ACL HLT 2011

49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

Proceedings of System Demonstrations

21 June 2011
Portland, Oregon, USA

Production and Manufacturing by
Omnipress, Inc.
2600 Anderson Street
Madison, WI 53704 USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-932432-90-9

Introduction

Welcome to the proceedings of the system demonstration session. This volume contains the papers of the system demonstrations presented at the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, held in Portland, Oregon, USA, on June 21, 2011.

The system demonstrations program offers the presentation of early research prototypes as well as interesting mature systems. The system demonstration chair and the members of the program committee received 46 submissions, 24 of which were selected for inclusion in the program after review by two members of the program committee.

I would like to thank the members of the program committee for their excellent job in reviewing the submissions and providing their support for the final decision.

Chair

Sadao Kurohashi (Kyoto Univeristy, Japan)

Program Committee:

Srinivas Bangalore (AT&T Labs Research, USA)
Tilman Becker (DFKI, Germany)
Pushpak Bhattacharyya (IIT, India)
Francis Bond (Nanyang Technological University, Singapore)
Josef van Genabith (Dublin City University, Ireland)
Kristiina Jokinen (University of Helsinki, Finland)
Tatsuya Kawahara (Kyoto University, Japan)
Adam Kilgarriff (Lexical Computing Ltd., UK)
Gary Geunbae Lee (POSTECH, Korea)
Qun Liu (Chinese Academy of Sciences, China)
Roberto Navigli (University of Roma, Italy)
Hoifung Poon (University of Washington, USA)
Kenji Sagae (University of Southern California, USA)
Satoshi Sekine (New York University, USA)
Takenobu Tokunaga (Titech, Japan)
Kentaro Torisawa (NICT, Japan)
Takehito Utsuro (University of Tsukuba, Japan)
Yeyi Wang (Microsoft, USA)
Jason Williams (AT&T Labs Research, USA)
Dekai Wu (HKUST, Hong Kong)
Xiaoyan Zhu (Tsinghua University, China)
Udo Hahn (Jena University, Germany)

Additional Reviewer:

Daisuke Kawahara (Kyoto University, Japan)

Table of Contents

Hindi to Punjabi Machine Translation System
Vishal Goyal and Gurpreet Singh Lehal 1

The ACL Anthology Searchbench
Ulrich Schäfer, Bernd Kiefer, Christian Spurk, Jörg Steffen and Rui Wang 7

Exploiting Readymades in Linguistic Creativity: A System Demonstration of the Jigsaw Bard
Tony Veale and Yanfen Hao 14

A Mobile Touchable Application for Online Topic Graph Extraction and Exploration of Web Content
Günter Neumann and Sven Schmeier 20

EdIt: A Broad-Coverage Grammar Checker Using Pattern Grammar
Chung-chi Huang, Mei-hua Chen, Shih-ting Huang and Jason S. Chang 26

MemeTube: A Sentiment-based Audiovisual System for Analyzing and Displaying Microblog Messages
Cheng-Te Li, Chien-Yuan Wang, Chien-Lin Tseng and Shou-De Lin 32

An ERP-based Brain-Computer Interface for text entry using Rapid Serial Visual Presentation and Language Modeling
Kenneth Hild, Umut Orhan, Deniz Erdogmus, Brian Roark, Barry Oken, Shalini Purwar, Hooman Nezamfar and Melanie Fried-Oken 38

Engkoo: Mining the Web for Language Learning
Matthew R. Scott, Xiaohua Liu, Ming Zhou and Microsoft Engkoo Team 44

Dr Sentiment Knows Everything!
Amitava Das and Sivaji Bandyopadhyay 50

Blast: A Tool for Error Analysis of Machine Translation Output
Sara Stymne 56

Prototyping virtual instructors from human-human corpora
Luciana Benotti and Alexandre Denis 62

An Interactive Machine Translation System with Online Learning
Daniel Ortiz-Martínez, Luis A. Leiva, Vicent Alabau, Ismael García-Varea and Francisco Casacuberta 68

Wikulu: An Extensible Architecture for Integrating Natural Language Processing Techniques with Wikis
Daniel Bär, Nicolai Erbs, Torsten Zesch and Iryna Gurevych 74

A Speech-based Just-in-Time Retrieval System using Semantic Search
Andrei Popescu-Belis, Majid Yazdani, Alexandre Nanchen and Philip N. Garner 80

MACAON An NLP Tool Suite for Processing Word Lattices
Alexis Nasr, Frederic Bechet, Jean-Francois Rey, Benoit Favre and Joseph Le Roux 86

Multimodal Menu-based Dialogue with Speech Cursor in DICO II+
Staffan Larsson, Alexander Berman and Jessica Villing 92

Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History
Oliver Ferschke, Torsten Zesch and Iryna Gurevych 97

An Efficient Indexer for Large N-Gram Corpora
Hakan Ceylan and Rada Mihalcea 103

SystemT: A Declarative Information Extraction System
Yunyao Li, Frederick Reiss and Laura Chiticariu 109

SciSumm: A Multi-Document Summarization System for Scientific Articles
Nitin Agarwal, Ravi Shankar Reddy, Kiran GVR and Carolyn Penstein Rosé 115

Clairlib: A Toolkit for Natural Language Processing, Information Retrieval, and Network Analysis
Amjad Abu-Jbara and Dragomir Radev 121

C-Feel-It: A Sentiment Analyzer for Micro-blogs
Aditya Joshi, Balamurali AR, Pushpak Bhattacharyya and Rajat Mohanty 127

IMASS: An Intelligent Microblog Analysis and Summarization System
Jui-Yu Weng, Cheng-Lun Yang, Bo-Nian Chen, Yen-Kai Wang and Shou-De Lin 133

An Interface for Rapid Natural Language Processing Development in UIMA
Balaji Soundrarajan, Thomas Ginter and Scott DuVall 139

Conference Program

Tuesday, June 21, 2011

10:30–1:00 *Hindi to Punjabi Machine Translation System*
Vishal Goyal and Gurpreet Singh Lehal

10:30–1:00 *The ACL Anthology Searchbench*
Ulrich Schäfer, Bernd Kiefer, Christian Spurk, Jörg Steffen and Rui Wang

10:30–1:00 *Exploiting Readymades in Linguistic Creativity: A System Demonstration of the Jigsaw Bard*
Tony Veale and Yanfen Hao

10:30–1:00 *A Mobile Touchable Application for Online Topic Graph Extraction and Exploration of Web Content*
Günter Neumann and Sven Schmeier

10:30–1:00 *EdIt: A Broad-Coverage Grammar Checker Using Pattern Grammar*
Chung-chi Huang, Mei-hua Chen, Shih-ting Huang and Jason S. Chang

10:30–1:00 *MemeTube: A Sentiment-based Audiovisual System for Analyzing and Displaying Microblog Messages*
Cheng-Te Li, Chien-Yuan Wang, Chien-Lin Tseng and Shou-De Lin

10:30–1:00 *An ERP-based Brain-Computer Interface for text entry using Rapid Serial Visual Presentation and Language Modeling*
Kenneth Hild, Umut Orhan, Deniz Erdogmus, Brian Roark, Barry Oken, Shalini Purwar, Hooman Nezamfar and Melanie Fried-Oken

10:30–1:00 *Engkoo: Mining the Web for Language Learning*
Matthew R. Scott, Xiaohua Liu, Ming Zhou and Microsoft Engkoo Team

10:30–1:00 *Dr Sentiment Knows Everything!*
Amitava Das and Sivaji Bandyopadhyay

10:30–1:00 *Blast: A Tool for Error Analysis of Machine Translation Output*
Sara Stymne

10:30–1:00 *Prototyping virtual instructors from human-human corpora*
Luciana Benotti and Alexandre Denis

10:30–1:00 *An Interactive Machine Translation System with Online Learning*
Daniel Ortiz-Martínez, Luis A. Leiva, Vicent Alabau, Ismael García-Varea and Francisco Casacuberta

Tuesday, June 21, 2011 (continued)

1:30–4:00 *Wikulu: An Extensible Architecture for Integrating Natural Language Processing Techniques with Wikis*
Daniel Bär, Nicolai Erbs, Torsten Zesch and Iryna Gurevych

1:30–4:00 *A Speech-based Just-in-Time Retrieval System using Semantic Search*
Andrei Popescu-Belis, Majid Yazdani, Alexandre Nanchen and Philip N. Garner

1:30–4:00 *MACAON An NLP Tool Suite for Processing Word Lattices*
Alexis Nasr, Frederic Bechet, Jean-Francois Rey, Benoit Favre and Joseph Le Roux

1:30–4:00 *Multimodal Menu-based Dialogue with Speech Cursor in DICO II+*
Staffan Larsson, Alexander Berman and Jessica Villing

1:30–4:00 *Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History*
Oliver Ferschke, Torsten Zesch and Iryna Gurevych

1:30–4:00 *An Efficient Indexer for Large N-Gram Corpora*
Hakan Ceylan and Rada Mihalcea

1:30–4:00 *SystemT: A Declarative Information Extraction System*
Yunyao Li, Frederick Reiss and Laura Chiticariu

1:30–4:00 *SciSumm: A Multi-Document Summarization System for Scientific Articles*
Nitin Agarwal, Ravi Shankar Reddy, Kiran GVR and Carolyn Penstein Rosé

1:30–4:00 *Clairlib: A Toolkit for Natural Language Processing, Information Retrieval, and Network Analysis*
Amjad Abu-Jbara and Dragomir Radev

1:30–4:00 *C-Feel-It: A Sentiment Analyzer for Micro-blogs*
Aditya Joshi, Balamurali AR, Pushpak Bhattacharyya and Rajat Mohanty

1:30–4:00 *IMASS: An Intelligent Microblog Analysis and Summarization System*
Jui-Yu Weng, Cheng-Lun Yang, Bo-Nian Chen, Yen-Kai Wang and Shou-De Lin

1:30–4:00 *An Interface for Rapid Natural Language Processing Development in UIMA*
Balaji Soundrarajan, Thomas Ginter and Scott DuVall

HINDI TO PUNJABI MACHINE TRANSLATION SYSTEM

Vishal Goyal
Department of Computer Science
Punjabi University, Patiala,India
vishal.pup@gmail.com

Gurpreet Singh Lehal
Department of Computer Science
Punjabi University, Patiala,India
gslehal@gmail.com

Abstract

Hindi-Punjabi being closely related language pair (Goyal V. and Lehal G.S., 2008) , Hybrid Machine Translation approach has been used for developing Hindi to Punjabi Machine Translation System. Non-availability of lexical resources, spelling variations in the source language text, source text ambiguous words, named entity recognition and collocations are the major challenges faced while developing this syetm. The key activities involved during translation process are preprocessing, translation engine and post processing. Lookup algorithms, pattern matching algorithms etc formed the basis for solving these issues. The system accuracy has been evaluated using intelligibility test, accuracy test and BLEU score. The hybrid syatem is found to perform better than the constituent systems.

Keywords: *Machine Translation, Computational Linguistics, Natural Language Processing, Hindi, Punjabi. Translate Hindi to Punjabi, Closely related languages.*

1 Introduction

Machine Translation system is a software designed that essentially takes a text in one language (called the source language), and translates it into another language (called the target language). There are number of approaches for MT like Direct based, Transform based, Interlingua based, Statistical etc. But the choice of approach depends upon the available resources and the kind of languages involved. In general, if the two languages are structurally similar, in particular as regards lexical correspondences, morphology and word order, the case for abstract syntactic analysis seems less convincing. Since the present research work deals with a pair of closely related language i.e. Hindi-Punjabi , thus direct word-to-word translation approach is the obvious choice. As some rule based approach has also been used, thus, Hybrid approach has been adopted for developing the system. An exhaustive survey has already been given for existing machine translations systems developed so far mentioning their accuracies and limitations. (Goyal V. and Lehal G.S., 2009).

2 System Architecture

2.1 Pre Processing Phase

The preprocessing stage is a collection of operations that are applied on input data to make it processable by the translation engine. In the first phase of Machine Translation system, various activities incorporated include text normalization, replacing collocations and replacing proper nouns.

2.2 Text Normalization

The variety in the alphabet, different dialects and influence of foreign languages has resulted in spelling variations of the same word. Such variations sometimes can be treated as errors in writing. (Goyal V. and Lehal G.S., 2010).

2.3 Replacing Collocations

After passing the input text through text normalization, the text passes through this Collocation replacement sub phase of Pre-processing phase. Collocation is two or more consecutive words with a special behavior. (Choueka :1988). For example, the collocation उत्तर प्रदेश (*uttar pradēsh)* if translated word to word, will be translated as ਜਵਾਬ ਰਾਜ (*javāb rāj)* but it must be translated as ਉੱਤਰ ਪ੍ਰਦੇਸ਼ (*uttar pradēsh)*. The accuracy of the results for collocation extraction using t-test is not accurate and includes number of such bigrams and trigrams that are not actually collocations. Thus, manually such entries were removed and actual collocations were further extracted. The

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 1–6,
Portland, Oregon, USA, 21 June 2011.

correct corresponding Punjabi translation for each extracted collocation is stored in the collocation table of the database. The collocation table of the database consists of 5000 such entries. In this sub phase, the normalized input text is analyzed. Each collocation in the database found in the input text will be replaced with the Punjabi translation of the corresponding collocation. It is found that when tested on a corpus containing about 1,00,000 words, only 0.001% collocations were found and replaced during the translation.

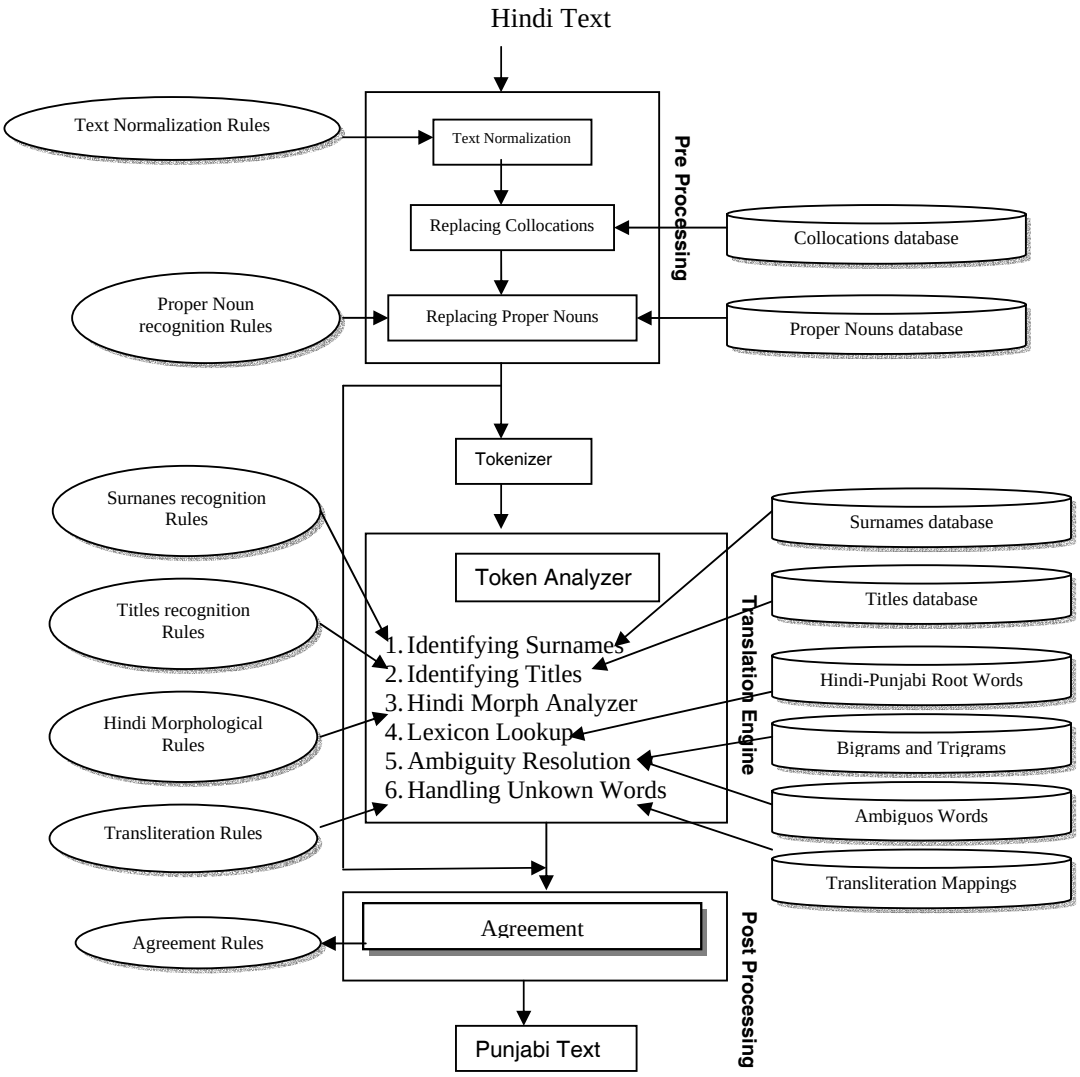


Figure 1 : Overview of Hindi-Punjabi Machine Translation System

2.4 Replacing Proper Nouns

A great proposition of unseen words includes proper nouns like personal, days of month, days of week, country names, city names, bank names, organization names, ocean names, river names, university names etc. and if translated word to word, their meaning is changed. If the meaning is not affected, even though this step fastens the translation process. Once these words are recognized and stored into the proper noun database, there is no need to decide about their translation or transliteration every time in the case of presence of such words in input text for translation. This gazetteer makes the translation accurate and fast. This list is self growing during each

translation. Thus, to process this sub phase, the system requires a proper noun gazetteer that has been complied offline. For this task, we have developed an offline module to extract proper nouns from the corpus based on some rules. Also, Named Entity recognition module has been developed based on the CRF approach (Sharma R. and Goyal V., 2011b).

2.5 Tokenizer

Tokenizers (also known as lexical analyzers or word segmenters) segment a stream of characters into meaningful units called tokens. The tokenizer takes the text generated by pre processing phase as input. Individual words or tokens are extracted and processed to generate its equivalent in the target language. This module, using space, a punctuation mark, as delimiter, extracts tokens (word) one by one from the text and gives it to translation engine for analysis till the complete input text is read and processed.

2.6 Translation Engine

The translation engine is the main component of our Machine Translation system. It takes token generated by the tokenizer as input and outputs the translated token in the target language. These translated tokens are concatenated one after another along with the delimiter. Modules included in this phase are explained below one by one.

2.6.1 Identifying Titles and Surnames

Title may be defined as a formal appellation attached to the name of a person or family by virtue of office, rank, hereditary privilege, noble birth, or attainment or used as a mark of respect. Thus word next to title and word previous to surname is usually a proper noun. And sometimes, a word used as proper name of a person has its own meaning in target language. Similarly, Surname may be defined as a name shared in common to identify the members of a family, as distinguished from each member's given name. It is also called family name or last name. When either title or surname is passed through the translation engine, it is translated by the system. This cause the system failure as these proper names should be transliterated instead of translation. For example consider the Hindi sentence श्रीमान हर्ष जी हमारे यहाँ पधारे। (*shrīmān harsh jī hamārē yahāṃ padhārē)*. In this sentence, हर्ष (*harsh)* has the meaning "joy". The equivalent translation of हर्ष (*harsh)* in target language is ਖੁਸ਼ੀ (*khushī)*. Similarly, consider the Hindi sentence प्रकाश सिंह हमारे यहाँ पधारे।. *(prakāsh siṃh hamārē yahāṃ padhārē)*. Here, प्रकाश (*prakāsh)* word is acting as proper noun and it must be transliterated and not translated because सिंह (*siṃh)* is surname and word previous to it is proper noun.

Thus, a small module has been developed for locating such proper nouns to consider them as title or surname. There is one special character '◦' in Devanagari script to mark the symbols like डा◦, प्रो◦. If this module found this symbol to be title or surname, the word next and previous to this token as the case may be for title or surname respectively, will be transliterated not translated. The title and surname database consists of 14 and 654 entries respectively. These databases can be extended at any time to allow new titles and surnames to be added. This module was tested on a large Hindi corpus and showed that about 2-5 % text of the input text depending upon its domain is proper noun. Thus, this module plays an important role in translation.

2.6.2 Hindi Morphological analyzer

This module finds the root word for the token and its morphological features.Morphological analyzer developed by IIT-H has been ported for Windows platform for making it usable for this system. (Goyal V. and Lehal G.S.,2008a)

2.6.3 Word-to-Word translation using lexicon lookup

If token is not a title or a surname, it is looked up in the HPDictionary database containing Hindi to Punjabi direct word to word translation. If it is found, it is used for translation. If no entry is found in HPDictionary database, it is sent to next sub phase for processing. The HPDictionary database consists of 54,127 entries.This database can be extended at any time to allow new entries in the dictionary to be added.

