noun_nom_sg_masc -> no_nom_sg_masc
Noun_nom_sg_fem -> no_nom_sg_fem
Noun_nom_sg_neut -> no_nom_sg_neut
...[21 generated productions omitted]...
```

7 Parsing System

7.1 Earley-Based Parser

Parsing algorithms for natural language text can be evaluated on a number of criteria. They need to be able to parse all well-formed CFGs, including (heavily) ambiguous ones. A direct relationship between the produced parse forests and the original, unmodified CFG is a desirable feature. Last, but not least, the parsing performance in terms of time and space must be good enough for real-world applications.

The Earley (1970) algorithm handles all CFGs and, when extended from its original recognizer form to a full parser, returns parse trees that correspond directly to the original grammar (i.e. not requiring any *ex ante* transformation of the CFG). However, performance has been seen as a problem when parsing heavily ambiguous sentences, with both time and space requirements being worst-case cubic ($O(N^3)$) in the sentence length *N*.

Tomita (1986) described *Shared Packed Parse Forests* (SPPFs), a data structure that avoids redundancy when generating and storing parse trees for ambiguous token spans. Tomita's parser was a Generalized LR parser of worst-case unbounded polynomial order. Later, Scott (2008) and Scott and Johnstone (2010) presented an Earley-based parser which produces a binarized SPPF representation of all derivations of a sentence in worst-case cubic time. This parser meets all of the above mentioned criteria.

We implemented the Earley-Scott-Johnstone algorithm in C++ and found it to perform well enough for parsing natural language text according to our highly complex and ambiguous CFG for Icelandic.[4]

[4] Our system takes just over 1 second of wall-clock time

7.2 Disambiguation

The parser produces a binarized SPPF that includes all possible derivations of a sentence. A typical ambiguity factor is around 1.8, meaning that a sentence of N tokens typically has around 1.8^N different parse trees.

A nonterminal node in the SPPF may have more than one family of children if there are multiple derivations of that nonterminal for the same token span. Note that a single packed subtree may occur within multiple parse trees in a forest, by virtue of the packing mechanism.

Once the parse forest has been derived, bottom-up scoring heuristics are applied in a disambiguation step to find a single "best" parse tree. We begin the disambiguation process at the terminal (bottom) level of the SPPF, by assigning a score to each token-terminal match based on the probability of that match. For instance, the word *ekki* can be both an adverb (meaning *not*) and a noun (meaning *sob*, as in crying). The former meaning is much more common than the latter one, and thus a match of *ekki* to a noun terminal gets a relatively lower score, while a match of *ekki* to an adverb terminal gets a relatively higher score. The scores are heuristically defined and hand-tuned by us.

The scores assigned at the terminal level are summed up as they percolate upwards in the SPPF, and further adjusted at nonterminal nodes. In the CFG specification, scores can be assigned to nonterminals using pragmas. More common grammatical constructs have positive scores, while less common or exceptional ones have negative scores.

When the traversal encounters an ambiguous nonterminal node with multiple families of children, the accumulated scores of the families are compared. The family (subtree) with the highest score "wins" and is retained, while the other families are pruned from the tree. At the root (top) nonterminal, a single highest-scoring tree remains and is returned. It is at this point that the PoS tag for each word token is finally determined.

7.3 Attachment of Prepositions

There is, however, one exception to the general case described in the last subsection: The mechanism for attaching prepositional phrases (PPs) to verb or noun phrases. This well-known problem manifests itself in ambiguous sentences such as: *Ég las blaðið í gær* (*I read the paper yesterday / I read yesterday's paper*); should the PP *í gær* (*yesterday*) be attached to the verb *las* (*read*) or to the noun *blaðið* (*the paper*)? The problem is not perfectly solvable since the correct attachment may depend on semantics. We approximate a solution by augmenting our database of almost 7,300 verbs with a hand-crafted list of prepositions that are commonly attached to each verb. For example, with the verb *tala*, meaning *talk*, we associate the preposition *við/acc* (meaning *to/accusative*), indicating that the preposition *við*, when governing an object in the accusative case, is commonly attached to the verb *tala*.

Recall that our parser generates an SPPF representing all possible parse trees, including ones where PPs are attached to verbs and ones where they are attached to nouns. We want to selectively boost the score of subtrees where a PP (such as *við/acc*) is attached to a verb phrase (VP) with a matching verb (such as *tala*). This makes such subtrees more likely to "win" in the disambiguation and tree pruning process, vs. subtrees where the same PP is attached to a noun or to a non-matching verb.