2.6.4 Resolving Ambiguity

Among number of approaches for disambiguation, the most appropriate approach to determine the correct meaning of a Hindi word in a particular usage for our Machine Translation system is to examine its context using N-gram approach. After analyzing the past experiences of various authors, we have chosen the value of n to be 3 and 2 i.e. trigram and bigram approaches respectively for our system. Trigrams are further categorized into three different types. First category of trigram consists of context one word previous to and one word next to the ambiguous word. Second category of trigram consists of context of two adjacent previous words to the ambiguous word. Third category of the trigram consists of context of two adjacent next words to the ambiguous word. Bigrams are also categorized into two categories. First category of the bigrams consists of context of one previous word to ambiguous word and second category of the bigrams consists of one context word next to ambiguous word. For this purpose, the Hindi corpus consisting of about 2 million words was collected from different sources like online newspaper daily news, blogs, Prem Chand stories, Yashwant jain stories, articles etc. The most common list of ambiguous words was found. We have found a list of 75 ambiguous words out of which the most frequent are से *sē* and और *aur*. (Goyal V. and Lehal G.S., 2011)

2.6.5 Handling Unknown Words

2.6.5.1 Word Inflectional Analysis and generation

In linguistics, a suffix (also sometimes called a *postfix* or *ending*) is an affix which is placed after the stem of a word. Common examples are case endings, which indicate the grammatical case of nouns or adjectives, and verb endings. Hindi is a (relatively) free word-order and highly inflectional language. Because of same origin, both languages have very similar structure and grammar. The difference is only in words and in pronunciation e.g. in Hindi it is लड़का and in Punjabi the word for boy is ਮੁੰਡਾ and even sometimes that is also not there like घर (*ghar)* and ਘਰ (*ghar)*. The inflection forms of both these words in Hindi and Punjabi are also similar. In this activity, inflectional analysis without using morphology has been performed for all those tokens that are not processed by morphological analysis module. Thus, for performing inflectional analysis, rule based approach has been followed. When the token is passed to this sub phase for inflectional analysis, If any pattern of the regular expression (inflection rule) matches with this token, that rule is applied on the token and its equivalent translation in Punjabi is generated based on the matched rule(s). There is also a check on the generated word for its correctness. We are using correct Punjabi words database for testing the correctness of the generated word.

2.6.5.2 Transliteration

This module is beneficial for handling out-of-vocabulary words. For example the word विशाल (*vishāl)* is transliterated as ਵਿਸ਼ਾਲ (*vishāl)* whereas translated as ਵੱਡਾ. There must be some method in every Machine Translation system for words like technical terms and proper names of persons, places, objects etc. that cannot be found in translation resources such as Hindi-Punjabi bilingual dictionary, surnames database, titles database etc and transliteration is an obvious choice for such words. (Goyal V. and Lehal G.S., 2009a).

2.7 Post-Processing

2.7.1 Agreement Corrections

In spite of the great similarity between Hindi and Punjabi, there are still a number of important agreement divergences in gender and number. The output generated by the translation engine phase becomes the input for post-processing phase. This phase will correct the agreement errors based on the rules implemented in the form of regular expressions. (Goyal V. and Lehal G.S., 2011)

3 Evaluation and Results

The evaluation document set consisted of documents from various online newspapers news, articles, blogs, biographies etc. This test bed consisted of 35500 words and was translated using our Machine Translation system.

3.1 Test Document

For our Machine Translation system evaluation, we have used benchmark sampling method for selecting the set of sentences. Input sentences are selected from randomly selected news (sports, politics, world, regional, entertainment, travel etc.), articles (published by various writers, philosophers etc.), literature (stories by Prem Chand, Yashwant jain etc.), Official language for office letters (The Language Officially used on the files in Government offices) and blogs (Posted by general public in forums etc.). Care has been taken to ensure that sentences use a variety of constructs. All possible constructs including simple as well as complex ones are incorporated in the set. The sentence set also contains all types of sentences such as declarative, interrogative, imperative and exclamatory. Sentence length is not restricted although care has been taken that single sentences do not become too long. Following table shows the test data set:

Table 1: Test data set for the evaluation of Hindi to Punjabi Machine Translation System

	Daily News	Articles	Official Language Quotes	Blog	Literature
Total Documents	100	50	01	50	20
Total Sentences	10,000	3,500	8,595	3,300	10,045
Total Words	93,400	21,674	36,431	15,650	95,580

3.2 Experiments

It is also important to choose appropriate evaluators for our experiments. Thus, depending upon the requirements and need of the above mentioned tests, 50 People of different professions were selected for performing experiments. 20 Persons were from villages that only knew Punjabi and did not know Hindi and 30 persons were from different professions having knowledge of both Hindi and Punjabi. Average ratings for the sentences of the individual translations were then summed up (separately according to intelligibility and accuracy) to get the average scores. Percentage of accurate sentences and intelligent sentences was also calculated separately by counting the number of sentences.

3.2.1 Intelligibility Evaluation

The evaluators do not have any clue about the source language i.e. Hindi. They judge each sentence (in target language i.e. Punjabi) on the basis of its comprehensibility. The target user is a layman who is interested only in the comprehensibility of translations. Intelligibility is effected by grammatical errors, mis-translations, and un-translated words.

3.2.1.1 Results

The response by the evaluators were analysed and following are the results:

- 70.3 % sentences got the score 3 i.e. they were perfectly clear and intelligible.
- 25.1 % sentences got the score 2 i.e. they were generally clear and intelligible.
- 3.5 % sentences got the score 1 i.e. they were hard to understand.
- 1.1 % sentences got the score 0 i.e. they were not understandable.

So we can say that about 95.40 % sentences are intelligible. These sentences are those which have score 2 or above. Thus, we can say that the direct approach can translate Hindi text to Punjabi Text with a consideably good accuracy.

3.2.2 Accuracy Evaluation / Fidelity Measure

The evaluators are provided with source text along with translated text. A highly intelligible output sentence need not be a correct translation of the source sentence. It is important to check whether the meaning of the source language sentence is preserved in the translation. This property is called accuracy.

3.2.2.1 Results

Initially Null Hypothesis is assumed i.e. the system's performance is NULL. The author assumes that system is dumb and does not produce any valuable output. By the intelligibility of the analysis and Accuracy analysis, it has been proved wrong.

The accuracy percentage for the system is found out to be 87.60%

Further investigations reveal that out of 13.40%:

- 80.6 % sentences achieve a match between 50 to 99%
- 17.2 % of remaining sentences were marked with less than 50% match against the correct sentences.

- Only 2.2 % sentences are those which are found unfaithful.

A match of lower 50% does not mean that the sentences are not usable. After some post editing, they can fit properly in the translated text. (Goyal, V., Lehal, G.S., 2009b)

3.2.2 BLEU Score:

As there is no Hindi –Parallel Corpus was available, thus for testing the system automatically, we generated Hindi-Parallel Corpus of about 10K Sentences. The BLEU score comes out to be 0.7801.

5 Conclusion

In this paper, a hybrid translation approach for translating the text from Hindi to Punjabi has been presented. The proposed architecture has shown extremely good results and if found to be appropriate for MT systems between closely related language pairs.

Copyright

Acknowlegement

We are thankful to Dr. Amba Kulkarni, University of Hyderabad for her support in providing technical assistance for developing this system.

References

Bharati, Akshar, Chaitanya, Vineet, Kulkarni, Amba P., Sangal, Rajeev. 1997. Anusaaraka: Machine Translation in stages. Vivek, A Quarterly in Artificial Intelligence, Vol. 10, No. 3. ,NCST, Banglore. India, pp. 22-25.

Goyal V., Lehal G.S. 2008. Comparative Study of Hindi and Punjabi Language Scripts, Napalese Linguistics, Journal of the Linguistics Society of Nepal, Volume 23, November Issue, pp 67-82.

Goyal V., Lehal, G. S. 2008a. Hindi Morphological Analyzer and Generator. In Proc.: 1st International Conference on Emerging Trends in Engineering and Technology, Nagpur, G.H.Raisoni College of Engineering, Nagpur, July16-19, 2008, pp. 1156-1159, IEEE Computer Society Press, California, USA.

Goyal V., Lehal G.S. 2009. Advances in Machine Translation Systems, Language In India, Volume 9, November Issue, pp. 138-150.

Goyal V., Lehal G.S. 2009a. A Machine Transliteration System for Machine Translation System: An Application on Hindi-Punjabi Language Pair. Atti Della Fondazione Giorgio Ronchi (Italy), Volume LXIV, No. 1, pp. 27-35.

Goyal V., Lehal G.S. 2009b. Evaluation of Hindi to Punjabi Machine Translation System. International Journal of Computer Science Issues, France, Vol. 4, No. 1, pp. 36-39.

Goyal V., Lehal G.S. 2010. Automatic Spelling Standardization for Hindi Text. In : 1st International Conference on Computer & Communication Technology, Moti Lal Nehru National Institute of technology, Allhabad, Sepetember 17-19, 2010, pp. 764-767, IEEE Computer Society Press, California.

Goyal V., Lehal G.S. 2011. N-Grams Based Word Sense Disambiguation: A Case Study of Hindi to Punjabi Machine Translation System. International Journal of Translation. (Accepted, In Print).

Goyal V., Lehal G.S. 2011a. Hindi to Punjabi Machine Translation System. In Proc.: International Conference for Information Systems for Indian Languages, Department of Computer Science, Punjabi University, Patiala, March 9-11, 2011, pp. 236-241, Springer CCIS 139, Germany.

Sharma R., Goyal V. 2011b. Named Entity Recognition Systems for Hindi using CRF Approach. In Proc.: International Conference for Information Systems for Indian Languages, Department of Computer Science, Punjabi University, Patiala, March 9-11, 2011, pp. 31-35, Springer CCIS 139, Germany.

The ACL Anthology Searchbench

Ulrich Schäfer Bernd Kiefer Christian Spurk Jörg Steffen Rui Wang
Language Technology Lab
German Research Center for Artificial Intelligence (DFKI)
D-66123 Saarbrücken, Germany
{ulrich.schaefer,kiefer,cspurk,steffen,wang.rui}@dfki.de
http://www.dfki.de/lt

Abstract

We describe a novel application for structured search in scientific digital libraries. The ACL Anthology Searchbench is meant to become a publicly available research tool to query the content of the ACL Anthology. The application provides search in both its bibliographic metadata and semantically analyzed full textual content. By combining these two features, very efficient and focused queries are possible. At the same time, the application serves as a showcase for the recent progress in natural language processing (NLP) research and language technology. The system currently indexes the textual content of 7,500 anthology papers from 2002–2009 with predicate-argument-like semantic structures. It also provides useful search filters based on bibliographic metadata. It will be extended to provide the full anthology content and enhanced functionality based on further NLP techniques.

1 Introduction and Motivation

Scientists in all disciplines nowadays are faced with a flood of new publications every day. In addition, more and more publications from the past become digitally available and thus even increase the amount. Finding relevant information and avoiding duplication of work have become urgent issues to be addressed by the scientific community.

The organization and preservation of scientific knowledge in scientific publications, vulgo text documents, thwarts these efforts. From a viewpoint of a computer scientist, scientific papers are just 'unstructured information'. At least in our own scientific community, Computational Linguistics, it is generally assumed that NLP could help to support search in such document collections.

The ACL Anthology[1] is a comprehensive electronic collection of scientific papers in our own field (Bird et al., 2008). It is updated regularly with new publications, but also older papers have been scanned and are made available electronically.

We have implemented the ACL Anthology Searchbench[2] for two reasons: Our first aim is to provide a more targeted search facility in this collection than standard web search on the anthology website. In this sense, the Searchbench is meant to become a service to our own community.

Our second motivation is to use the developed system as a showcase for the progress that has been made over the last years in precision-oriented deep linguistic parsing in terms of both efficiency and coverage, specifically in the context of the DELPH-IN community[3]. Our system also uses further NLP techniques such as unsupervised term extraction, named entity recognition and part-of-speech (PoS) tagging.

By automatically precomputing normalized semantic representations (predicate-argument structure) of each sentence in the anthology, the search space is structured and allows to find equivalent or related predicates even if they are expressed differ-

[1] http://www.aclweb.org/anthology

[2] http://aclasb.dfki.de

[3] http://www.delph-in.net – DELPH-IN stands for DEep Linguistic Processing with HPSG INitiative.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 7–13,
Portland, Oregon, USA, 21 June 2011.

ently, e.g. in passive constructions, using synonyms, etc. By storing the semantic sentence structure along with the original text in a structured full-text search engine, it can be guaranteed that recall cannot fall behind the baseline of a fulltext search.

In addition, the Searchbench also provides detailed bibliographic metadata for filtering as well as autosuggest texts for input fields computed from the corpus – two further key features one can expect from such systems today, nevertheless very important for efficient search in digital libraries.

We describe the offline preprocessing and deep parsing approach in Section 2. Section 3 concentrates on the generation of the semantic search index. In Section 4, we describe the search interface. We conclude in Section 5 and present an outlook to future extensions.

2 Parsing the ACL Anthology

The basis of the search index for the ACL Anthology are its original PDF documents, currently 8,200 from the years 2002 through 2009. To overcome quality problems in text extraction from PDF, we use a commercial PDF extractor based on OCR techniques. This approach guarantees uniform and high-quality textual representations even from older papers in the anthology (before 2000) which mostly were scanned from printed paper versions.

The general idea of the semantics-oriented access to scholarly paper content is to parse each sentence they contain with the open-source HPSG (Pollard and Sag, 1994) grammar for English (ERG; Flickinger (2002)) and then distill and index semantically structured representations for search.

To make the deep parser robust, it is embedded in a NLP workflow. The coverage (percentage of full deeply parsed sentences) on the anthology corpus could be increased from 65 % to now more than 85 % through careful combination of several robustness techniques; for example: (1) *chart pruning*, directed search during parsing to increase performance, and also coverage for longer sentences (Cramer and Zhang, 2010); (2) *chart mapping*, a novel method for integrating preprocessing information in exactly the way the deep grammar expects it (Adolphs et al., 2008); (3) new version of the ERG with better handling of open word classes; (4) more fine-grained named entity recognition, including recognition of citation patterns; (5) new, better suited parse ranking model (WeScience; Flickinger et al. (2010)). Because of limited space, we will focus on (1) and (2) below. A more detailed description and further results are available in (Schäfer and Kiefer, 2011).

Except for a small part of the named entity recognition components (citations, some terminology) and the parse ranking model, there are no further adaptations to genre or domain of the text corpus. This implies that the NLP workflow could be easily and modularly adapted to other (scientific or non-scientific) domains—mainly thanks to the generic and comprehensive language modelling in the ERG.

The NLP preprocessing component workflow is implemented using the Heart of Gold NLP middleware architecture (Schäfer, 2006). It starts with sentence boundary detection (SBR) and regular expression-based tokenization using its built-in component JTok, followed by the trigram-based PoS tagger TnT (Brants, 2000) trained on the Penn Treebank (Marcus et al., 1993) and the named entity recognizer SProUT (Drożdżyński et al., 2004).

2.1 Precise Preprocessing Integration with Chart Mapping

Tagger output is combined with information from the named entity recognizer, e.g. delivering hypothetical information on citation expressions. The combined result is delivered as input to the deep parser PET (Callmeier, 2000) running the ERG. Here, citations, for example, can be treated as either persons, locations or appositions.

Concerning punctuation, the ERG can make use of information on opening and closing quotation marks. Such information is often not explicit in the input text, e.g. when, as in our setup, gained through OCR which does not distinguish between ‘ and ’ or “ and ”. However, a tokenizer can often guess (reconstruct) leftness and rightness correctly. This information, passed to the deep parser via chart mapping, helps it to disambiguate.

2.2 Increased Processing Speed and Coverage through Chart Pruning

In addition to a well-established discriminative maximum entropy model for post-analysis parse selec-

tion, we use an additional generative model as described in Cramer and Zhang (2010) to restrict the search space during parsing. This restriction increases efficiency, but also coverage, because the parse time was restricted to at most 60 CPU seconds on a standard PC, and more sentences could now be parsed within these bounds. A 4 GB limit for main memory consumption was far beyond what was ever needed. We saw a small but negligible decrease in parsing accuracy, 5.4 % best parses were not found due to the pruning of important chart edges.

Ninomiya et al. (2006) did a very thorough comparison of different performance optimization strategies, and among those also a local pruning strategy similar to the one used here. There is an important difference between the systems, in that theirs works on a reduced context-free backbone first and reconstructs the results with the full grammar, while PET uses the HPSG grammar directly, with subsumption packing and partial unpacking to achieve a similar effect as the packed chart of a context-free parser.

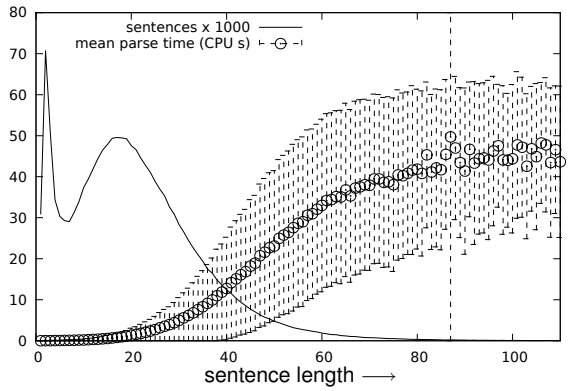


Figure 1: Distribution of sentence length and mean parse times for mild pruning

In total, we parsed 1,537,801 sentences, of which 57,832 (3.8 %) could not be parsed because of lexicon errors. Most of them were caused by OCR artifacts resulting in unexpected punctuation character combinations. These can be identified and will be deleted in the future.

Figure 1 displays the average parse time of processing with a mild chart pruning setting, together with the mean quadratic error. In addition, it contains the distribution of input sentences over sentence length. Obviously, the vast majority of sentences has a length of at most 60 words[4]. The parse times only grow mildly due to the many optimization techniques in the original system, and also the new chart pruning method. The sentence length distribution has been integrated into Figure 1 to show that the predominant part of our real-world corpus can be processed using this information-rich method with very low parse times (overall average parse time < 2 s per sentence).

The large amount of short inputs is at first surprising, even more so that most of these inputs can not be parsed. Most of these inputs are non-sentences such as headings, enumerations, footnotes, table cell content. There are several alternatives to deal with such input, one to identify and handle them in a preprocessing step, another to use a special root condition in the deep analysis component that is able to combine phrases with well-defined properties for inputs where no spanning result could be found.

We employed the second method, which has the advantage that it handles a larger range of phenomena in a homogeneous way. Figure 2 shows the change in percentage of unparsed and timed out inputs for the mild pruning method with and without the root condition combining fragments.

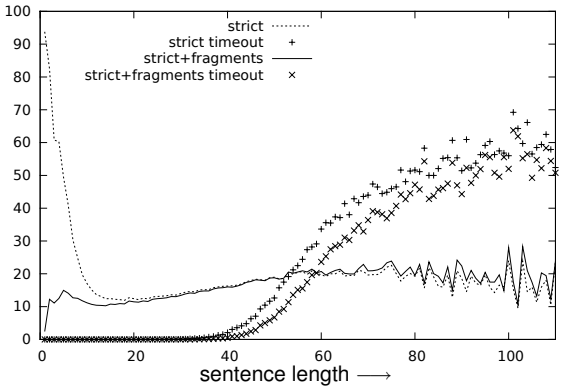


Figure 2: Unparsed and timed out sentences with and without fragment combination

Figure 2 shows that this changes the curve for unparsed sentences towards more expected characteristics and removes the uncommonly high percentage of short sentences for which no parse can be computed. Together with the parses for fragmented

[4]It has to be pointed out that extremely long sentences also may be non-sentences resulting from PDF extraction errors, missing punctuation etc. No manual correction took place.

Semantic content of the sentence you have selected: "Our systems achieved good performance on the SRL task, easily beating the baseline."

SEMANTIC SUBJECT	SEMANTIC PREDICATE	SEMANTIC FIRST OBJECT	SEMANTIC SECOND OBJECT	ADJUNCTS
	beating	the baseline.		easily
Our systems	achieved	good performance on the SRL task,		

Figure 3: Multiple semantic tuples may be generated for a sentence

input, we get a recall (sentences with at least one parse) over the whole corpus of 85.9 % (1,321,336 sentences), without a significant change for any of the other measures, and with potential for further improvement.

3 Semantic Tuple Extraction with DMRS

In contrast to shallow parsers, the ERG not only handles detailed syntactic analyses of phrases, compounds, coordination, negation and other linguistic phenomena that are important for extracting semantic relations, but also generates a formal semantic representation of the meaning of the input sentence in the Minimal Recursion Semantics (MRS) representation format (Copestake et al., 2005). It consists of elementary predications for each word and larger constituents, connected via argument positions and variables, from which predicate-argument structure can be extracted.

MRS representations resulting from deep parsing are still relatively close to linguistic structures and contain more detailed information than a user would like to query and search for. Therefore, an additional extraction and abstraction step is performed before storing semantic structures in the search index.

Firstly, MRS is converted to DMRS (Copestake, 2009), a dependency-style version of MRS that eases extraction of predicate-argument structure using the implementation in LKB (Copestake, 2002). The representation format we devised for the search index we call *semantic tuples*, in fact quintuples <subject, predicate, first object, second object, adjuncts>; example in Figure 3. The basic extraction algorithm consists of the following three steps: (1) calculate the closure for each elementary predication based on the EQ (variable equivalence) relation, and group the predicates and entities in each closure respectively; (2) extract the relations of the groups, which results in a graph as a whole; (3) recursively traverse the graph, form one semantic tuple for each predicate, and fill in the corresponding information under its scope, i.e. subject, object, etc.

In the example shown in Figure 3, entity groups like 'our systems', 'the baseline', and 'good performance on the SRL task', as well as predicate groups 'beating' and 'achieved' are formed at the first step. In the second step, the graph structure is extracted, i.e., the relation between the groups. Finally, two semantic tuples are filled in with both the predicates and the corresponding information. Notice that the modifier(s) of the entity belong to the same entity group, but the modifier(s) of the predicate will be put into the Adjuncts slot. Similarly, the coordination of the entities will be put into one entity group, while the coordination of predicates will form multiple semantic tuples.

Since we are extracting predicate-argument structure, syntactic variations such as passive constructions and relative clauses will be all 'normalized' into the same form. Consequently, 'the book which I read', 'I read the book', and 'the book was read by me' will form the exact same semantic tuple <I, read, the book, N/A, N/A>. The resulting tuple structures along with their associated text are stored in an Apache Solr/Lucene[5] server which receives queries from the Searchbench user interface.

4 Searchbench User Interface

The Searchbench user interface (UI) is a web application running in every modern, JavaScript-enabled web browser. As can be seen in Figure 4, the UI is divided into three parts: (1) a sidebar on the left (*Filters View*), where different filters can be set that constrain the list of found documents; (2) a list of found documents matching the currently set filters in the upper right part of the UI (*Results View*); (3)

[5] http://lucene.apache.org/solr

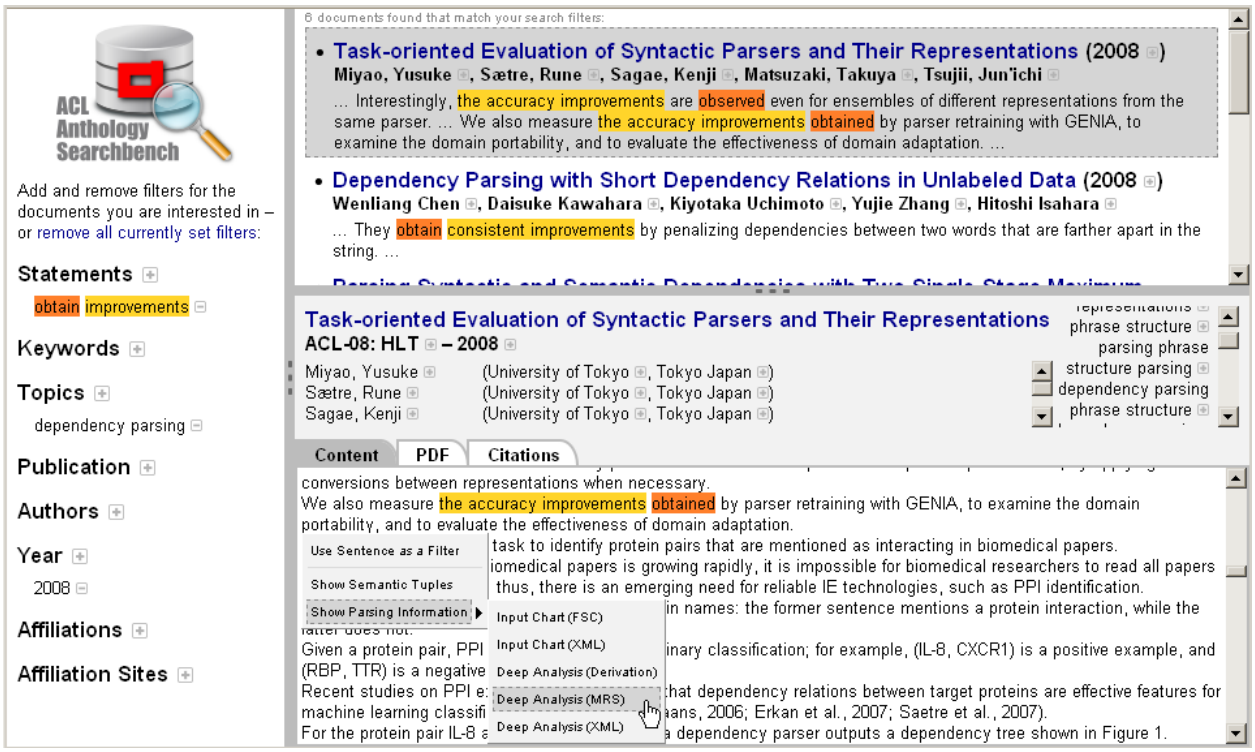


Figure 4: Searchbench user interface with different filters set and currently looking at the debug menu for a sentence.

the *Document View* in the lower right part of the UI with different views of the current document.

A focus in the design of the UI has been to allow the user to very quickly browse the papers of the ACL Anthology and then to find small sets of relevant documents based on metadata and content. This is mainly achieved by these techniques: (i) changes in the collection of filters automatically update the Results View; (ii) metadata and searchable content from both the Results View and the Document View can easily be used with a single click as new filters; (iii) filters can easily be removed with a single click; (iv) manually entering filter items is assisted by sensible autosuggestions computed from the corpus; (v) accidental filter changes can easily be corrected by going back in the browser history.

The following kinds of filters are supported: **Statements** (filter by semantic statements, i.e., the actual content of sentences, see Section 4.1), **Keywords** (filter by simple keywords with a full-text search), **Topics** (filter by topics of the articles that were extracted with an extended approach of the unsupervised term extractor of Frantzi et al. (1998)), **Publication** (filter by publication title/event), **Authors** (filter by author names), **Year** (filter by publication year), **Affiliations** (filter by affiliation organizations), **Affiliation Sites** (filter by affiliation cities and countries)[6]. Found papers always match *all* currently set filters. For each filter type multiple different filter items can be set; one could search for papers written jointly by people from different research institutes on a certain topic, for example. Matches of the statements filter and the keywords filter are highlighted in document snippets for each paper in the Results View and in the currently selected paper of the Document View.

Besides a header displaying the metadata of the currently selected paper (including the automatically extracted topics on the right), the Document View provides three subviews of the selected paper: (1) the *Document Content View* is a raw list of the sentences of the paper and provides different kinds of interaction with these sentences; (2) the *PDF View* shows the original PDF version of the paper; (3) the *Citations View* provides citation information includ-

[6]Affiliations have been added using the ACL Anthology Network data (Radev et al., 2009).

ing link to the ACL Anthology Network (Radev et al., 2009).

Figure 4 shows the search result for a query combining a statement ('obtain improvements'), a topic 'dependency parsing' and the publication year 2008. As can be seen in the Results View, six papers match these filters; sentences with semantically similar predicates and passive voice are found, too.

4.1 Semantic Search

The main feature which distinguishes the ACL Anthology Searchbench from other search applications for scientific papers is the semantic search in paper content. This enables the search for (semantic) statements in the paper content as opposed to searching for keywords in the plain text. Our use of the term "statement" is loosely along the lines of the same term used in logic. Very simple sentences often bear a single statement only, while more complex sentences (especially when having multiple clauses) contain multiple statements. Each of the semantic tuples extracted from the papers of the ACL Anthology (cf. Section 3) corresponds to a statement.

The Statements filter is responsible for the semantic search. Statements used in filters may be underspecified, e.g., one may search for statements with a certain semantic subject but with arbitrary semantic predicates and objects. There are two ways in which a new statement filter can be set: (1) entering a statement manually; (2) clicking a sentence in the Document Content View and choosing the statements of this sentence that shall be set as new statement filters (cf. Figure 5), i.e. it is possible to formulate and refine queries 'by example'.

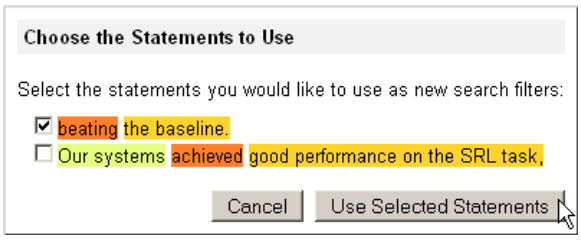


Figure 5: Dialog for choosing statements to be used as new filters (for sentence "Our systems achieved good performance on the SRL task, easily beating the baseline.").

Throughout the user interface, no distinction is made between the different kinds of semantic objects and adjuncts so as to make it easy also for non-linguists to use the search and to be more robust against bad analyses of the parser. Therefore, the different semantic parts of a statement are highlighted in three different colors only, depending on whether a part is the semantic subject, the semantic predicate or anything else (object/adjunct).

In order to disengage even further from the concrete wording and make the semantic search even more 'meaning-based', we additionally search for synonyms of the semantic predicates in statement filters. These synonyms have been computed as an intersection of the most frequent verbs (semantic predicates) in the anthology corpus with WordNet synsets (Fellbaum, 1998), the main reason being reduction of the number of meanings irrelevant for the domain. This relatively simple approach could of course be improved, e.g. by active learning from user clicks in search results etc.

5 Summary and Outlook

We have described the ACL Anthology Searchbench, a novel search application for scientific digital libraries. The system is fully implemented and indexes 7,500 papers of the 8,200 parsed ones. For the other 700, bibliographic metadata was missing. These and the remaining 10,000 papers are currently being processed and will be added to the search index. The goal of the Searchbench is both to serve as a showcase for benefits and improvement of NLP for text search and at the same time provide a useful tool for researchers in Computational Linguistics. We believe that the tool by now already supports targeted search in a large collection of digital research papers better than standard web search engines. An evaluation comparing Searchbench query results with web search is in progress.

Optionally, the Searchbench runs in a linguistic debug mode providing NLP output a typical user would not need. These analyses are accessible from a context menu on each sentence (cf. Figure 4). Both a tabular view of the semantic tuples of a sentence (cf. Figure 3) and different kinds of information related to the parsing of the sentence (including the MRS and a parse tree) can be displayed.

Future work, for which we are urgently seeking funding, could include integration of further

NLP-based features such as coreference resolution or question answering, as well as citation classification and graphical navigation along the ideas in Schäfer and Kasterka (2010).

Acknowledgments

We are indebted to Peter Adolphs, Bart Cramer, Dan Flickinger, Stephan Oepen, Yi Zhang for their support with ERG and PET extensions such as chart mapping and chart pruning. Melanie Reiplinger, Benjamin Weitz and Leonie Grön helped with preprocessing. We also thank the anonymous reviewers for their encouraging comments. The work described in this paper has been carried out in the context of the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research, and in the context of the world-wide DELPH-IN consortium.

References

Peter Adolphs, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Daniel Flickinger, and Bernd Kiefer. 2008. Some fine points of hybrid natural language parsing. In *Proceedings of LREC-2008*, pages 1380–1387, Marrakesh, Morocco.

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 *Proceedings of LREC-2008*, pages 1755–1759, Marrakesh, Morocco.

Torsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proc. of ANLP*, pages 224–231, Seattle, WA.

Ulrich Callmeier. 2000. PET – A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–108.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl Pollard. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3(2–3):281–332.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI publications, Stanford.

Ann Copestake. 2009. Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proc. of EACL*, pages 1–9.

Bart Cramer and Yi Zhang. 2010. Constraining robust constructions for broad-coverage parsing with precision grammars. In *Proceedings of COLING-2010*, pages 223–231, Beijing, China.

Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. Shallow processing with unification and typed feature structures – foundations and applications. *Künstliche Intelligenz*, 2004(1):17–23.

Christiane Fellbaum, editor. 1998. *WordNet, An Electronic Lexical Database*. MIT Press.

Dan Flickinger, Stephan Oepen, and Gisle Ytrestøl. 2010. WikiWoods: Syntacto-semantic annotation for English Wikipedia. In *Proceedings of LREC-2010*, pages 1665–1671.

Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Dan Flickinger, Stephan Oepen, Hans Uszkoreit, and Jun'ichi Tsujii, editors, *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*, pages 1–17. CSLI Publications, Stanford, CA.

Katerina T. Frantzi, Sophia Ananiadou, and Jun'ichi Tsujii. 1998. The C-value/NC-value method of automatic recognition for multi-word terms. In *Proceedings of ECDL*, pages 585–604.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, 19:313–330.

Takashi Ninomiya, Yoshimasa Tsuruoka, Yusuke Miyao, Kenjiro Taura, and Jun'ichi Tsujii. 2006. Fast and scalable HPSG parsing. *Traitement automatique des langues (TAL)*, 46(2).

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *Proceedings of the ACL-2009 Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, Singapore.

Ulrich Schäfer and Uwe Kasterka. 2010. Scientific authoring support: A tool to navigate in typed citation graphs. In *Proceedings of the NAACL-HLT 2010 Workshop on Computational Linguistics and Writing*, pages 7–14, Los Angeles, CA.

Ulrich Schäfer and Bernd Kiefer. 2011. Advances in deep parsing of scholarly paper content. In Raffaella Bernardi, Sally Chambers, Björn Gottfried, Frédérique Segond, and Ilya Zaihrayeu, editors, *Advanced Language Technologies for Digital Libraries*, LNCS Hot Topics Series. Springer. to appear.

Ulrich Schäfer. 2006. Middleware for creating and combining multi-dimensional NLP markup. In *Proceedings of the EACL-2006 Workshop on Multi-dimensional Markup in Natural Language Processing*, pages 81–84, Trento, Italy.

Exploiting Readymades in Linguistic Creativity:

A System Demonstration of the *Jigsaw Bard*

Tony Veale
School of Computer Science and Informatics,
University College Dublin,
Belfield, Dublin D4, Ireland.
Tony.Veale@UCD.ie

Yanfen Hao
School of Computer Science and Informatics,
University College Dublin,
Belfield, Dublin D4, Ireland.
Yanfen.Hao@UCD.ie

Demonstration System can be viewed at: *http://www.educatedinsolence.com/jigsaw*

Abstract

Large lexical resources, such as corpora and databases of Web ngrams, are a rich source of pre-fabricated phrases that can be reused in many different contexts. However, one must be careful in how these resources are used, and noted writers such as George Orwell have argued that the use of canned phrases encourages sloppy thinking and results in poor communication. Nonetheless, while Orwell prized home-made phrases over the readymade variety, there is a vibrant movement in modern art which shifts artistic creation from the production of novel artifacts to the clever reuse of readymades or *objets trouvés*. We describe here a system that makes creative reuse of the linguistic readymades in the *Google* ngrams. Our system, the *Jigsaw Bard,* thus owes more to Marcel Duchamp than to George Orwell. We demonstrate how textual readymades can be identified and harvested on a large scale, and used to drive a modest form of linguistic creativity.

1 Introduction

In a much-quoted essay from 1946 entitled *Politics and the English Language*, the writer and thinker George Orwell outlines his prescription for halting a perceived decline in the English language. He argues that language and thought form a tight feedback cycle that can be either virtuous or vicious. Lazy language can thus promote lazy thinking, and vice versa. Orwell pours scorn on two particular forms of lazy language: the expedient use of overly familiar metaphors merely because they come quickly to mind, even though they have lost their power to evoke vivid images,; and the use of readymade turns of phrase as substitutes for individually crafted expressions. While a good writer bends words to his meaning, Orwell worries that a lazy writer bends his meaning to convenient words.

Orwell is especially scornful about readymade phrases which, when over-used, "are tacked together like the sections of a prefabricated henhouse." A writer who operates by "mechanically repeating the familiar phrases" and "gumming together long strips of words which have already been set in order by someone else" has, he argues, "gone some distance toward turning himself into a machine." Given his derogatory mechanistic view of the use of readymade phrases, Orwell would not be surprised to learn that computers are highly proficient in the large-scale use of familiar phrases, whether acquired from large text corpora or from the Google ngrams (see Brants and Franz, 2006).

Though argued with passion, there are serious holes in Orwell's logic. If one should "never use a metaphor, simile or other figure of speech which you are used to seeing in print", how then are familiar metaphors ever to become *dead* metaphors and thereby enrich the language with new terms and new senses? And if one cannot use familiar readymade phrases, how can one make playful – and creative – allusions to the writings of others, or

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 14–19,
Portland, Oregon, USA, 21 June 2011.

mischievously subvert the conventional wisdom of platitudes and clichés? Orwell's use of the term *readymade* is entirely negative, yet the term is altogether more respectable in the world of modern art, thanks to its use by artists such as Marcel Duchamp. For many artists, a readymade object is not a substitute, but a starting point, for creativity.

Also called an *objet trouvé* or *found object*, a readymade emerges from an artist's encounter with an object whose aesthetic merits are overlooked in its banal, everyday contexts of use; when this object is moved to an explicitly artistic context, such as an art gallery, viewers are better able to appreciate these merits. The artist's insight is to recognize the transformational power of this non-obvious context switch. Perhaps the most famous (and notorious) readymade in the world of art is Marcel Duchamp's *Fountain*, a humble urinal that becomes an elegantly curved piece of sculpture when viewed with the right mindset. Duchamp referred to his *objets trouvés* as "assisted readymades" because they allow an artist to remake the act of creation as one of pure insight and inspired recognition rather than one of manual craftsmanship (see Taylor, 2009). In computational terms, the Duchampian notion of a readymade allows creativity to be modeled not as a construction problem but as a decision problem. A computational Duchamp need not explore an abstract conceptual space of potential ideas, as in Boden (1994). However, a Duchampian agent must instead be exposed to the multitude of potentially inspiring real-world stimuli that a human artist encounters everyday.

Readymades represent a serendipitous form of creativity that is poorly served by exploratory models of creativity, such as that of Boden (1994), and better served by the investment models such as the *buy-low-sell-high* theory of Sternberg and Lubart (1995). In this view, creators and artists find unexpected or untapped value in unfashionable objects or ideas that already exist, and quickly move their gaze elsewhere once the public at large come to recognize this value. Duchampian creators invest in everyday objects, just as Duchamp found artistic merit in urinals, bottles and combs. From a linguistic perspective, these everyday objects are commonplace words and phrases which, when wrenched from their conventional contexts of use, are free to take on enhanced meanings and provide additional returns to the investor. The realm in which a maker of linguistic readymades operates is not the real world, and not an abstract conceptual space, but the realm of texts: large corpora become rich hunting grounds for investors in linguistic *objets trouvés*.

This proposal is demonstrated in computational form in the following sections. We show how a rich vocabulary of cultural stereotypes can be acquired from the Web, and how this vocabulary facilitates the implementation of a decision procedure for recognizing potential readymades in large corpora – in this case, the Google database of Web ngrams (Brants and Franz, 2006). This decision procedure provides a robust basis for a simile-generation system called *The Jigsaw Bard*. The cognitive / linguistic intuitions that underpin the *Bard*'s concept of textual readymades are put to the empirical test in section 5. While readymades remain a contentious notion in the public's appreciation of artistic creativity – despite Duchamp's *Fountain* being considered one of the most influential artworks of the 20th century – we shall show that the notion of a linguistic readymade has significant practical merit in the realms of text generation and computational creativity.

2 Linguistic Readymades

Readymades are the result of artistic *appropriation*, in which an object with cultural resonance – an image, a phrase, a quote, a name, a thing – is reused in a new context with a new meaning. As a fertile source of cultural reference points, language is an equally fertile medium for appropriation. Thus, in the constant swirl of language and culture, movie quotes suggest song lyrics, which in turn suggest movie titles, which suggest book titles, or restaurant names, or the names of racehorses, and so on, and on. The 1996 movie *The Usual Suspects* takes its name from a memorable scene in 1942's *Casablanca*, as does the Woody Allen play and movie *Play it Again Sam*. The 2010 art documentary *Exit Through the Gift Shop*, by graffiti artist Banksy, takes its name from a banal sign sometimes seen in museums and galleries: the sign, suggestive as it is of creeping commercialism, makes the perfect readymade for a film that laments the mediocrity of commercialized art.

Appropriations can also be combined to produce novel mashups; consider, for instance, the use of tweets from rapper Kanye West as alternate

captions for cartoon images from the *New Yorker* magazine (see hashtag *#KanyeNew-YorkerTweets*). Hashtags can themselves be linguistic readymades. When free-speech advocates use the hashtag *#IAMSpartacus* to show solidarity with users whose tweets have incurred the wrath of the law, they are appropriating an emotional line from the 1960 film *Spartacus*. Linguistic readymades, then, are well-formed text fragments that are often highly quotable because they carry some figurative content which can be reused in different contexts.

A quote like "*round up the usual suspects*" or "*I am Spartacus*" requires a great deal of cultural knowledge to appreciate. Since literal semantics only provides a small part of their meaning, a computer's ability to recognize linguistic readymades is only as good as the cultural knowledge at its disposal. We thus explore here a more modest form of readymade – phrases that can be used as evocative image builders in similes – as in:

a wet haddock
snow in January
a robot fish
a bullet-ridden corpse

Each phrase can be found in the Google 1T database of Web ngrams – snippets of Web text (of one to five words) that occur on the web with a frequency of 40 or higher (Brants and Franz, 2006). Each is likely a literal description of a real object or event – even "robot fish", which describes an autonomous marine vehicle whose movements mimic real fish. But each exhibits figurative potential as well, providing a memorable description of physical or emotional coldness. Whether or not each was ever used in a figurative sense before is not the point: once this potential is recognized, each phrase becomes a reusable linguistic readymade for the construction of a vivid figurative comparison, as in "*as cold as a robot fish*". We now consider the building blocks from which these comparisons can be ready-made..

3 A Vocabulary of Cultural Stereotypes

How does a computer acquire the knowledge that fish, snow, January, bullets and corpses are cultural signifiers of coldness? Much the same way that humans acquire this knowledge: by attending to the way these signifiers are used by others, especially when they are used in cultural clichés like proverbial similes (e.g., "as cold as a fish").

In fact, folk similes are an important vector in the transmission of cultural knowledge: they point to, and exploit, the shared cultural touchstones that speakers and listeners alike can use to construct and intuit meanings. Taylor (1954) catalogued thousands of proverbial comparisons and similes from California, identifying just as many building blocks in the construction of new phrases and figurative meanings. Only the most common similes can be found in dictionaries, as shown by Norrick (1986), while Moon (2008) demonstrates that large-scale corpus analysis is needed to identify folk similes with a breadth approaching that of Taylor's study. However, Veale and Hao (2007) show that the World-Wide Web is the ultimate resource for harvesting similes.