In order for this to work correctly, we must partially unpack the parse forest. Specifically, PP nodes are unpacked (cloned) to become independent children of any ambiguous parent nonterminal nodes up to and including the enclosing VP node. This is done in a special top-down preprocessing pass before the main disambiguation scoring and pruning pass. During the scoring pass, we keep track of current VP scopes (such as the scope of *tala*) and assign scores to preposition terminal nodes (such as *við*) depending on whether the preposition matches an enclosing verb. Since the PP nodes have been unpacked, they can have different scores in different subtrees and can be independently pruned from the forest, which the SPPF structure would otherwise not allow.

8 Demonstration

Our system is open and available both as a public website[5] and as a GNU GPLv3-licensed Python package for NLP researchers and developers.[6]

to parse, disambiguate and process a typical 22-sentence news article from the web, on a quad-core Intel® i7 based GNU/Linux server.

[5] `https://greynir.is` (only in Icelandic).

[6] The source code for the CFG and the parsing system, written in Python 3 and C++, is available at `https://github.com/mideind/ReynirPackage`.

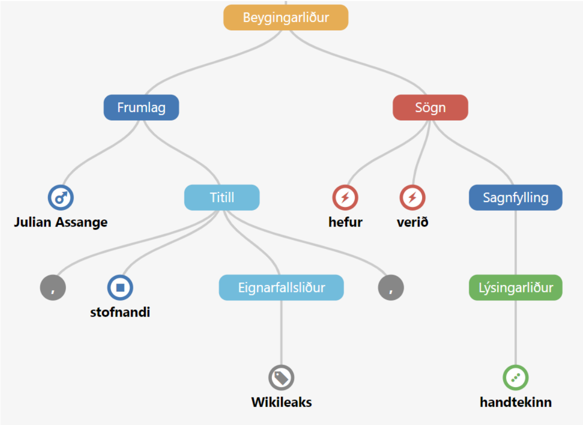


Figure 1: A part of a simplified parse tree, as displayed on the `greynir.is` website. The sentence is *Julian Assange, founder of Wikileaks, has been arrested*. (*Beygingarliður*, *Frumlag*, *Eignarfallsliður*, *Sögn*, *Sagnfylling*, *Lýsingarliður*) = (*IP*, *Subject*, *Possessive Phrase*, *VP*, *Predicative Complement*, *Adjective Phrase*).

The website is refreshed every 30 minutes with new articles from the major Icelandic news outlets. The news articles can be browsed, and parse trees for individual sentences can be viewed in a graphical format by clicking on them (see Figure 1). Sentences that the system is not able to parse are marked in red. The system shows the names and titles of people who have been mentioned in recent news articles, the definitions of named entities, and geographical locations that occur in recent news. This information is picked up from tokens and parse trees.

Apart from browsing news articles, users are able to enter their own text and have it parsed and checked by the system. As with news articles, the parse tree of a user-entered sentence can be displayed by clicking on the sentence.

9 Evaluation

On the basis of the current CFG, our parser is able to derive at least one parse tree from about 90% of sentences in our database of 8.7 million sentences extracted from news articles. The remaining 10% are divided between sentences in languages other than Icelandic, number-oriented text in currently unparsable format such as sports results and data tables, sentences containing severe spelling or grammar errors, and valid sentences which our CFG does not yet cover.

Recall	**Precision**	**F-measure**	**Average Crossing**
`71.15%`	`72.67%`	`71.90%`	`4.18`

Table 2: *evalb* results for 78 hand-annotated sentences.

To estimate the accuracy of our CFG and the parser, we hand-annotated 80 randomly selected sentences from news articles using a simplified, syntactically bracketed annotation scheme similar to that of the Penn Treebank. The trees output from our parser were converted from their internal representation, corresponding directly to the CFG, to the simplified scheme via a set of mapping rules. Spelling and grammar errors were corrected before parsing. Results for this small test corpus, using the *evalb* tool (Sekine and Collins, 2013), are shown in Table 2.

Out of 80 sentences, two could not be parsed as they contain syntactic structures not presently covered by the CFG. Out of 1,444 word tokens, 66 (4.6%) were OOV of which the de-compounding algorithm handles 35.

Many errors are caused by wrong attachment of PPs and subclauses, or when NPs from phrases or clauses deeper in the tree are erroneously attached as objects of verbs in the main clause, instead of correctly identifying a complement clause or PP as the object.

Both error types affect all intermediate levels in the tree, and thus lower *evalb*'s reported scores severely. We continue to work on our grammar and our parsing system to address these errors, as well as developing a gold standard of parsed news text for more accurate evaluation.

10 Conclusion

We have presented a system consisting of an open-source, wide-coverage CFG for Icelandic, and an accompanying parser. Our system demonstrates that it is practical to develop a wide-coverage CFG for an MRL, such as Icelandic, with a parsing system that performs well enough for real-world use cases. Such a system can be used *inter alia* for information extraction from news websites, grammar correction, and to generate dependency annotated corpora as well as treebanks for training deep neural network-based parsers.