Veale and Hao use the Google API to find many instances of the pattern "*as ADJ as a|an* *" on the web, where ADJ is an adjectival property and * is the Google wildcard. WordNet (Fellbaum, 1998) is used to provide a set of over 2,000 different values for ADJ, and the text snippets returned by Google are parsed to extract the basic simile bindings. Once the bindings are annotated to remove noise, as well as frequent uses of irony, this Web harvest produces over 12,000 cultural bindings between a noun (such as *fish*, or *robot*) and its most stereotypical properties (such as *cold, wet, stiff, logical, heartless*, etc.). Stereotypical properties are acquired for approx. 4,000 common English nouns. This is a set of building blocks on a larger scale than even that of Taylor, allowing us to build on Veale and Hao (2007) to identify readymades in their hundreds of thousands in the Google ngrams.

However, to identify readymades as resonant variations on cultural stereotypes, we need a certain fluidity in our treatment of adjectival properties. The phrase "*wet haddock*" is a readymade for coldness because "wet" accentuates the "cold" that we associate with "haddock" (via the web simile "*as cold as a haddock*"). In the words of Hofstadter (1995), we need to build a *SlipNet* of properties whose structure captures the propensity of properties to mutually and coherently reinforce each other, so that phrases which subtly accentuate an unstated property can be recognized. In the vein of Veale and Hao (2007), we use the Google API to harvest the elements of this SlipNet.

We hypothesize that the construction "*as* ADJ_1 *and* ADJ_2 *as*" shows ADJ_1 and ADJ_2 to be mutually reinforcing properties, since they can be seen to work together as a single complex property in a single comparison. Thus, using the full complement of adjectival properties used by Veale and Hao (2007), we harvest all instances of the patterns "*as ADJ and * as*" and "*as * and ADJ as*" from Google, noting the combinations that are found and their frequencies. These frequencies provide link weights for the Hofstadter-style SlipNet that is then constructed. In all, over 180,000 links are harvested, connecting over 2,500 adjectival properties to one other. We put the intuitions behind this SlipNet to the empirical test in section five.

4 Harvesting Readymades from Corpora

In the course of an average day, a creative writer is exposed to a constant barrage of linguistic stimuli, any small portion of which can strike a chord as a potential readymade. In this casual inspiration phase, the observant writer recognizes that a certain combination of words may produce, in another context, a meaning that is more than the sum of its parts. Later, when an apposite phrase is needed to strike a particular note, this combination may be retrieved from memory (or from a trusty notebook), *if* it has been recorded and suitably indexed.

Ironically, Orwell (1946) suggests that lazy writers "shirk" their responsibility to be "scrupulous" in their use of language by "simply throwing [their] mind open and letting the ready-made phrases come crowding in". For Orwell, words just get in the way, and should be kept at arm's length until the writer has first allowed a clear meaning to crystallize. This is dubious advice, as one expects a creative writer to keep an open mind when considering *all* the possibilities that present themselves. Yet Orwell's proscription suggests how a computer should go about the task of harvesting readymades from corpora: by throwing its mind open to the possibility that a given ngram may one day have a second life as a creative readymade in another context, the computer allows the phrases that match some simple image-building criteria to come crowding in, so they can be stored in a database.

Given a rich vocabulary of cultural stereotypes and their properties, computers are capable of indexing and recalling a considerably larger body of resonant combinations than the average human. The necessary barrage of linguistic stimuli can be provided by the Google 1T database of Web ngrams (Brants and Franz, 2006). Trawling these ngrams, a modestly creative computer can recognize well-formed combinations of cultural elements that might serve as a vivid vehicle of description in a future comparison. For every phrase **P** in the ngrams, where **P** combines stereotype nouns and/or adjectival modifiers, the computer simply poses the following question: is there an unstated property **A** such that the simile "*as **A** as **P***" is a meaningful and memorable comparison? The property **A** can be simple, as in "as *dark* as a chocolate espresso", or complex, as in "as *dark and sophisticated* as a chocolate martini". In either case, the phrase **P** is tucked away, and indexed under the property **A** until such time as the computer needs to produce a vivid evocation of **A**.

The following patterns are used to identify potential readymades in the Web ngrams:

(1) $Noun_{S1}$ $Noun_{S2}$
where both nouns denote stereotypes that share an unstated property Adj_A. The property Adj_A serves to index this combination. Example: "as cold as a *robot fish*".

(2) $Noun_{S1}$ $Noun_{S2}$
where both nouns denote stereotypes with salient properties Adj_{A1} and Adj_{A2} respectively, such that Adj_{A1} and Adj_{A2} are mutually reinforcing. The combination is indexed on Adj_{A1}+Adj_{A2}. Example: "as dark and sophisticated as a *chocolate martini*".

(3) Adj_A $Noun_S$
where $Noun_S$ denotes a cultural stereotype, and the adjective Adj_A denotes a property that mutually reinforces an unstated but salient property Adj_{SA} of the stereotype. Example: "as cold as a *wet haddock*". The combination is indexed on Adj_{SA}.

More complex structures for **P** are also possible, as in the phrases "*a lake of tears*" (a melancholy way to accentuate the property "wet") and "*a statue in a library*" (for "silent" and "quiet"). In this current description, we focus on 2-gram phrases only.

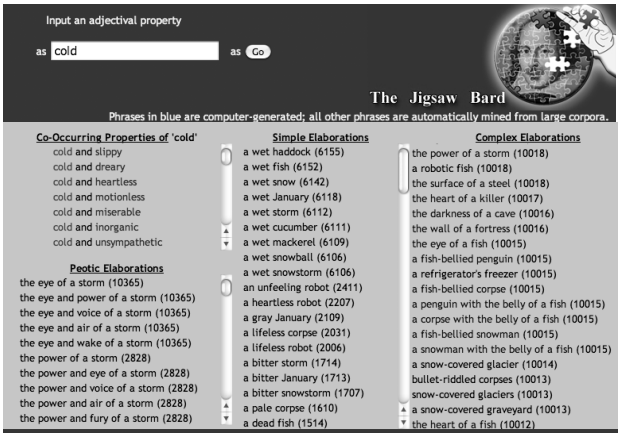


Figure 1. *Screenshot of The Jigsaw Bard, retrieving linguistic readymades for the input property "cold". See http://www.educatedinsolence.com/jigsaw*

Using these patterns, our application – the *Jigsaw Bard* (see Figure 1) – pre-builds a vast collection of figurative similes well in advance of the time it is asked to use or suggest any of them. Each phrase **P** is syntactically well-formed, and because **P** occurs relatively frequently on the Web, it is likely to be semantically well-formed as well. Just as Duchamp side-stepped the need to physically originate anything, but instead appropriated prefabricated artifacts, the *Bard* likewise side-steps the need for natural-language generation. Each phrase it proposes has the ring of linguistic authenticity; because this authenticity is rooted in another, more literal context, the *Bard* also exhibits its own Duchamp-like (if Duchamp-*lite*) creativity. We now consider the scale of the *Bard*'s generativity, and the quality of its insights.

5 Empirical Evaluation

The vastness of the web, captured in the large-scale sample that is the Google ngrams, means the *Jigsaw Bard* finds considerable grist for its mill in the phrases that match (1)…(3). Thus, the most restrictive pattern, pattern (1), harvests approx. 20,000 phrases from the Google 2-grams, for almost a thousand simple properties (indexing an average of 29 phrases under each property, such as "*swan song*" for "*beautiful*"). Pattern (2) – which allows a blend of stereotypes to be indexed under a complex property – harvests approx. 170,000 phrases from the 2-grams, for approx. 70,000 complex properties (indexing an average of 12 phrases under each, such as "*hospital bed*" for "*comfortable and safe*"). Pattern (3) – which pairs a stereotype noun with an adjective that draws out a salient property of the stereotype – is similarly productive: it harvests approx. 150,000 readymade 2-grams for over 2,000 simple properties (indexing an average of 125 phrases per property, as in "*youthful knight*" for "heroic" and "*zealous convert*" for "devout").

The *Jigsaw Bard* is best understood as a creative thesaurus: for any given property (or blend of properties) selected by the user, the *Bard* presents a range of apt similes constructed from linguistic readymades. The numbers above show that, recall-wise, the *Bard* has sufficient coverage to work robustly as a thesaurus. Quality-wise, users must make their own determinations as to which similes are most suited to their descriptive purposes, yet it is important that suggestions provided by the *Bard* are sensible and well-motivated. As such, we must be empirically satisfied about two key intuitions: first, that salient properties are indeed acquired from the Web for our vocabulary of stereotypes (this point relates to the aptness of the similes suggested by the *Bard*); and second, that the adjectives connected by the SlipNet really do mutually reinforce each other (this point relates to the coherence of complex properties, and to the ability of readymades to accentuate unstated properties).

Both intuitions can be tested using Whissell's (1989) dictionary of affect, a psycholinguistic resource used for sentiment analysis that assigns a pleasantness score of between 1.0 (least pleasant) and 3.0 (most pleasant) to over 8,000 commonplace words. We should thus be able to predict the pleasantness of a stereotype noun (like *fish*) using a weighted average of the pleasantness of its salient properties (like *cold*, *slippery*). We should also be able to predict the pleasantness of an adjective using a weighted average of the pleasantness of its adjacent adjectives in the SlipNet. (In each case, weights are provided by relevant web frequencies.)

We can use a two-tailed Pearson test ($p < 0.05$) to compare the predictions made in each case to the actual pleasantness scores provided by Whissell's dictionary, and thereby assess the quality of the knowledge used to make the predictions. In the first case, predictions of the pleasantness of stereotype nouns based on the pleasantness of their salient properties (i.e., predicting the pleasantness of Y from the Xs in "*as X as Y*") have a positive

correlation of **0.5** with Whissell; conversely, ironic properties yield a negative correlation of **–0.2**. In the second, predictions of the pleasantness of adjectives based on their relations in the SlipNet (i.e., predicting the pleasantness of X from the Ys in "*as X and Y as*") have a positive correlation of **0.7**. Though pleasantness is just one dimension of lexical affect, it is one that requires a broad knowledge of a word, its usage and its denotations to accurately estimate. In this respect, the *Bard* is well served by a large stock of stereotypes and a coherent network of informative properties.

6 Conclusions

Fishlov (1992) has argued that poetic similes represent a conscious deviation from the norms of non-poetic comparison. His analysis shows that poetic similes are longer and more elaborate, and are more likely to be figurative and to flirt with incongruity. Creative similes do not necessarily use words that are longer, or rarer, or fancier, but use many of the same cultural building blocks as non-creative similes. Armed with a rich vocabulary of building blocks, the *Jigsaw Bard* harvests a great many readymade phrases from the Google ngrams – from the evocative "chocolate martini" to the seemingly incongruous "robot fish" – that can be used to evoke an wide range of properties.

This generativity makes the *Bard* scalable and robust. However, any creativity we may attribute to it comes not from the phrases themselves – they are readymades, after all – but from the recognition of the subtle and often complex properties they evoke. The *Bard* exploits a sweet-spot in our understanding of linguistic creativity, and so, as presented here, is merely a starting point for our continued exploitation of linguistic readymades, rather than an end in itself. By harvesting more complex syntactic structures, and using more sophisticated techniques for analyzing the figurative potential of these phrases, the *Bard* and its ilk may gradually approach the levels of poeticity discussed by Fishlov. For now, it is sufficient that even simple techniques serve as the basis of a robust and practical thesaurus application.

7 Hardware Requirements

The Jigsaw Bard is designed to be a lightweight application that compiles its comprehensive database of readymades in advance. It's run-time demands are low, it has no special hardware requirements, and runs in a standard Web browser.

Acknowledgments

This work was funded in part by Science Foundation Ireland (SFI), via the Centre for Next Generation Localization (CNGL).

References

Margaret Boden, 1994. Creativity: A Framework for Research, Behavioural and Brain Sciences 17(3), 558-568.

Thorsten Brants. and Alex Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium.

Christiane Fellbaum. (ed.) 2008. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge.

David Fishlov. 1992. Poetic and Non-Poetic Simile: Structure, Semantics, Rhetoric. *Poetics Today*, 14(1).

Douglas R Hofstadter. 1995. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, NY.

Rosamund Moon. 2008. Conventionalized as-similes in English: A problem case. *International Journal of Corpus Linguistics* 13(1), 3-37.

Neal Norrick,. 1986. Stock Similes. *Journal of Literary Semantics* XV(1), 39-52.

George Orwell. 1946. Politics And The English Language. *Horizon* **13**(76), 252-265.

Robert J Sternberg. and T. Ivan Lubart, 1995. *Defying the crowd: Cultivating creativity in a culture of conformity*. Free Press, New York.

Archer Taylor. 1954. Proverbial Comparisons and Similes from California. *Folklore Studies* 3. Berkeley: University of California Press.

Michael R. Taylor. (2009). *Marcel Duchamp: Étant donnés* (Philadelphia Museum of Art). Yale University Press.

Tony Veale and Yanfen Hao. 2007. Making Lexical Ontologies Functional and Context-Sensitive. *In Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*.

Cynthia Whissell. 1989. The dictionary of affect in language. In R. Plutchnik & H. Kellerman (eds.) *Emotion: Theory and research*. New York: Harcourt Brace, 113-131.

A Mobile Touchable Application for Online Topic Graph Extraction and Exploration of Web Content

Günter Neumann and Sven Schmeier
Language Technology Lab, DFKI GmbH
Stuhlsatzenhausweg 3, D-66123 Saarbrücken
{neumann|schmeier}@dfki.de

Abstract

We present a mobile touchable application for online topic graph extraction and exploration of web content. The system has been implemented for operation on an iPad. The topic graph is constructed from N web snippets which are determined by a standard search engine. We consider the extraction of a topic graph as a specific empirical collocation extraction task where collocations are extracted between chunks. Our measure of association strength is based on the pointwise mutual information between chunk pairs which explicitly takes their distance into account. An initial user evaluation shows that this system is especially helpful for finding new interesting information on topics about which the user has only a vague idea or even no idea at all.

1 Introduction

Today's Web search is still dominated by a document perspective: a user enters one or more keywords that represent the information of interest and receives a ranked list of documents. This technology has been shown to be very successful when used on an ordinary computer, because it very often delivers concrete documents or web pages that contain the information the user is interested in. The following aspects are important in this context: 1) Users basically have to know what they are looking for. 2) The documents serve as answers to user queries. 3) Each document in the ranked list is considered independently.

If the user only has a vague idea of the information in question or just wants to explore the information space, the current search engine paradigm does not provide enough assistance for these kind of searches. The user has to read through the documents and then eventually reformulate the query in order to find new information. This can be a tedious task especially on mobile devices. Seen in this context, current search engines seem to be best suited for "one-shot search" and do not support content-oriented interaction.

In order to overcome this restricted document perspective, and to provide a mobile device searches to "find out about something", we want to help users with the web content exploration process in two ways:

1. We consider a user query as a specification of a topic that the user wants to know and learn more about. Hence, the search result is basically a graphical structure of the topic and associated topics that are found.

2. The user can interactively explore this topic graph using a simple and intuitive touchable user interface in order to either learn more about the content of a topic or to interactively expand a topic with newly computed related topics.

In the first step, the topic graph is computed on the fly from the a set of web snippets that has been collected by a standard search engine using the initial user query. Rather than considering each snippet in isolation, all snippets are collected into one document from which the topic graph is computed. We consider each topic as an entity, and the edges

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 20–25,
Portland, Oregon, USA, 21 June 2011.

between topics are considered as a kind of (hidden) relationship between the connected topics. The content of a topic are the set of snippets it has been extracted from, and the documents retrievable via the snippets' web links.

A topic graph is then displayed on a mobile device (in our case an iPad) as a touch-sensitive graph. By just touching on a node, the user can either inspect the content of a topic (i.e, the snippets or web pages) or activate the expansion of the graph through an on the fly computation of new related topics for the selected node.

In a second step, we provide additional background knowledge on the topic which consists of explicit relationships that are generated from an online Encyclopedia (in our case Wikipedia). The relevant background relation graph is also represented as a touchable graph in the same way as a topic graph. The major difference is that the edges are actually labeled with the specific relation that exists between the nodes.

In this way the user can explore in an uniform way both new information nuggets and validated background information nuggets interactively. Fig. 1 summarizes the main components and the information flow.

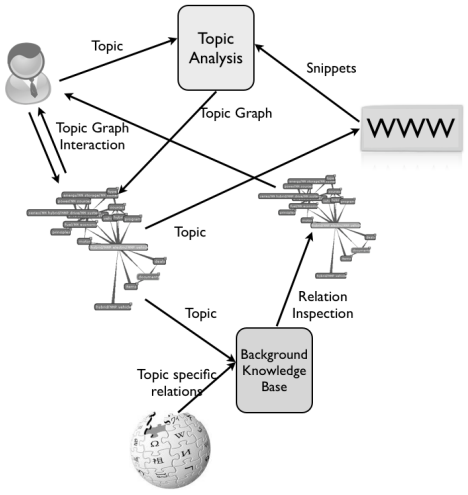


Figure 1: Blueprint of the proposed system.

2 Touchable User Interface: Examples

The following screenshots show some results for the search query "Justin Bieber" running on the current iPad demo–app. At the bottom of the iPad screen, the user can select whether to perform text exploration from the Web (via button labeled "i–GNSSMM") or via Wikipedia (touching button "i–MILREX"). The Figures 2, 3, 4, 5 show results for the "i–GNSSMM" mode, and Fig. 6 for the "i-MILREX" mode. General settings of the iPad demo-app can easily be changed. Current settings allow e.g., language selection (so far, English and German are supported) or selection of the maximum number of snippets to be retrieved for each query. The other parameters mainly affect the display structure of the topic graph.

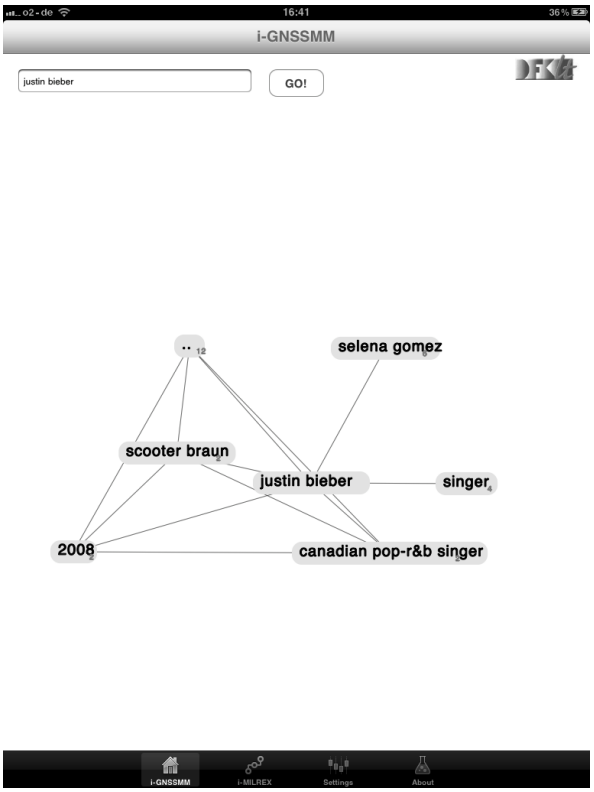


Figure 2: The topic graph computed from the snippets for the query "Justin Bieber". The user can double touch on a node to display the associated snippets and web pages. Since a topic graph can be very large, not all nodes are displayed. Nodes, which can be expanded are marked by the number of hidden immediate nodes. A single touch on such a node expands it, as shown in Fig. 3. A single touch on a node that cannot be expanded adds its label to the initial user query and triggers a new search with that expanded query.

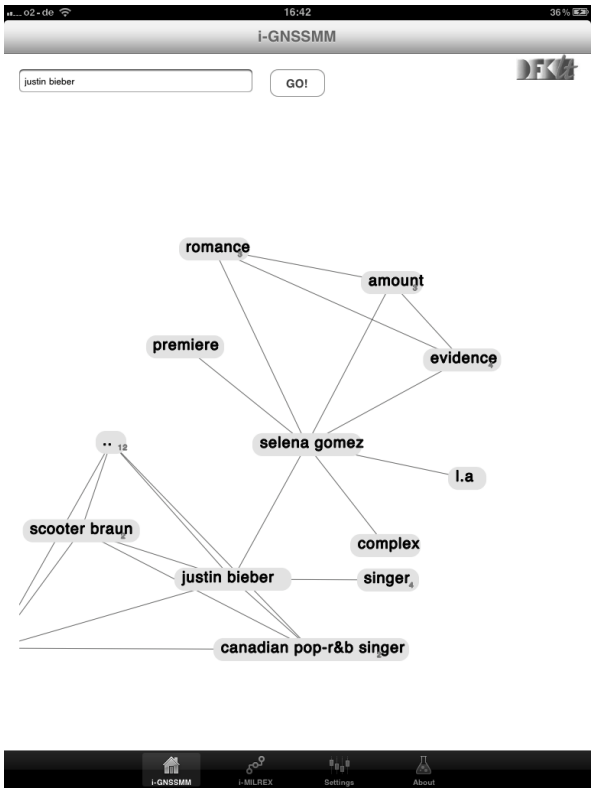


Figure 3: The topic graph from Fig. 2 has been expanded by a single touch on the node labeled "selena gomez". Double touching on that node triggers the display of associated web snippets (Fig. 4) and the web pages (Fig. 5).

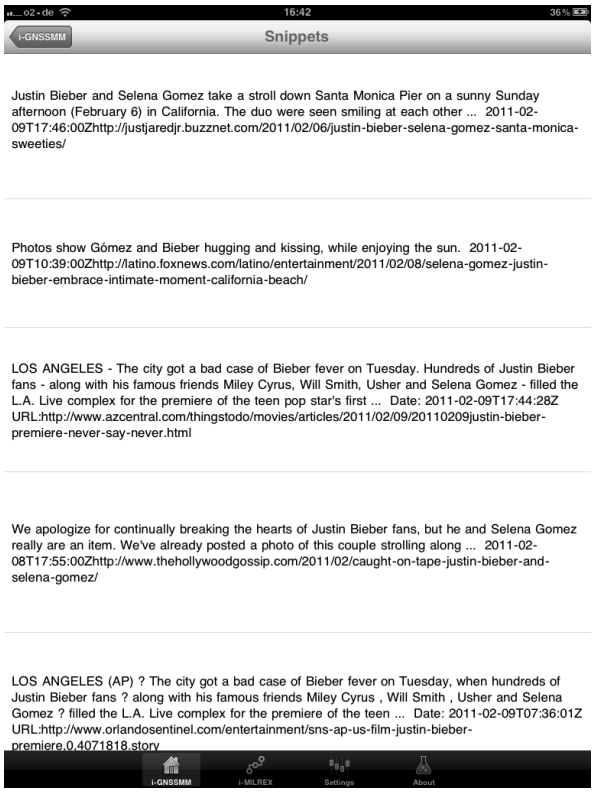


Figure 4: The snippets that are associated with the node label "selena gomez" of the topic graph from Fig. 3.In order to go back to the topic graph, the user simply touches the button labeled i-GNSSMM on the left upper corner of the iPad screen.

3 Topic Graph Extraction

We consider the extraction of a topic graph as a specific *empirical collocation extraction task*. However, instead of extracting collations between words, which is still the dominating approach in collocation extraction research, e.g., (Baroni and Evert, 2008), we are extracting collocations between chunks, i.e., word sequences. Furthermore, our measure of association strength takes into account the distance between chunks and combines it with the PMI (pointwise mutual information) approach (Turney, 2001).

The core idea is to compute a set of chunk–pair–distance elements for the N first web snippets returned by a search engine for the topic Q, and to compute the topic graph from these elements.[1] In general for two chunks, a single chunk–pair–distance element stores the distance between the chunks by counting the number of chunks in–between them. We distinguish elements which have the same words in the same order, but have different distances. For example, (Peter, Mary, 3) is different from (Peter, Mary, 5) and (Mary, Peter, 3).

We begin by creating a document S from the N-first web snippets so that each line of S contains a complete snippet. Each textline of S is then tagged with Part–of–Speech using the SVM-Tagger (Giménez and Màrquez, 2004) and chunked in the next step. The chunker recognizes two types of word chains. Each chain consists of longest matching sequences of words with the same PoS class, namely noun chains or verb chains, where an element of a noun chain belongs to one of the extended noun tags[2], and elements of a verb

[1] For the remainder of the paper N=1000. We are using Bing (`http://www.bing.com/`) for Web search.

[2] Concerning the English PoS tags, "word/PoS" expressions that match the following regular expression are considered as extended noun tag: "/(N(N|P))|/VB(N|G)|/IN|/DT". The En-

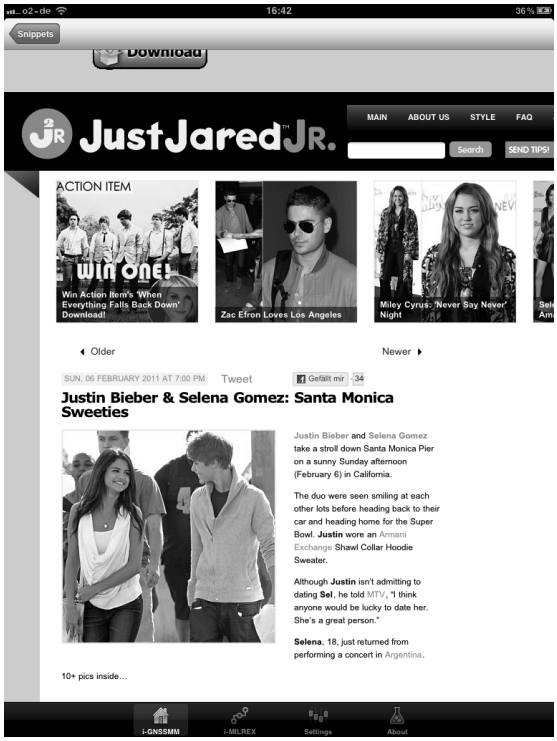


Figure 5: The web page associated with the first snippet of Fig. 4. A single touch on that snippet triggers a call to the iPad browser in order to display the corresponding web page. The left upper corner button labeled "Snippets" has to be touched in order to go back to the snippets page.

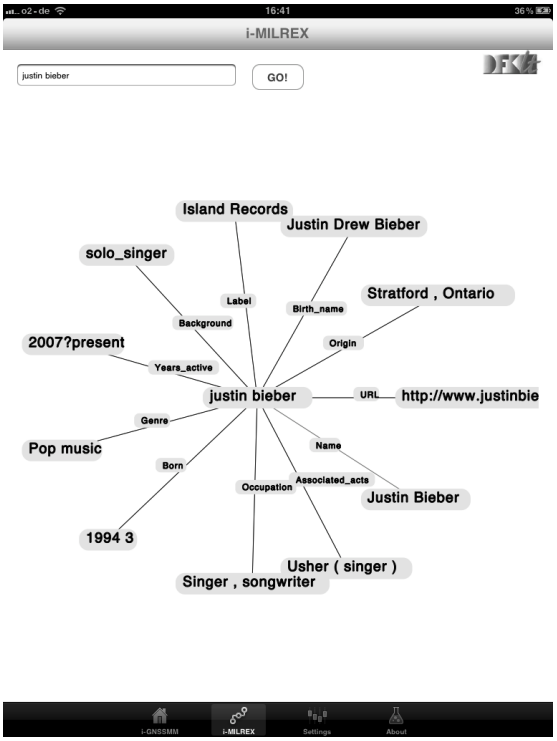


Figure 6: If mode "i–MILREX" is chosen then text exploration is performed based on relations computed from the info–boxes extracted from Wikipedia. The central node corresponds to the query. The outer nodes represent the arguments and the inner nodes the predicate of a info–box relation. The center of the graph corresponds to the search query.

chain only contains verb tags. We finally apply a kind of "phrasal head test" on each identified chunk to guarantee that the right–most element only belongs to a proper noun or verb tag. For example, the chunk "a/DT british/NNP formula/NNP one/NN racing/VBG driver/NN from/IN scotland/NNP" would be accepted as proper NP chunk, where "compelling/VBG power/NN of/IN" is not.

Performing this sort of shallow chunking is based on the assumptions: 1) noun groups can represent the arguments of a relation, a verb group the relation itself, and 2) web snippet chunking needs highly robust NL technologies. In general, chunking crucially depends on the quality of the embedded PoS–tagger. However, it is known that PoS–tagging performance of even the best taggers decreases substantially when applied on web pages (Giesbrecht and Evert, 2009). Web snippets are even harder to process because they are not necessary contiguous pieces of texts, and usually are not syntactically well-formed paragraphs due to some intentionally introduced breaks (e.g., denoted by ... betweens text fragments). On the other hand, we want to benefit from PoS tagging during chunk recognition in order to be able to identify, on the fly, a shallow phrase structure in web snippets with minimal efforts.

The chunk–pair–distance model is computed from the list of chunks. This is done by traversing the chunks from left to right. For each chunk c_i, a set is computed by considering all remaining chunks and their distance to c_i, i.e., $(c_i, c_{i+1}, dist_{i(i+1)})$, $(c_i, c_{i+2}, dist_{i(i+2)})$, etc. We do this for each chunk list computed for each web snippet. The distance $dist_{ij}$ of two chunks c_i and c_j is computed directly from the chunk list, i.e., we do not count the position

glish Verbs are those whose PoS tag start with VB. We are using the tag sets from the Penn treebank (English) and the Negra treebank (German).

of ignored words lying between two chunks.

The motivation for using chunk–pair–distance statistics is the assumption that the strength of hidden relationships between chunks can be covered by means of their collocation degree and the frequency of their relative positions in sentences extracted from web snippets; cf. (Figueroa and Neumann, 2006) who demonstrated the effectiveness of this hypothesis for web–based question answering.

Finally, we compute the frequencies of each chunk, each chunk pair, and each chunk pair distance. The set of all these frequencies establishes the chunk–pair–distance model CPD_M. It is used for constructing the topic graph in the final step. Formally, a topic graph $TG = (V, E, A)$ consists of a set V of nodes, a set E of edges, and a set A of node actions. Each node $v \in V$ represents a chunk and is labeled with the corresponding PoS–tagged word group. Node actions are used to trigger additional processing, e.g., displaying the snippets, expanding the graph etc.

The nodes and edges are computed from the chunk–pair–distance elements. Since, the number of these elements is quite large (up to several thousands), the elements are ranked according to a weighting scheme which takes into account the frequency information of the chunks and their collocations. More precisely, the weight of a chunk–pair–distance element $cpd = (c_i, c_j, D_{ij})$, with $D_{i,j} = \{(freq_1, dist_1), (freq_2, dist_2), ..., (freq_n, dist_n)\}$, is computed based on PMI as follows:

$$PMI(cpd) = log_2((p(c_i, c_j)/(p(c_i) * p(c_j)))$$
$$= log_2(p(c_i, c_j)) - log_2(p(c_i) * p(c_j))$$

where relative frequency is used for approximating the probabilities $p(c_i)$ and $p(c_j)$. For $log_2(p(c_i, c_j))$ we took the (unsigned) polynomials of the corresponding Taylor series[3] using $(freq_k, dist_k)$ in the k-th Taylor polynomial and adding them up:

$$PMI(cpd) = (\sum_{k=1}^{n} \frac{(x_k)^k}{k}) - log_2(p(c_i) * p(c_j))$$

$$, where\ x_k = \frac{freq_k}{\sum_{k=1}^{n} freq_k}$$

[3]In fact we used the polynomials of the Taylor series for $ln(1 + x)$. Note also that k is actually restricted by the number of chunks in a snippet.

The visualized topic graph TG is then computed from a subset $CPD'_M \subset CPD_M$ using the m highest ranked cpd for fixed c_i. In other words, we restrict the complexity of a TG by restricting the number of edges connected to a node.

4 Wikipedia's Infoboxes

In order to provide query specific background knowledge we make use of Wikipedia's infoboxes. These infoboxes contain facts and important relationships related to articles. We also tested DBpedia as a background source (Bizer et al., 2009). However, it turned out that currently it contains too much and redundant information. For example, the Wikipedia infobox for Justin Bieber contains eleven basic relations whereas DBpedia has fifty relations containing lots of redundancies. In our current prototype, we followed a straightforward approach for extracting infobox relations: We downloaded a snapshot of the whole English Wikipedia database (images excluded), extracted the infoboxes for all articles if available and built a Lucene Index running on our server. We ended up with 1.124.076 infoboxes representing more than 2 million different searchable titles. The average access time is about 0.5 seconds. Currently, we only support exact matches between the user's query and an infobox title in order to avoid ambiguities. We plan to extend our user interface so that the user may choose different options. Furthermore we need to find techniques to cope with undesired or redundant information (see above). This extension is not only needed for partial matches but also when opening the system to other knowledgesources like DBpedia, newsticker, stock information and more.

5 Evaluation

For an initial evaluation we had 20 testers: 7 came from our lab and 13 from non–computer science related fields. 15 persons had never used an iPad before. After a brief introduction to our system (and the iPad), the testers were asked to perform three different searches (using Google, i–GNSSMM and i–MILREX) by choosing the queries from a set of ten themes. The queries covered definition questions like *EEUU* and *NLF*, questions about persons like *Justin Bieber*, *David Beckham*, *Pete Best*, *Clark*

Kent, and *Wendy Carlos* , and general themes like *Brisbane*, *Balancity*, and *Adidas*. The task was not only to get answers on questions like "Who is . . . " or "What is . . . " but also to acquire knowledge about background facts, news, rumors (gossip) and more interesting facts that come into mind during the search. Half of the testers were asked to first use Google and then our system in order to compare the results and the usage on the mobile device. We hoped to get feedback concerning the usability of our approach compared to the well known internet search paradigm. The second half of the participants used only our system. Here our research focus was to get information on user satisfaction of the search results. After each task, both testers had to rate several statements on a Likert scale and a general questionnaire had to be filled out after completing the entire test. Table 1 and 2 show the overall result.

Table 1: Google

#Question	v.good	good	avg.	poor
results first sight	55%	40%	15%	-
query answered	71%	29%	-	-
interesting facts	33%	33%	33%	-
suprising facts	33%	-	-	66%
overall feeling	33%	50%	17%	4%

Table 2: i-GNSSMM

#Question	v.good	good	avg.	poor
results first sight	43%	38%	20%	-
query answered	65%	20%	15%	-
interesting facts	62%	24%	10%	4%
suprising facts	66%	15%	13%	6%
overall feeling	54%	28%	14%	4%

The results show that people in general prefer the result representation and accuracy in the Google style. Especially for the general themes the presentation of web snippets is more convenient and more easy to understand. However when it comes to interesting and suprising facts users enjoyed exploring the results using the topic graph. The overall feeling was in favor of our system which might also be due to the fact that it is new and somewhat more playful.

The replies to the final questions: *How successful were you from your point of view? What did you like most/least? What could be improved?* were informative and contained positive feedback. Users felt they had been successful using the system. They liked the paradigm of the explorative search on the iPad and preferred touching the graph instead of reformulating their queries. The presentation of background facts in i–MILREX was highly appreciated. However some users complained that the topic graph became confusing after expanding more than three nodes. As a result, in future versions of our system, we will automatically collapse nodes with higher distances from the node in focus. Although all of our test persons make use of standard search engines, most of them can imagine to using our system at least in combination with a search engine even on their own personal computers.

6 Acknowledgments

The presented work was partially supported by grants from the German Federal Ministry of Economics and Technology (BMWi) to the DFKI Theseus projects (FKZ: 01MQ07016) TechWatch–Ordo and Alexandria4Media.

References

Marco Baroni and Stefan Evert. 2008. *Statistical methods for corpus exploitation*. In A. Lüdeling and M. Kytö (eds.), Corpus Linguistics. An International Handbook, Mouton de Gruyter, Berlin.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Soren Auer, Christian Becker, Richard Cyganiak, 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): 154165.

Alejandro Figueroa and Günter Neumann. 2006. *Language Independent Answer Prediction from the Web*. In proceedings of the 5th FinTAL, Finland.

Eugenie Giesbrecht and Stefan Evert. 2009. *Part-of-speech tagging - a solved task? An evaluation of PoS taggers for the Web as corpus*. In proceedings of the 5th Web as Corpus Workshop, San Sebastian, Spain.

Jesús Giménez and Lluís Màrquez. 2004. *SVMTool: A general PoS tagger generator based on Support Vector Machines*. In proceedings of LREC'04, Lisbon, Portugal.

Peter Turney. 2001. *Mining the web for synonyms: PMI-IR versus LSA on TOEFL.* In proceedings of the 12th ECML, Freiburg, Germany.

EdIt: A Broad-Coverage Grammar Checker Using Pattern Grammar

Chung-Chi Huang Mei-Hua Chen

Institute of Information Systems and Applications, National Tsing Hua University, HsinChu, Taiwan, R.O.C. 300

Shih-Ting Huang Jason S. Chang

Department of Computer Science, National Tsing Hua University, HsinChu, Taiwan, R.O.C. 300

{u901571,chen.meihua,koromiko1104,Jason.jschang}@gmail.com

Abstract

We introduce a new method for learning to detect grammatical errors in learner's writing and provide suggestions. The method involves parsing a reference corpus and inferring grammar patterns in the form of a sequence of content words, function words, and parts-of-speech (e.g., "*play ~ role in* Ving" and "*look forward to* Ving"). At runtime, the given passage submitted by the learner is matched using an extended Levenshtein algorithm against the set of pattern rules in order to detect errors and provide suggestions. We present a prototype implementation of the proposed method, EdIt, that can handle a broad range of errors. Promising results are illustrated with three common types of errors in non-native writing.

1 Introduction

Recently, an increasing number of research has targeted language learners' need in editorial assistance including detecting and correcting grammar and usage errors in texts written in a second language. For example, Microsoft Research has developed the ESL Assistant, which provides such a service to ESL and EFL learners.

Much of the research in this area depends on hand-crafted rules and focuses on certain error types. Very little research provides a general framework for detecting and correcting all types of errors. However, in the sentences of ESL writing, there may be more than one errors and one error may affect the performance of handling other errors. Erroneous sentences could be more efficiently identified and corrected if a grammar checker handles all errors at once, using a set of pattern rules that reflect the predominant usage of the English language.

Consider the sentences, "*He play an important roles to close this deals.*" and "*He looks forward to hear you.*" The first sentence contains inaccurate word forms (i.e., *play, roles,* and *deals*), and rare usage (i.e., "*role to close*"), while the second sentence use the incorrect verb form of "*hear*". Good responses to these writing errors might be (a) Use "*played*" instead of "play." (b) Use "*role*" instead of "*roles*", (c) Use "*in closing*" instead of "*to close*" (d) Use "*to hearing*" instead of "*to hear*", and (e) insert "*from*" between "*hear*" and "*you*." These suggestions can be offered by learning the patterns rules related to "**play ~ role**" and "**look forward**" based on analysis of ngrams and collocations in a very large-scale reference corpus. With corpus statistics, we could learn the needed phraseological tendency in the form of pattern rules such as "**play ~ role in V-ing**) and "**look forward to V-ing**." The use of such pattern rules is in line with the recent theory of Pattern Grammar put forward by Hunston and Francis (2000).

We present a system, EdIt, that automatically learns to provide suggestions for rare/wrong usages in non-native writing. Example EdIt responses to a

text are shown in Figure 1. EdIt has retrieved the related pattern grammar of some ngram and collocation sequences given the input (e.g., "*play ~ role in* V-ing[1]", and "*look forward to* V-ing"). EdIt learns these patterns during pattern extraction process by syntactically analyzing a collection of well-formed, published texts.

At run-time, EdIt first processes the input passages in the article (e.g., "*He play an important roles to close* ") submitted by the L2 learner. And EdIt tag the passage with part of speech information, and compares the tagged sentence against the pattern rules anchored at certain collocations (e.g., "*play ~ role*" and "*look forward*"). Finally, EdIt finds the minimum-edit-cost patterns matching the passages using an extended Levenshtein's algorithm (Levenshtein, 1966). The system then highlights the edits and displays the pattern rules as suggestions for correction. In our prototype, EdIt returns the preferred word form and preposition usages to the user directly (see Figure 1); alternatively, the actual surface words (e.g., "closing" and "deal") could be provided.

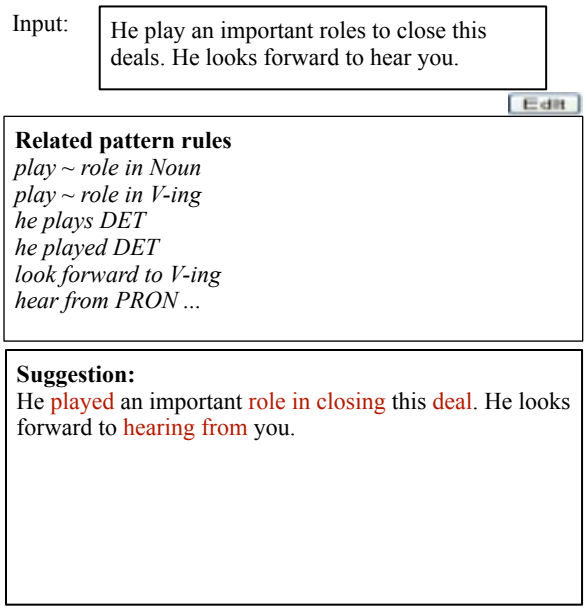


Figure 1. Example responses to the non-native writing.

2 Related Work

Grammar checking has been an area of active research. Many methods, rule-oriented or data-driven, have been proposed to tackle the problem of detecting and correcting incorrect grammatical and usage errors in learner texts. It is at times no easy to distinguish these errors. But Fraser and Hodson (1978) shows the distinction between these two kinds of errors.

For some specific error types (e.g., article and preposition error), a number of interesting rule-based systems have been proposed. For example, Uria et al. (2009) and Lee et al. (2009) leverage heuristic rules for detecting Basque determiner and Korean particle errors, respectively. Gamon et al. (2009) bases some of the modules in ESL Assistant on rules derived from manually inspecting learner data. Our pattern rules, however, are automatically derived from readily available well-formed data, but nevertheless very helpful for correcting errors in non-native writing.

More recently, statistical approaches to developing grammar checkers have prevailed. Among unsupervised checkers, Chodorow and Leacock (2000) exploits negative evidence from edited textual corpora achieving high precision but low recall, while Tsao and Wible (2009) uses general corpus only. Additionally, Hermet et al. (2008) and Gamon and Leacock (2010) both use Web as a corpus to detect errors in non-native writing. On the other hand, supervised models, typically treating error detection/correction as a classification problem, may train on well-formed texts as in the methods by De Felice and Pulman (2008) and Tetreault et al. (2010), or with additional learner texts as in the method proposed by Brockett et al. (2006). Sun et al. (2007) describes a method for constructing a supervised detection system trained on raw well-formed and learner texts without error annotation.