References

Qaiser Abbas. 2016. Morphologically rich Urdu grammar parsing using Earley algorithm. *Natural Language Engineering* 22(5):775–810. https://doi.org/10.1017/S1351324915000133.

Kristín Bjarnadóttir. 2012. The Database of Modern Icelandic Inflection. In *Proceedings of the workshop "Language Technology for Normalization of Less-Resourced Languages", SaLTMiL 8 – AfLaT*. Istanbul, Turkey, LREC 2012. http://www.lexis.hi.is/kristinb/lrec2012-dmii.pdf.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation* (2):597–620. https://doi.org/10.1007/s11168-004-7431-3.

Ted Briscoe, Claire Grover, Bran Boguraav, and John Carroll. 1987. A Formalism and Environment for the Development of a Large Grammar of English. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 2*. Milan, Italy, IJCAI'87. http://dl.acm.org/citation.cfm?id=1625995.1626017.

Aoife Cahill. 2008. Treebank-Based Probabilistic Phrase Structure Parsing. *Language and Linguistics Compass* 2(1):18–40. https://doi.org/https://doi.org/10.1111/j.1749-818X.2007.00046.x.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13(2):94–102. https://doi.org/10.1145/362007.362035.

Robert Gaizauskas, Mark Hepple, Horacio Saggion, Mark A. Greenwood, and Kevin Humphreys. 2005. SUPPLE: A Practical Parser for Natural Language Engineering Applications. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, Canada. http://aclweb.org/anthology/W05-1527.

Donald Hindle. 1989. Acquiring Disambiguation Rules from Text. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*. Vancouver, Canada, ACL '89. https://doi.org/10.3115/981623.981638.

Anton Karl Ingason, Hrafn Loftsson, Eiríkur Rögnvaldsson, Einar Freyr Sigurðsson, and Joel C. Wallenberg. 2014. Rapid Deployment of Phrase Structure Parsing for Related Languages: A Case Study of Insular Scandinavian. In *Proceedings of the 9^{th} International Conference on Language Resources and Evaluation*, Reykjavik, Iceland, LREC 2014. http://www.lrec-conf.org/proceedings/lrec2014/pdf/855_Paper.pdf.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, 2nd edition.

Alexandra Kinyon and Carlos A. Prolo. 2002. A Classification of Grammar Development Strategies. In *Proceedings of the 2002 Workshop on Grammar Engineering and Evaluation - Volume 15*. COLING-GEE '02. https://doi.org/10.3115/1118783.1118790.

Peter Ljunglöf and Mats Wirén. 2010. Syntactic Parsing. In N. Indurkhya and F. J. Damerau, editor, *Handbook of Natural Language Processing*, CRC Press. 2nd edition.

Hrafn Loftsson and Eiríkur Rögnvaldsson. 2007. IceParser: An Incremental Finite-State Parser for Icelandic. In *Proceedings of the 16^{th} Nordic Conference of Computational Linguistics*. Tartu, Estonia, NODALIDA 2007. https://aclweb.org/anthology/papers/W/W07/W07-2419/.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurðsson, and Joel Wallenberg. 2012. The Icelandic Parsed Historical Corpus (IcePaHC). In *Proceedings of the 8^{th} International Conference on Language Resources and Evaluation*, Istanbul, Turkey, LREC 2012. http://www.lrec-conf.org/proceedings/lrec2012/pdf/440_Paper.pdf.

Elizabeth Scott. 2008. SPPF-Style Parsing From Earley Recognisers. *Electronic Notes in Theoretical Computer Science* 203:53 – 67. https://doi.org/10.1016/j.entcs.2008.03.044.

Elizabeth Scott and Adrian Johnstone. 2010. Recognition is Not Parsing - SPPF-style Parsing from Cubic Recognisers. *Science of Computer Programming* 75(1-2):55–70. https://doi.org/10.1016/j.scico.2009.07.001.

Satoshi. Sekine and Michael J. Collins. 2013. The evalb software. `http://cs.nyu.edu/cs/projects/proteus/evalb`.

Masaru Tomita. 1986. *Efficient parsing for natural language*. Kluwer Academic Publishers, Boston.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing Morphologically Rich Languages: Introduction to the Special Issue. *Computational Linguistics* 39(1):15–22. https://doi.org/10.1162/COLI_a_00133.

Fei Xia. 2001. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania. https://dl.acm.org/citation.cfm?id=934569.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control* 10(2):189–208. https://doi.org/10.1016/S0019-9958(67)80007-X.

Author Index