Recent work has been done on incorporating word class information into grammar checkers. For example, Chodorow and Leacock (2000) exploit bigrams and trigrams of function words and part-of-speech (PoS) tags, while Sun et al. (2007) use labeled sequential patterns of function, time expression, and part-of-speech tags. In an approach similar to our work, Tsao and Wible (2009) use a combined ngrams of words forms, lemmas, and part-of-speech tags for research into constructional phenomena. The main differences are that we anchored each pattern rule in lexical collocation so as to avoid deriving rules that is may have two

[1] In the pattern rules, we translate the part-of-speech tag to labels that are commonly used in learner dictionaries. For instance, we use *V-ing* for the tag VBG denoting the progressive verb form, and *Pron* and *Pron$* denotes a pronoun and a possessive pronoun respectively.

consecutive part-of-speech tags (e.g, "*V Pron$ socks off"*). The pattern rules we have derived are more specific and can be effectively used in detecting and correcting errors.

In contrast to the previous research, we introduce a broad-coverage grammar checker that accommodates edits such as substitution, insertion and deletion, as well as replacing word forms or prepositions using pattern rules automatically derived from very large-scale corpora of well-formed texts.

3 The EdIt System

Using supervised training on a learner corpus is not very feasible due to the limited availability of large-scale annotated non-native writing. Existing systems trained on learner data tend to offer high precision but low recall. Broad coverage grammar checkers may be developed using readily available large-scale corpora. To detect and correct errors in non-native writing, a promising approach is to automatically extract lexico-syntactical pattern rules that are expected to distinguish correct and in correct sentences.

3.1 Problem Statement

We focus on correcting grammatical and usage errors by exploiting pattern rules of specific collocation (elastic or rigid such as "*play ~ rule*" or "*look forward*"). For simplification, we assume that there is no spelling errors. EdIt provides suggestions to common writing errors[2] of the following correlated with essay scores[3].

(1) wrong word form
  (A) singular determiner preceding plural noun
  (B) wrong verb form: concerning modal verbs (e.g., "would said"), subject-verb agreement, auxiliary (e.g., "should have tell the truth"), gerund and infinitive usage (e.g., "look forward to see you" and "in an attempt to helping you")

(2) wrong preposition (or infinitive-to)
  (A) wrong preposition (e.g., "to depends of it")
  (B) wrong preposition and verb form (e.g., "to play an important role to close this deal")

(3) transitivity errors
  (A) transitive verb (e.g., "to discuss about the matter" and "to affect to his decision")
  (B) intransitive verb (e.g., "to listens the music")

The system is designed to find pattern rules related to the errors and return suggestionst. We now formally state the problem that we are addressing.

Problem Statement: We are given a reference corpus *C* and a non-native passage *T*. Our goal is to detect grammatical and usage errors in *T* and provide suggestions for correction. For this, we *extract* a set of *pattern rules*, $u_1,\ldots, u_m$ from *C* such that the rules reflect the predominant usage and are likely to distinguish most errors in non-native writing.

In the rest of this section, we describe our solution to this problem. First, we define a strategy for identifying predominant phraseology of frequent ngrams and collocations in Section 3.2. Afer that, we show how EdIt proposes grammar correctionsedits to non-native writing at run-time in Section 3.3.

3.2 Deriving Pattern Rules

We attempt to derive patterns (e.g., "*play ~ role in V-ing*") from *C* expected to represent the immediate context of collocations (e.g., "*play ~ role*" or "*look forward*"). Our derivation process consists of the following four-stage:

Stage 1. Lemmatizing, POS Tagging and Phrase chunking. In the first stage, we lemmatize and tag sentences in *C*. Lemmatization and POS tagging both help to produce more general pattern rules from ngrams or collocations. The based phrases are used to extract collocations.

Stage 2. Ngrams and Collocations. In the second stage of the training process, we calculate ngrams and collocations in *C*, and pass the frequent ngrams and collocations to Stage 4.

We employ a number of steps to acquire statistically significant collocations--determining the pair of head words in adjacent base phrases, calculating their pair-wise mutual information values, and filtering out candidates with low MI values.

Stage 3. onstructing Inverted Files. In the third stage in the training procedure, we build up inverted files for the lemmas in *C* for quick access in Stage 4. For each word lemma we store surface words, POS tags, pointers to sentences with base phrases marked.

[2] See (Nicholls, 1999) for common errors.

[3] See (Leacock and Chodorow, 2003) and (Burstein et al., 2004) for correlation.

```
procedure GrammarChecking(T,PatternGrammarBank)
(1) Suggestions=""//candidate suggestions
(2) sentences=sentenceSplitting(T)
    for each sentence in sentences
(3)  userProposedUsages=extractUsage(sentence)
     for each userUsage in userProposedUsages
(4)   patGram=findPatternGrammar(userUsage.lexemes,
                          PatternGrammarBank)
(5)   minEditedCost=SystemMax; minEditedSug=""
      for each pattern in patGram
(6)     cost=extendedLevenshtein(userUsage,pattern)
        if cost<minEditedCost
(7)       minEditedCost=cost; minEditedSug=pattern
      if minEditedCost>0
(8)     append (userUsage,minEditedSug) to Suggestions
(9) Return Suggestions
```

Figure 2. Grammar suggestion/correction at run-time

```
procedure extendedLevenshtein(userUsage,pattern)
(1) allocate and initialize costArray
    for i in range(len(userUsage))
      for j in range(len(pattern))
        if equal(userUsage[i],pattern[j]) //substitution
(2a)     substiCost=costArray[i-1,j-1]+0
        elseif sameWordGroup(userUsage[i],pattern[j])
(2b)     substiCost=costArray[i-1,j-1]+0.5
(2c)    else substiCost=costArray[i-1,j-1]+1
        if equal(userUsage[i+1],pattern[j+1]) //deletion
(3a)     delCost=costArray[i-1,j]+smallCost
(3b)    else delCost=costArray[i-1,j]+1
        if equal(userUsage[i+1],pattern[j+1]) //insertion
(4a)     insCost=costArray[i,j-1]+smallCost
(4b)    else insCost=costArray[i,j-1]+1
(5)     costArray[i,j]=min(substiCost,delCost,insCost)
(6) Return costArray[len(userUsage),len(pattern)]
```

Figure 3. Algorithm for identifying errors

Stage 4. Deriving pattern rules. In the fourth and final stage, we use the method described in a previous work (Chen et al., 2011) and use the inverted files to find all sentences containing a give word and collocation. Words surrounding a collocation are identified and generalized based on their corresponding POS tags. These sentences are then transformed into a set of n-gram of words and POS tags, which are subsequently counted and ranked to produce pattern rules with high frequencies.

3.3 Run-Time Error Correction

Once the patterns rules are derived from a corpus of well-formed texts, EdIt utilizes them to check grammaticality and provide suggestions for a given text via the procedure in Figure 2.

In Step (1) of the procedure, we initiate a set *Suggestions* to collect grammar suggestions to the user text *T* according to the bank of pattern grammar *PatternGrammarBank*. Since EdIt system focuses on grammar checking at sentence level, *T* is heuristically split (Step (2)).

For each *sentence*, we extract ngram and POS tag sequences *userUsage* in *T*. For the example of "He play an important roles. He looks forword to hear you", we extract ngram such as *he V DET, play an JJ NNS, play ~ roles to V, this NNS, look forward to VB,* and *hear Pron.*

For each *userUsage*, we first access the pattern rules related to the word and collocation within (e.g., play-role patterns for "*play ~ role to close*") Step (4). And then we compare *userUsage* against these rules (from Step (5) to (7)). We use the extended Levenshtein's algorithm shown in Figure 3 to compare *userUsage* and pattern rules.

If only partial matches are found for *userUsage*, that could mean we have found a potential errors. We use *minEditedCost* and *minEditedSug* to contrain the patterns rules found for error suggestions (Step (5)). In the following, we describe how to find minimal-distance edits.

In Step (1) of the algorithm in Figure 3 we allocate and initialize *costArray* to gather the dynamic programming based cost to transform *userUsage* into a specific contextual rule *pattern*. Afterwards, the algorithm defines the cost of performing substitution (Step (2)), deletion (Step (3)) and insertion (Step (4)) at i-indexed *userUsage* and j-indexed *pattern*. If the entries *userUsage*[*i*] and *pattern*[*j*] are equal literally (e.g., "VB" and "VB") or grammatically (e.g., "DT" and "Pron$"), no edit is needed, hence, no cost (Step (2a)). On the other hand, since learners tend to select wrong word form and preposition, we set a lower cost for substitution among different word forms of the same lemma or lemmas with the same POS tag (e.g., *replacing V with V-ing* or *replacing to with in*". In addition to the conventional deletion and insertion (Step (3b) and (4b) respectively), we look ahead to the elements *userUsage*[*i*+1] and *pattern*[*j*+1] considering the fact that "with or without preposition" and "transitive or intransitive verb" often puzzles EFL learners (Step (3a) and (4a)). Only a small edit cost is counted if the next elements in *userUsage* and *Pattern* are "equal". In Step (6) the extended Levenshtein's algorithm returns the minimum edit cost of revising *userUsage* using *pattern*.

Once we obtain the costs to transform the *userUsage* into a similar, frequent pattern rules, we propose the minimum-cost rules as suggestions for

correction (e.g., "*play ~ role in V-ing*" for revising "*play ~ role to V*") (Step (8) in Figure 2), if its minimum edit cost is greater than zero. Otherwise, the usage is considered valid. Finally, the *Suggestions* accumulated for *T* are returned to users (Step (9)). Example input and editorial suggestions returned to the user are shown in Figure 1. Note that pattern rules involved flexible collocations are designed to take care of long distance dependencies that might be always possible to cover with limited ngram (for *n* less than 6). In addition, the long patter rules can be useful even when it is not clear whether there is an error when looking at a very narrow context. For example, "hear" can be either be transitive or intransitive depending on context. In the context of "look forward to" and person noun object, it is should be intransitive and require the preposition "from" as suggested in the results provided by EdIt (see Figure 1).

In existing grammar checkers, there are typically many modules examining different types of errors and different module may have different priority and conflict with one another. Let us note that this general framework for error detection and correction is an original contribution of our work. In addition, we incorporate probabilities conditioned on word positions in order to weigh edit costs. For example, the conditional probability of *V* to immediately follow "look forward to" is virtually 0, while the probability of *V-ing* to do so is approximates 0.3. Those probabilistic values are used to weigh different edits.

4 Experimental Results

In this section, we first present the experimental setting in EdIt (Section 4.1). Since our goal is to provide to learners a means to efficient broad-coverage grammar checking, EdIt is web-based and the acquisition of the pattern grammar in use is offline. Then, we illustrate three common types of errors, scores correlated, EdIt[4] capable of handling.

4.1 Experimental Setting

We used British National Corpus (BNC) as our underlying general corpus *C*. It is a 100 million British English word collection from a wide range of sources. We exploited GENIA tagger to obtain the lemmas, PoS tags and shallow parsing results of *C*'s sentences, which were all used in constructing inverted files and used as examples for GRASP to infer lexicalized pattern grammar.

Inspired by (Chen et al., 2011) indicating EFL learners tend to choose incorrect prepositions and following word forms following a VN collocation, and (Gamon and Leacock, 2010) showing fixed-length and fixed-window lexical items are the best evidence for correction, we equipped EdIt with pattern grammar rules consisting of fixed-length (from one- to five-gram) lexical sequences or VN collocations and their fixed-window usages (e.g., "IN(*in*) VBG" after "play ~ role", for window 2).

4.2 Results

We examined three types of errors and the mixture of them for our correction system (see Table 1). In this table, results of ESL Assistant are shown for comparison, and grammatical suggestions are underscored. As suggested, lexical and PoS information in learner texts is useful for a grammar checker, pattern grammar EdIt uses is easily accessible and effective in both grammaticality and usage check, and a weighted extension to Levenshtein's algorithm in EdIt accommodates substitution, deletion and insertion edits to learners' frequent mistakes in writing.

5 Future Work and Summary

Many avenues exist for future research and improvement. For example, we could augment pattern grammar with lexemes' PoS information in that the contexts of a word of different PoS tags vary. Take *discuss* for instance. The present tense verb *discuss* is often followed by determiners and nouns while the passive is by the preposition *in* as in "… is discussed in Chapter one." Additionally, an interesting direction to explore is enriching pattern grammar with semantic role labels (Chen et al., 2011) for simple semantic check.

In summary, we have introduced a method for correcting errors in learner text based on its lexical and PoS evidence. We have implemented the method and shown that the pattern grammar and extended Levenshtein algorithm in this method are promising in grammar checking. Concerning EdIt's broad coverage over different error types, simplicity in design, and short response time, we plan to evaluate it more fully: with or without conditional probability using majority voting or not.

[4] At http://140.114.214.80/theSite/EdIt_demo2/

Erroneous sentence	EdIt suggestion	ESL Assistant suggestion
Incorrect word form		
… a sunny days …	a sunny N	a sunny day
every days, I …	every N	every day
I would said to …	would V	would say
he play a …	he V-ed	none
… should have tell the truth	should have V-en	should have to tell
… look forward to see you	look forward to V-ing	none
… in an attempt to seeing you	an attempt to V	none
… be able to solved this problem	able to V	none
Incorrect preposition		
he plays an important role to close …	play ~ role in	none
he has a vital effect at her.	have ~ effect on	effect on her
it has an effect on reducing …	have ~ effect of V-ing	none
… depend of the scholarship	depend on	depend on
Confusion between intransitive and transitive verb		
he listens the music.	missing "to" after "listens"	missing "to" after "listens"
it affects to his decision.	unnecessary "to"	unnecessary "to"
I understand about the situation.	unnecessary "about"	unnecessary "about"
we would like to discuss about this matter.	unnecessary "about"	unnecessary "about"
Mixed		
she play an important roles to close this deals.	she V-ed; an Adj N; play ~ role in V-ing; this N	play an important role; close this deal
I look forward to hear you.	look forward to V-ing; missing "from" after "hear"	none

Table 1. Three common score-related error types and their examples with suggestions from EdIt and ESL Assistant.

References

C. Brockett, W. Dolan, and M. Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the ACL*.

J. Burstein, M. Chodorow, and C. Leacock. 2004. Automated essay evaluation: the *criterion online writing service*. *AI Magazine*, 25(3):27-36.

M. H. Chen, C. C. Huang, S. T. Huang, H. C. Liou, and J. S. Chang. 2011. A cross-lingual pattern retrieval framework. In *Proceedings of the CICLing*.

M. Chodorow and C. Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of the NAACL*, pages 140-147.

R. De Felice and S. Pulman. 2008. A classifer-based approach to preposition and determiner error correction in L2 English. In *COLING*.

I. S. Fraser and L. M. Hodson. 1978. Twenty-one kicks at the grammar horse. *English Journal*.

M. Gamon, C. Leacock, C. Brockett, W. B. Bolan, J. F. Gao, D. Belenko, and A. Klementiev. Using statistical techniques and web search to correct ESL errors. *CALICO*, 26(3): 491-511.

M. Gamon and C. Leacock. 2010. Search right and thou shalt find … using web queries for learner error detection. In *Proceedings of the NAACL*.

M. Hermet, A. Desilets, S. Szpakowicz. 2008. Using the web as a linguistic resource to automatically correct lexico-syntatic errors. In *LREC*, pages 874-878.

S. Hunston and G. Francis. 2000. Pattern grammar: a corpus-driven approach to the lexical grammar of English.

C. M. Lee, S. J. Eom, and M. Dickinson. 2009. Toward analyzing Korean learner particles. In *CALICO*.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707-710.

C. Leacock and M. Chodorow. 2003. Automated grammatical error detection.

D. Nicholls. 1999. The Cambridge Learner Corpus – error coding and analysis for writing dictionaries and other books for English Learners.

G. H. Sun, X. H. Liu, G. Cong, M. Zhou, Z. Y. Xiong, J. Lee, and C. Y. Lin. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *ACL*.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for prepositions selection and error detection. In *Proceedings of the ACL*, pages 353-358.

N. L. Tsao and D. Wible. 2009. A method for unsupervised broad-coverage lexical error detection and correction. In *NAACL Workshop*, pages 51-54.

MemeTube: A Sentiment-based Audiovisual System for Analyzing and Displaying Microblog Messages

Cheng-Te Li[1] **Chien-Yuan Wang**[2] **Chien-Lin Tseng**[2] **Shou-De Lin**[1,2]
[1] Graduate Institute of Networking and Multimedia
[2] Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
{d98944005, sdlin}@csie.ntu.edu.tw {gagedark, moonspirit.wcy}@gmail.com

Abstract

Micro-blogging services provide platforms for users to share their feelings and ideas on the move. In this paper, we present a search-based demonstration system, called MemeTube, to summarize the sentiments of microblog messages in an audiovisual manner. MemeTube provides three main functions: (1) recognizing the sentiments of messages (2) generating music melody automatically based on detected sentiments, and (3) produce an animation of real-time piano playing for audiovisual display. Our MemeTube system can be accessed via: http://mslab.csie.ntu.edu.tw/memetube/ .

1 Introduction

Micro-blogging services such as Twitter[1], Plurk[2], and Jaiku[3], are platforms that allow users to share immediate but short messages with friends. Generally, the micro-blogging services possess some signature properties that differentiate them from conventional weblogs and forum. First, microblogs deal with almost *real-time messaging,* including instant information, expression of feelings, and immediate ideas. It also provides a source of crowd intelligence that can be used to investigate common feelings or potential trends about certain news or concepts. However, this real-time property can lead to the production of an enormous number of messages that recipients must digest. Second, micro-blogging is *time-traceable*. The temporal information is crucial because contextual posts that appear close together are, to some extent, correlated. Third, the style of micro-blogging posts tends to be *conversation-based* with a sequence of responses. This phenomenon indicates that the posts and their responses are highly correlated in many respects. Fourth, micro-blogging is *friendship-influenced*. Posts from a particular user can also be viewed by his/her friends and might have an impact on them (e.g. the empathy effect) implicitly or explicitly. Therefore, posts from friends in the same period may be correlated sentiment-wise as well as content-wise.

We leverage the above properties to develop an automatic and intuitive Web application, MemeTube, to analyze and display the sentiments behind messages in microblogs. Our system can be regarded as a sentiment-driven, music-based summarization framework as well as a novel audiovisual presentation of art. MemeTube is designed as a search-based tool. The system flow is as shown in Figure 1. Given a query (either a keyword or a user id), the system first extracts a series of relevant posts and replies based on keyword matching. Then sentiment analysis is applied to determine the sentiment of the posts. Next a piece of music is composed to reflect the detected sentiments. Finally, the messages and music are fed into the animation generation model, which displays a piano keyboard that plays automatically.

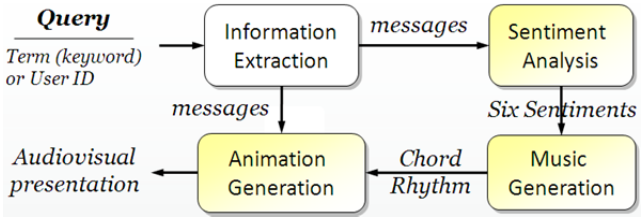


Figure 1: The system flow of our MemeTube.

The contributions of this work can be viewed from three different perspectives.

- From system perspective of view, we demo a novel Web-based system, MemeTube, as a kind of search-based sentiment presentation, musi-

[1] http://www.twitter.com
[2] http://www.plurk.com
[3] http://www.jaiku.com/

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 32–37,
Portland, Oregon, USA, 21 June 2011.

calization, and visualization tool for microblog messages. It can serve as a real-time sentiment detector or an interactive microblog audio-visual presentation system.

- Technically, we integrate a language-model-based classifier approach with a Markov-transition model to exploit three kinds of information (i.e., contextual, response, and friendship information) for sentiment recognition. We also integrate the sentiment-detection system with a real-time rule-based harmonic music and animation generator to display streams of messages in an audiovisual format.
- Conceptually, our system demonstrates that, instead of simply using textual tags to express sentiments, it is possible to exploit audio (i.e., music) and visual (i.e., animation) cues to present microblog users' feelings and experiences. In this respect, the system can also serve as a Web-based art piece that uses NLP-technologies to concretize and portray sentiments.

2 Related Works

Related works can be divided into two parts: sentiment classification in microblogs, and sentiment-based audiovisual presentation for social media. For the first part, most of related literatures focus on exploiting different classification methods to separate positive and negative sentiments by a variety of textual and linguistics features, as shown in Table 1. Their accuracy ranges from 60%~85% depending on different setups. The major difference between our work and existing approaches is that our model considers three kinds of additional information (i.e., contextual, response and friendship information) for sentiment recognition.

In recent years, a number of studies have investigated integrating emotions and music in certain media applications. For example, Ishizuka and Onisawa (2006) generated variations of theme music to fit the impressions of story scenes represented by textual content or pictures. Kaminskas (2009) aligned music with user-selected points of interests for recommendation. Li and Shan (2007) produced painting slideshows with musical accompaniment. Hua et al. (2004) proposed a Photo2Video system that allows users to specify incident music that expresses their feelings about the photos. To the best of our knowledge, MemeTube is the first attempt to exploit AI techniques to create harmonic audio-visual experiences and interactive emotion-based summarization for microblogs.

Table 1: Summary of related works that detect sentiments in microblogs.

	Features	Methods
Pak and Paroubek 2010	statistic counting of adjectives	Naive Bayes
Chen et al. 2008	POS tags, emoticons	SVM
Lewin and Pribula 2009	smileys, keywords	Maximum Entropy, SVM
Riley 2009	n-grams, smileys, hashtags, replies, URLs, usernames, emoticons	Naive Bayes,
Prasad 2010	n-grams	Naïve Bayes
Go et al. 2009	usernames, sequential patterns of keywords, POS tags, n-grams	Naive Bayes, Maximum Entropy, SVM
Li et al. 2009	several dictionaries about different kinds of keywords	Keyword Matching
Barbosa and Feng 2010	retweets, hashtag, replies, URLs, emoticons, upper cases	SVM
Sun et al. 2010	keyword counting and Chinese dictionaries	Naive Bayes, SVM
Davidov et al. 2010	n-grams, word patterns, punctuation information	*k*-Nearest Neighbor
Bermingham and Smeaton 2010	n-grams and POS tags	Binary Classification

3 Sentiment Analysis of Microblog Posts

First, we develop a classification model as our basic sentiment recognition mechanism. Given a training corpus of posts and responses annotated with sentiment labels, we train an n-gram language model for each sentiment. Then, we use such model to calculate the probability that a post expresses the sentiment *s* associated with that model:

$$Pr(p|s) = Pr(w_1, \cdots, w_m|s) \prod_{i=1}^{m} Pr\left(w_i|w_{i-(n-1)}, \cdots, w_{i-1}, s\right),$$

where *w* is the sequence of words in the post. We also use the common Laplace smoothing method.

For each post p and each sentiment $s \in S$, our classifier calculates the probability that such post expresses the sentiment $Pr(s|p)$ using Bayes rule:

$$Pr(s|p) = Pr(s)Pr(p|s)$$

$Pr(s)$ is estimated directly by counting, while $Pr(p|s)$ can be derived by using the learned lan-

guage models. This allow us to produce a distribution of sentiments for a given post p, denoted as S_p.

However, the major challenge in the microblog sentiment detection task is that the length of each post is limited (i.e., posts on Twitter are limited to 140 characters). Consequently, there might not be enough information for a sentiment detection system to exploit. To solve this problem, we propose to utilize the three types of information mentioned earlier. We discuss each type in detail below.

3.1 Response Factor

We believe the sentiment of a post is highly correlated with (but not necessary similar to) that of responses to the post. For example, an angry post usually triggers angry responses, but a sad post usually solicits supportive responses. We propose to learn the correlation patterns of sentiments from the data and use them to improve the recognition.

To achieve such goal, from the data, we learn the probability $P(Sentiment_{post}|Sentiment_{response})$, which represents the conditional probability of a post given responses. Then we use such probability to construct a transition matrix M_r, where $M_{r_{ij}} = P(Sentiment_{post} = j \mid Sentiment_{response} = i)$.

With M_r, we can generate the adjusted sentiment distribution of the post S'_p as:

$$S'_p = \alpha \times \frac{\sum_{i=1}^{k} W_{r_i} S_{r_i} M_r}{k} + (1-\alpha) S_p ,$$

where S_p denotes the original sentiment distribution of the post, and S_{r_i} is the sentiment distribution of the i^{th} response determined by the abovementioned language model approach. In addition, $W_{r_i} = 1/(t_{response_i} - t_{post})$ represents the weight of the response since it is preferable to assign higher weights to closer responses. There is also a global parameter α that determines how much the system should trust the information derived from the responses to the post. If there is no response to a post, we simply assign $S'_p = S_p$.

3.2 Context Factor

It is assumed that the sentiment of a microblog post is correlated with the author's previous posts (i.e., the 'context' of the post). We also assume that, for each person, there is a sentiment transition matrix M_c that represents how his/her sentiments change over time. The $(i,j)^{th}$ element in M_c represents the conditional probability from the sentiment of the previous post to that of the current post: $P(Sentiment(P_t) = j \mid Sentiment(P_{t-1}) = i)$.

The diagonal elements stand for the consistency of the emotion state of a person. Conceivably, a capricious person's diagnostic $M_{c_{ii}}$ values will be lower than those of a calm person. The matrix M_c can be learned directly from the annotated data.

Let S_t represent the detected sentiment distribution of an existing post at time t. We want to adjust S_t based on the previous posts from $t - \Delta t$ to t, where Δt is a given temporal threshold. The system first extracts a set of posts from the same author posted from time $t - \Delta t$ to t and determines their sentiment distributions $\{S_{t_1}, S_{t_2}, \dots, S_{t_k}\}$, where $t - \Delta t < t_1, t_2, \dots, t_k < t$ using the same classifier. Then, the system utilizes the following update equation to obtain an adjusted sentiment distribution S'_t:

$$S'_t = \alpha \times \frac{\sum_{i=1}^{k} W_{t_i} S_{t_i} M_c}{k} + (1-\alpha) S_t ,$$

where $W_{t_i} = 1/(t - t_i)$. The parameters W_{t_i}, k, α are defined similar to the previous case. If there is no post in the defined interval, the system will leave S_t unchanged.

3.3 Friendship Factor

We also assume that the friends' emotions are correlated with each other. This is because friends affect each other, and they are more likely to be in the same circumstances, and thus enjoy/suffer similarly. Our hypothesis is that the sentiment of a post and the sentiments of the author's friends' recent posts might be correlated. Therefore, we can treat the friends' recent posts in the same way as the recent posts of the author, and learn the transition matrix M_f, where $M_{f_{ij}} = P(Sentiment_{user}(P_t) = j \mid Sentiment_{user's\ friend}(P_{t-1}) = i)$, and apply the technique proposed in the previous section to improve the recognition accuracy.

However, it is not necessarily true that all friends have similar emotional patterns. One's sentiment transition matrix M_c might be very different from that of the other, so we need to be careful when using such information to adjust our recognition outcomes. We propose to only consider posts from friends with similar emotional patterns.

To achieve our goal, we first learn every user's contextual sentiment transition matrix M_c from the data. In M_c, each row represents a distribution that sums to one; therefore, we can compare two matrixes M_{c_1} and M_{c_2} by averaging the symmetric KL-divergence of each row. That is,

$$Similarity(M_1, M_2) = Average_{i=1}^{n} KL\big(Row(M_1, i), Row(M_2, i)\big).$$

Two persons are considered as having similar emotion pattern if their contextual sentiment transition matrixes are similar. After a set of similar friends are identified, their recent posts (i.e., from $t - \Delta t$ to t) are treated in the same way as the posts by the author, and we use the method proposed previously to fine-tune the recognition outcomes.

4 Music Generation

For each microblog post retrieved according to the query, we can derive its sentiment distribution (as a vector of probabilities) by using the above method. Next, the system transforms every sentiment distribution into an affective vector comprised of a valence value and an arousal value. The valence value represents the positive-to-negative sentiment, while the arousal value represents the intense-to-silent level.

We exploit the mapping from each type of sentiment to a two-dimensional affective vector based on the two-dimensional emotion model of Russell (1980). Using the model we extract the affective score vectors of the six emotions (see Table 2) used in our experiments. The mapping enables us to transform a sentiment distribution S_p into an affective score vector by weighted sum approach. For example, given a distribution of (Anger=20%, Surprise=20%,Disgust=10%, Fear=10%, Joy=10%, Sadness=30%), the two-dimensional affective vector can be computed as 0.2*(-0.25, 1) + 0.2*(0.5, 0.75) + 0.1*(-0.75, -0.5) + 0.1*(-0.75, 0.5) + 0.1*(1, 0.25) + 0.3*(-1, -0.25). Finally, the affective vector of each post will be summed to represent the sentiment of the given query in terms of the valence and arousal values.

Table 2: Affective score vector for each sentiment label.

Sentiment Label	Affective Score Vector
Anger	(-0.25, 1)
Surprise	(0.5, 0.75)
Disgust	(-0.75, -0.5)
Fear	(-0.75, 0.5)
Joy	(1, 0.25)
Sadness	(-1, -0.25)

Next the system transforms the affective vector into music elements through chord set selection (based on the valence value) and rhythm determination (based on the arousal value). For chord set selection, we design nine basic chord sets as {A, Am, Bm, C, D, Dm, Em, F, G}, where each chord set consists of some basic notes. The chord sets are used to compose twenty chord sequences. Half of the chord sequences are used for weakly positive to strongly positive sentiments and the other half are used for weakly negative to strongly negative sentiments. The valence value is therefore divided into twenty levels, and gradually shifts from strongly positive to strongly negative. The chord sets ensure that the resulting auditory presentation is in harmony (Hewitt 2008). For rhythm determination, we divide the arousal values into five levels to decide the tempo/speed of the music. Higher arousal values generate music with a faster tempo while lower ones lead to slow and easy-listening music.

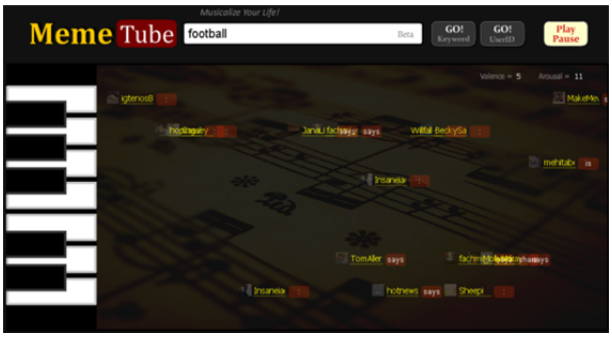


Figure 2: A snapshot of the proposed MemeTube.


Figure 3: The animation with automatic piano playing.

5 Animation Generation

In this final stage, our system produces real-time animation for visualization. The streams of messages are designed to flow as if they were playing a piece of a piano melody. We associate each message with a note in the generated music. When a post message flows from right to left and touches a piano key, the key itself blinks once and the corresponding tone of the key is produced. The message flow and the chord/rhythm have to be synchronized so that it looks as if the messages themselves are playing the piano. The system also allows users to highlight the body of a message by moving the

cursor over the flowing message. A snapshot is shown in Figure 2 and the sequential snapshots of the animation are shown in Figure 3.

6 Evaluations on Sentiment Detection

We collect the posts and responses from every effective user, users with more than 10 messages, of Plurk from January 31st to May 23rd, 2009. In order to create the diversity for the music generation system, we decide to use six different sentiments, as shown in Table 2, rather than using only three sentiment types, positive, negative and neutral, as most of the systems in Table 1 have used. The sentiment of each sentence is labeled automatically using the emoticons. This is similar to what many people have proposed for evaluation (Davidov et al. 2010; Sun et al. 2010; Bifet and Frank 2010; Go et al. 2009; Pak and Paroubek 2010; Chen et al. 2010). We use data from January 31st to April 30th as training set, May 1st to 23rd as testing data. For the purpose of observing the result of using the three factors, we filter the users without friends, the posts without responses, and the posts without previous post in 24 hour in testing data. We also manually label the sentiments on the testing data (totally 1200 posts, 200 posts for each sentiment).

We use three metrics to evaluate our model: accuracy, Root-Mean-Square Error for valence (denoted by RMSE(**V**)) and RMSE for arousal (denoted by RMSE(**A**)). The RMSE values are generated by comparing the affective vector of the predicted sentiment distribution with the affective vector of the answer. Our basic model reaches 33.8% in accuracy, 0.78 in the RMSE(V) and 0.64 in RMSE(A). Note that RMSE≈0.5 means that there is roughly one quarter (25%) error in the valence/arousal values as they range from [-1,1].

Note that the main reason the accuracy is not extremely high is that we are dealing with 6 classes. When we combine angry, disgust, fear, and sadness into one negative sense and the rest as positive senses, our system reaches **78.7%** in accuracy, which is competitive to the state-of-the-art algorithms as shown in the related work section. However, doing such generalization will lose the flexibility of producing more fruitful and diverse pieces of music. Therefore we choose more fine-grained classes for our experiment.

Figure 3 shows the results of exploiting the response, context, and friendship. Note RMSE≈0.5 means that there is roughly one quarter (25%) error in the valence/arousal values as they range from [-1,1]. The results show that considering all three additional factors can achieve the best results and decent improvement over the basic LM model.

Table 3: The results after adding addition info (note that for RMSE, the lower value the better)

	LM	Response	Context	Friend	**Combine**
Accuracy	33.8%	34.7%	34.8%	35.1%	**36.5%**
RMSE(V)	0.784	0.683	0.684	0.703	**0.679**
RMSE(A)	0.640	0.522	0.516	0.538	**0.514**

7 System Demo

We create video clips of five different queries for demonstration, which is downloadable from: http://mslab.csie.ntu.edu.tw/memetube/demo/. This demo page contains the resulting clips of four keyword queries (including *football*, *volcano*, *Monday*, *big bang*) and a user id query *mstcgeek*. Here we briefly describe each case. (1) The video for query term, *football*, was recorded on February 7th 2011, results in a relatively positive and extremely intense atmosphere. It is reasonable because the NFL Super Bowl was played on February 6th, 2011. The valence value is not as high as the arousal value because some fans might not be very happy to see their favorite team losing the game. (2) The query, *volcano*, was also recorded on February 7th 2011. The resulting video expresses negative valence and neutral arousal. After checking the posts, we have learned that it is because the Japanese volcano Mount Asama has continued to erupt. Some users are worried and discussed about the potential follow-up disasters. (3) The query *Monday* was performed on February 6th 2011, which is a Sunday night. The negative valence reflects the "blue Monday" phenomenon, which leads to some heavy, less smooth melody. (4) The term *big bang* turns out to be very positive on both valence and arousal, mainly because, besides its relatively neutral meaning in physics, this term also refers to a famous comic show that some people in Plurk love to watch. We also use one user id as query: the user-id *mstcgeek* is the official account of Microsoft Taiwan. This user often uses cheery texts to share some videos about their products or provide some discounts of their product, which leads to relatively hyped music.

8 Conclusion

Microblog, as a daily journey and social networking service, generally captures the dynamics of the change of feelings over time of the authors and their friends. In MemeTube, the affective vector is generated by aggregating the sentiment distribution of each post; thus, it represents the majority's opinion (or sentiment) about a topic. In this sense, our system can be regarded as providing users with an audiovisual experience to learn collective opinion of a particular topic. It also shows how NLP techniques can be integrated with knowledge about music and visualization to create a piece of interesting network art work. Note that MemeTube can be regarded as a flexible framework as well since each component can be further refined independently. Therefore, our future works are threefold: For sentiment analysis, we will consider more sophisticated ways to improve the baseline accuracy and to aggregate individual posts into a collective consensus. For music generation, we plan to add more instruments and exploit learning approaches to improve the selection of chords. For visualization, we plan to add more interactions between music, sentiments, and users.

Acknowledgements

This work was supported by National Science Council, National Taiwan University and Intel Corporation under Grants NSC99-2911-I-002-001, 99R70600, and 10R80800.

References

Barbosa, L., and Feng, J. 2010. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of International Conference on Computational Linguistics (COLING'10), 36–44.

Bermingham, A., and Smeaton, A. F. 2010. Classifying Sentiment in Microblogs: is Brevity an Advantage? In Proceedings of ACM International Conference on Information and Knowledge Management (CIKM'10), 1183–1186.

Chen, M. Y.; Lin, H. N.; Shih, C. A.; Hsu, Y. C.; Hsu, P. Y.; and Hewitt, M. 2008. Music Theory for Computer Musicians. Delmar.

Hsieh, S. K. 2010. Classifying Mood in Plurks. In Proceedings of Conference on Computational Linguistics and Speech Processing (ROCLING 2010), 172–183.

Davidov, D.; Tsur, O.; and Rappoport, A. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In Proceedings of International Conference on Computational Linguistics (COLING'10), 241–249.

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter Sentiment Classification using Distant Supervision. *Technical Report*, Stanford University.

Hua, X. S.; Lu, L.; and Zhang, H. J. 2004. Photo2Video - A System for Automatically Converting Photographic Series into Video. In Proceedings of ACM International Conference on Multimedia (MM'04), 708–715.

Ishizuka, K., and Onisawa, T. 2006. Generation of Variations on Theme Music Based on Impressions of Story Scenes. In Proceedings of ACM International Conference on Game Research and Development, 129–136.

Kaminskas, M. 2009. Matching Information Content with Music. In Proceedings of ACM International Conference on Recommendation System (RecSys'09), 405–408.

Lewin, J. S., and Pribula, A. 2009. Extracting Emotion from Twitter. *Technical Report*, Stanford University.

Li, C. T., and Shan, M. K. 2007. Emotion-based Impressionism Slideshow with Automatic Music Accompaniment. In Proceedings of ACM International Conference on Multimedia (MM'07), 839–842.

Li, S.; Zheng, L.; Ren, X.; and Cheng, X. 2009. Emotion Mining Research on Micro-blog. In Proceedings of IEEE Symposium on Web Society, 71–75.

Pak, A., and Paroubek, P. 2010. Twitter Based System: Using Twitter for Disambiguating Sentiment Ambiguous Adjectives. In Proceedings of International Workshop on Semantic Evaluation, (ACL'10), 436–439.

Pak, A., and Paroubek, P. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of International Conference on Language Resources and Evaluation (LREC'10), 1320–1326.

Prasad, S. 2010. Micro-blogging Sentiment Analysis Using Bayesian Classification Methods. *Technical Report*, Stanford University.

Riley, C. 2009. Emotional Classification of Twitter Messages. *Technical Report*, UC Berkeley.

Russell, J. A. 1980. Circumplex Model of Affect. Journal of Personality and Social Psychology, 39(6):1161–1178.

Strapparava, C., and Valitutti, A. 2004. Wordnet-affect: an Affective extension of wordnet. In Proceedings of International Conference on Language Resources and Evaluation, 1083–1086.

Sun, Y. T.; Chen, C. L.; Liu, C. C.; Liu, C. L.; and Soo, V. W. 2010. Sentiment Classification of Short Chinese Sentences. In Proceedings of Conference on Computational Linguistics and Speech Processing (ROCLING'10), 184–198.

Yang, C.; Lin, K. H. Y.; and Chen, H. H. 2007. Emotion Classification Using Web Blog Corpora. In Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 275–278.

An ERP-based Brain-Computer Interface for text entry using Rapid Serial Visual Presentation and Language Modeling

K.E. Hild°, U. Orhan†, D. Erdogmus†, B. Roark°, B. Oken°, S. Purwar†, H. Nezamfar†, M. Fried-Oken°
°Oregon Health and Science University †Cognitive Systems Lab, Northeastern University
{hildk,roarkb,oken,friedm}@ohsu.edu {orhan,erdogmus,purwar,nezamfar}@ece.neu.edu

Abstract

Event related potentials (ERP) corresponding to stimuli in electroencephalography (EEG) can be used to detect the intent of a person for brain computer interfaces (BCI). This paradigm is widely used to build letter-by-letter text input systems using BCI. Nevertheless using a BCI-typewriter depending only on EEG responses will not be sufficiently accurate for single-trial operation in general, and existing systems utilize many-trial schemes to achieve accuracy at the cost of speed. Hence incorporation of a language model based prior or additional evidence is vital to improve accuracy and speed. In this demonstration we will present a BCI system for typing that integrates a stochastic language model with ERP classification to achieve speedups, via the rapid serial visual presentation (RSVP) paradigm.

1 Introduction

There exist a considerable number of people with severe motor and speech disabilities. Brain computer interfaces (BCI) are a potential technology to create a novel communication environment for this population, especially persons with completely paralyzed voluntary muscles (Wolpaw, 2007; Pfurtscheller et al., 2000). One possible application of BCI is typing systems; specifically, those BCI systems that use electroencephalography (EEG) have been increasingly studied in the recent decades to enable the selection of letters for expressive language generation (Wolpaw, 2007; Pfurtscheller et al., 2000; Treder and Blankertz, 2010). However, the use of noninvasive techniques for letter-by-letter systems lacks efficiency due to low signal to noise ratio and variability of background brain activity. Therefore current BCI-spellers suffer from low symbol rates and researchers have turned to various hierarchical symbol trees to achieve system speedups (Serby et al., 2005; Wolpaw et al., 2002; Treder and Blankertz, 2010). Slow throughput greatly diminishes the practical usability of such systems. Incorporation of a language model, which predicts the next letter using the previous letters, into the decision-making process can greatly affect the performance of these systems by improving the accuracy and speed.

As opposed to the matrix layout of the popular P300-Speller (Wolpaw, 2007), shown in Figure 1, or the hexagonal two-level hierarchy of the Berlin BCI (Treder and Blankertz, 2010), we utilize another well-established paradigm: rapid serial visual presentation (RSVP), shown in Figure 2. This paradigm relies on presenting one stimulus at a time at the focal point of the screen. The sequence of stimuli are presented at relatively high speeds, each subsequent stimulus replacing the previous one, while the subject tries to perform mental target matching between the intended symbol and the presented stimuli. EEG responses corresponding to the visual stimuli are classified using regularized discriminant analysis (RDA) applied to stimulus-locked temporal features from multiple channels.

The RSVP interface is of particular utility for the most impaired users, including those suffering from locked-in syndrome (LIS). Locked-in syndrome can result from traumatic brain injury, such as a brainstem stroke[1], or from neurodegenerative diseases such as amyotrophic lateral sclerosis (ALS or Lou Gehrig's disease). The condition is characterized by near total paralysis, though the individuals are cognitively intact. While vision is retained, the motor control impairments extend to eye movements. Often the only reliable movement that can be made by

[1] Brain stem stroke was the cause of LIS for Jean-Dominique Bauby, who dictated his memoir *The Diving Bell and the Butterfly* via eyeblinks (Bauby, 1997).

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	1	2	3	4
5	6	7	8	9	_

Figure 1: Spelling grid such as that used for the P300 speller (Farwell and Donchin, 1988). '_' denotes space.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 38–43,
Portland, Oregon, USA, 21 June 2011.

Figure 2: RSVP scanning interface.

an individual is a particular muscle twitch or single eye blink, if that. Such users have lost the voluntary motor control sufficient for such an interface. Relying on extensive visual scanning or complex gestural feedback from the user renders a typing interface difficult or impossible to use for the most impaired users. Simpler interactions via brain-computer interfaces (BCI) hold much promise for effective text communication for these most impaired users. Yet these simple interfaces have yet to take full advantage of language models to ease or speed typing. In this demonstration, we will present a language-model enabled interface that is appropriate for the most impaired users.

In addition, the RSVP paradigm provides some useful interface flexibility relative to the grid-based paradigm. First, it allows for auditory rather than visual scanning, for use by the visually impaired or when visual access is inconvenient, such as in face-to-face communication. Auditory scanning is less straightforward when using a grid. Second, multi-character substrings can be scanned in RSVP, whereas the kind of dynamic re-organization of a grid that would be required to support this can be very confusing. Finally, language model integration with RSVP is relatively straightforward, as we shall demonstrate. See Roark et al. (2010) for methods integrating language modeling into grid scanning.

2 RSVP based BCI and ERP Classification

RSVP is an experimental psychophysics technique in which visual stimulus sequences are displayed on a screen over time on a fixed focal area and in rapid succession. The Matrix-P300-Speller used by Wadsworth and Graz groups (especially g.tec, Austria) opts for a spatially distributed presentation of possible symbols, highlighting them in different orders and combinations to elicit P300 responses. Berlin BCI's recent variation utilizes a 2-layer tree structure where the subject chooses among six units (symbols or sets of these) where the options are laid out on the screen while the subject focuses on a central focal area that uses an RSVP-like paradigm to elicit P300 responses. Full screen awareness is required. In contrast, our approach is to distribute the stimuli temporally and present one symbol at a time using RSVP and seek a binary response to find the desired letter, as shown in Figure 2. The latter method has the advantage of not requiring the user to look at different areas of the screen, which can be an important factor for those with LIS.

Our RSVP paradigm utilizes stimulus sequences consisting of the 26 letters in the English alphabet plus symbols for space and backspace, presented in a randomly ordered sequence. When the user sees the target symbol, the brain generates an evoked response potential (ERP) in the EEG; the most prominent component of this ERP is the P300 wave, which is a positive deflection in the scalp voltage primarily in frontal areas and that generally occurs with a latency of approximately 300 ms. This natural novelty response of the brain, occurring when the user detects a rare, sought-after target, allows us to make binary decisions about the user's intent.

The intent detection problem becomes a signal classification problem when the EEG signals are windowed in a stimulus-time-locked manner starting at stimulus onset and extending for a sufficient duration – in this case 500ms. Consider Figure 3, which shows the trial-averaged temporal signals from various EEG channels corresponding to target and non-target (distractor) symbols. This graph shows a clear effect between 300 and 500 ms for the target symbols that is not present for the distractor symbols (the latter of which clearly shows a component having a periodicity of 400 ms, which is expected in this case since a new image was presented every 400 ms). Figure 4, on the other hand, shows the magnitude of the trial and distractor responses at channel Cz on a single-trial basis, rather than averaged over all trials. The signals acquired from each EEG channel are incorporated and classified to determine the class label: ERP or non-ERP.

Our system functions as follows. First, each channel is band-pass filtered. Second, each channel is temporally-windowed. Third, a linear dimension reduction (using principal components analysis) is learned using training data and is subsequently applied to the EEG data when the system is being used. Fourth, the data vectors obtained for each channel and a given stimulus are concatenated to create the data matrix corresponding to the specified stimulus. Fifth, Regularized Discriminant Analysis (RDA) (Friedman, 1989), which estimates conditional probability densities for each class using

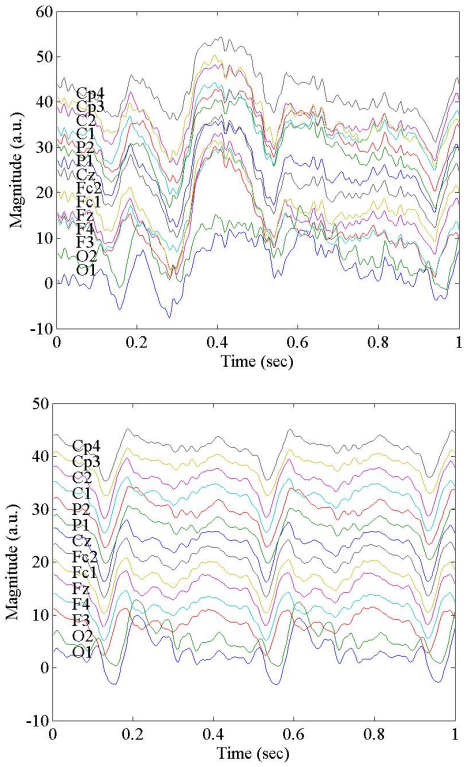


Figure 3: Trial-averaged EEG data corresponding to the target response (top) and distractor response (bottom) for a 1 second window.

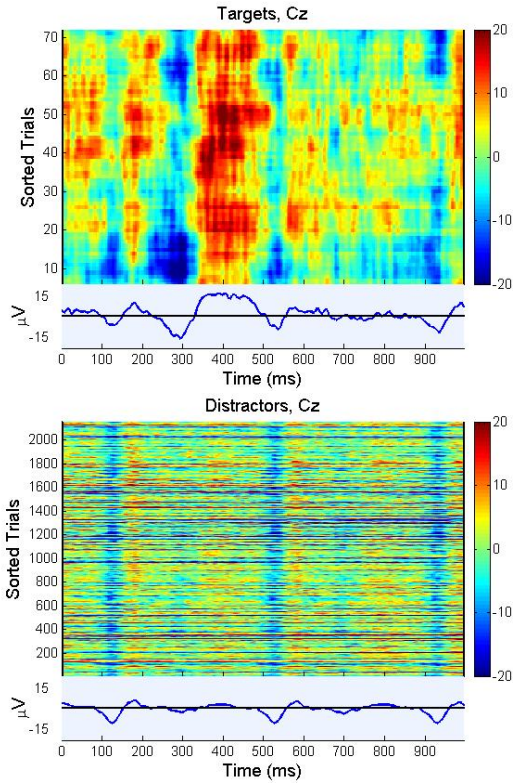


Figure 4: Single-trial EEG data at channel Cz corresponding to the target response (top) and distractor response (bottom) for a 1 second window.

Kernel Density Estimation (KDE), is used to determine a purely EEG-based classification discriminant score for each stimulus. Sixth, the conditional probability of each letter given the typed history is obtained from the language model. Seventh, Bayesian fusion (which assumes the EEG-based information and the language model information are statistically independent given the class label) is used to combine the RDA discriminant score and the language model score to generate an overall score, from which we infer whether or not a given stimulus represents an intended (target) letter.

RDA is a modified quadratic discriminant analysis (QDA) model. Assuming each class has a multivariate normal distribution and assuming classification is made according to the comparison of posterior distributions of the classes, the optimal Bayes classifier resides within the QDA model family. QDA depends on the inverse of the class covariance matrices, which are to be estimated from training data. Hence, for small sample sizes and high-dimensional data, singularities of these matrices are problematic. RDA applies regularization and shrinkage procedures to the class covariance matrix estimates in an attempt to minimize problems associated with singularities. The shrinkage procedure makes the class covariances closer to the overall data covariance, and therefore to each other, thus making the quadratic boundary more similar to a linear boundary. Shrinkage is applied as

$$\hat{\mathbf{\Sigma}}_c(\lambda) = (1-\lambda)\hat{\mathbf{\Sigma}}_c + \lambda\hat{\mathbf{\Sigma}}, \qquad (1)$$

where λ is the shrinkage parameter, $\hat{\mathbf{\Sigma}}_c$ is the class covariance matrix estimated for class $c \in \{0,1\}$, $c=0$ corresponds to the non-target class, $c=1$ corresponds to the target class, and $\hat{\mathbf{\Sigma}}$ is the weighted average of class covariance matrices. Regularization is administered as

$$\hat{\mathbf{\Sigma}}_c(\lambda,\gamma) = (1-\gamma)\hat{\mathbf{\Sigma}}_c(\lambda) + \frac{\gamma}{d}\mathrm{tr}[\hat{\mathbf{\Sigma}}_c(\lambda)]\mathbf{I}, \qquad (2)$$

where γ is the regularization parameter, $\mathrm{tr}[\cdot]$ is the trace function, and d is the dimension of the data vector.

After carrying out the regularization and shrinkage on the estimated covariance matrices, the Bayesian classification rule (Duda et al., 2001) is applied by comparing the log-likelihood ratio (using

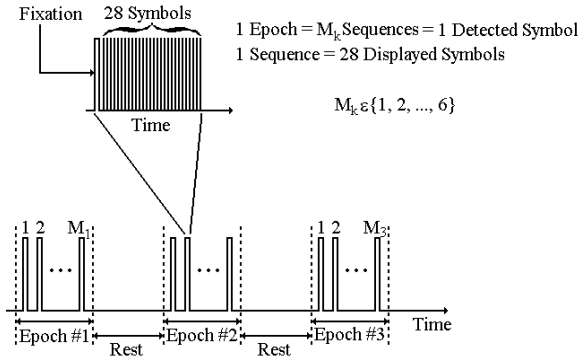


Figure 5: Timing of stimulus sequence presentation

the posterior probability distributions) with a confidence threshold. The confidence threshold can be chosen so that the system incorporates the relative risks or costs of making an error for each class. The corresponding log-likelihood ratio is given by

$$\delta_{\text{RDA}}(\mathbf{x}) = \log \frac{f_{\mathcal{N}}(\mathbf{x}; \hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\Sigma}}_1(\lambda, \gamma))\hat{\pi}_1}{f_{\mathcal{N}}(\mathbf{x}; \hat{\boldsymbol{\mu}}_0, \hat{\boldsymbol{\Sigma}}_0(\lambda, \gamma))\hat{\pi}_0}, \quad (3)$$

where $\boldsymbol{\mu}_c$ and $\hat{\pi}_c$ are the estimates of the class means and priors, respectively, $\mathbf{x}$ is the data vector to be classified, and $f_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the pdf of a multivariate normal distribution.

The set of visual stimuli (letters plus two extra symbols, in our case) can be shown multiple times to achieve a higher classification accuracy for the EEG-based classifier. The information obtained from showing the visual stimuli multiple times can easily be combined by assuming the trials are statistically independent, as is commonly assumed in EEG-based spellers[2]. Figure 5 presents a diagram of the timing of the presentation of stimuli. We define a sequence to be a randomly-ordered set of all the letters (and the space and backspace symbols). The letters are randomly ordered for each sequence because the magnitude of the ERP, hence the quality of the EEG-based classification, is commonly thought to depend on how surprised the user is to find the intended letter. Our system also has a user-defined parameter by which we are able to limit the maximum number of sequences shown to the user before our system makes a decision on the (single) intended letter. Thus we are able to operate in single-trial or multi-trial mode. We use the term *epoch* to denote all the sequences that are used by our system to make a decision on a single, intended letter. As can be seen in the timing diagram shown in Figure 5, epoch k contains between 1 and M_k sequences. This figure shows the onset of each sequence, each fixation image (which is shown at the beginning of each sequence), and each letter using narrow pulses. After each sequence is shown, the cumulative (overall) score for all letters is computed. The cumulative scores are non-negative and sum to one (summing over the 28 symbols). If the number of sequences shown is less than the user-defined limit and if the maximum cumulative score is less than 0.9, then another randomly-ordered sequence is shown to the user. Likewise, if either the maximum number of sequences has already been shown or if the maximum cumulative score equals or exceeds 0.9, then the associated symbol (for all symbols except the backspace) is added to the end of the list of previously-detected symbols, the user is able to take a break of indefinite length, and then the system continues with the next epoch. If the symbol having the maximum cumulative score is the backspace symbol, then the last item in the list of previously-detected symbols is removed and, like before, the user can take a break and then the system continues with the next epoch.

[2]The typical number of repetitions of visual stimuli is on the order of 8 or 16, although g.tec claims one subject is able to achieve reliable operation with 2 trials (verbal communication).

3 Language Modeling

Language modeling is important for many text processing applications, e.g., speech recognition or machine translation, as well as for the kind of typing application being investigated here (Roark et al., 2010). Typically, the prefix string (what has already been typed) is used to predict the next symbol(s) to be typed. The next letters to be typed become highly predictable in certain contexts, particularly word-internally. In applications where text generation/typing speed is very slow, the impact of language modeling can become much more significant. BCI-spellers, including the RSVP Keyboard paradigm presented here, can be extremely low-speed, letter-by-letter writing systems, and thus can greatly benefit from the incorporation of probabilistic letter predictions from an accurate language model.

For the current study, all language models were estimated from a one million sentence (210M character) sample of the NY Times portion of the English Gigaword corpus. Models were character n-grams, estimated via relative frequency estimation. Corpus normalization and smoothing methods were as described in Roark et al. (2010). Most importantly for

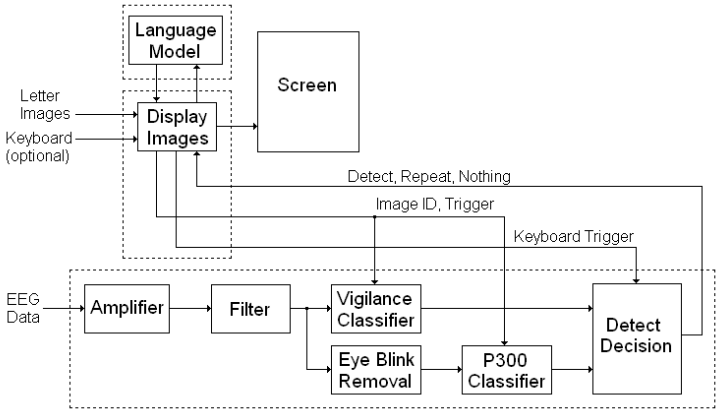


Figure 6: Block diagram of system architecture.

this work, the corpus was case normalized, and we used Witten-Bell smoothing for regularization.

4 System Architecture

Figure 6 shows a block diagram of our system. We use a Quad-core, 2.53 GHz laptop, with system code written in Labview, Matlab, and C. We also use the Psychophysics Toolbox[3] to preload the images into the video card and to display the images at precisely-defined temporal intervals. The type UB g.USBamp EEG-signal amplifier, which is manufactured by g.tec (Austria), has 24 bits of precision and has 16 channels. We use a Butterworth bandpass filter of 0.5 to 60 Hz, a 60 Hz notch filter, a sampling rate of 256 Hz, and we buffer the EEG data until we have 8 samples of 16-channel EEG data, at which point the data are transmitted to the laptop. We use either g.BUTTERfly or g.LADYbird active electrodes, a g.GAMMA cap, and the g.GAMMAsys active electrode system.

The output of the amplifier is fed to the laptop via a USB connection with a delay that is both highly variable and unknown a priori. Consequently, we are unable to rely on the laptop system clock in order to synchronize the EEG data and the onset of the visual stimuli. Instead, synchronization between the EEG data and the visual stimuli is provided by sending a parallel port trigger, via an express card-to-parallel port adaptor, to one of the digital inputs of the amplifier, which is then digitized along with the EEG data. The parallel port to g.tec cable was custom-built by Cortech Solutions, Inc. (Wilmington, North Carolina, USA). The parallel port trigger is sent immediately after the laptop monitor sends the vertical retrace signal. The mean and the standard deviation of the delay needed to trigger the parallel port has been measured to be on the order of tens of microseconds, which should be sufficiently small for our purposes.

[3] http://psychtoolbox.org/wikka.php?wakka=HomePage

5 Results

Here we report data collected from 2 subjects, one of whom is a LIS subject with very limited experience using our BCI system, and the other a healthy subject with extensive experience using our BCI system. The symbol duration was set to 400 ms, the duty cycle was set to 50%, and the maximum number of sequences per trial was set to 6. Before testing, the classifier of our system was trained on data obtained as each subject viewed 50 symbols with 3 sequences per epoch (the classifier was trained once for the LIS subject and once for the healthy subject). The healthy subject was specifically instructed to neither move nor blink their eyes, to the extent possible, while the symbols are being flashed on the screen in front of them. Instead, they were to wait until the rest period, which occurs after each epoch, to move or to blink. The subjects were free to produce whatever text they wished. The only requirement given to them concerning the chosen text was that they must not, at any point in the experiment, change what they are planning to type and they must correct all mistakes using the backspace symbol.

Figure 7 shows the results for the non-expert, LIS subject. A total of 10 symbols were correctly typed by this subject, who had chosen to spell, "THE_STEELERS_ARE_GOING_TO_...". Notice that the number of sequences shown exceeds the maximum value of 6 for 3 of the symbols. This occurs when the specified letter is mistyped one or more times. For example, for each mistyped non-backspace symbol, a backspace is required to delete

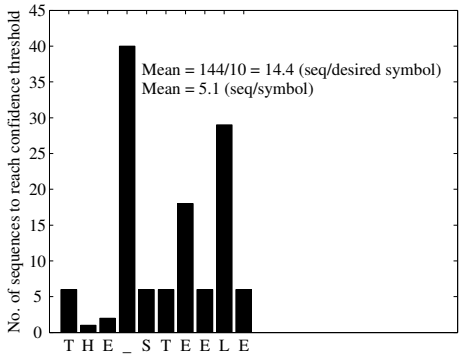


Figure 7: Number of sequences to reach the confidence threshold for the non-expert, LIS subject.

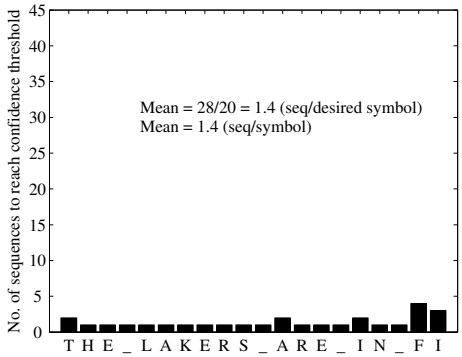


Figure 8: Number of sequences to reach the confidence threshold for the expert, healthy subject.

the incorrect symbol. Likewise, if a backspace symbol is detected although it was not the symbol that the subject wished to type, then the correct symbol must be retyped. As shown in the figure, the mean number of sequences for each correctly-typed symbol is 14.4 and the mean number of sequences per symbol is 5.1 (the latter of which has a maximum value of 6 in this case).

Figure 8 shows the result for the expert, healthy subject. A total of 20 symbols were correctly typed by this subject, who had chosen to spell, "THE_LAKERS_ARE_IN_FIRST_PLACE". The mean number of sequences for each correctly-typed symbol for this subject is 1.4 and the mean number of sequences per symbol is also 1.4. Notice that in 15 out of 20 epochs the classifier was able to detect the intended symbol on the first epoch, which corresponds to a single-trial presentation of the symbols, and no mistakes were made for any of the 20 symbols.

There are two obvious explanations as to why the healthy subject performed better than the LIS subject. First, it is possible that the healthy subject was using a non-neural signal, perhaps an electromyographic (EMG) signal stemming from an unintended muscle movement occurring synchronously with the target onset. Second, it is also possible that the LIS subject needs more training in order to learn how to control the system. We believe the second explanation is correct and are currently taking steps to make sure the LIS subject has additional time to train on our system in hopes of resolving this question quickly.

Acknowledgments

This work is supported by NSF under grants ECCS0929576, ECCS0934506, IIS0934509, IIS0914808, BCS1027724 and by NIH under grant 1R01DC009834-01. The opinions presented here are those of the authors and do not necessarily reflect the opinions of the funding agencies.

References

J.-D. Bauby. 1997. *The Diving Bell and the Butterfly*. Knopf, New York.

R.O. Duda, P.E. Hart, and D.G. Stork. 2001. *Pattern classification*. Citeseer.

L.A. Farwell and E. Donchin. 1988. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroenceph Clin. Neurophysiol.*, 70:510–523.

J.H. Friedman. 1989. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175.

G. Pfurtscheller, C. Neuper, C. Guger, W. Harkam, H. Ramoser, A. Schlogl, B. Obermaier, and M. Pregenzer. 2000. Current trends in Graz brain-computer interface (BCI) research. *IEEE Transactions on Rehabilitation Engineering*, 8(2):216–219.

B. Roark, J. de Villiers, C. Gibbons, and M. Fried-Oken. 2010. Scanning methods and language modeling for binary switch typing. In *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*, pages 28–36.

H. Serby, E. Yom-Tov, and G.F. Inbar. 2005. An improved P300-based brain-computer interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 13(1):89–98.

M.S. Treder and B. Blankertz. 2010. (C) overt attention and visual speller design in an ERP-based brain-computer interface. *Behavioral and Brain Functions*, 6(1):28.

J.R. Wolpaw, N. Birbaumer, D.J. McFarland, G. Pfurtscheller, and T.M. Vaughan. 2002. Brain-computer interfaces for communication and control. *Clinical neurophysiology*, 113(6):767–791.

J.R. Wolpaw. 2007. Brain–computer interfaces as new brain output pathways. *The Journal of Physiology*, 579(3):613.

Engkoo: Mining the Web for Language Learning

Matthew R. Scott, Xiaohua Liu, Ming Zhou, Microsoft Engkoo Team
Microsoft Research Asia
No. 5, Dan Ling Street, Haidian District, Beijing, 100080, China
{mrscott, xiaoliu, mingzhou, engkoo}@microsoft.com

Abstract

This paper presents *Engkoo* [1], a system for exploring and learning language. It is built primarily by mining translation knowledge from billions of web pages - using the Internet to catch language in motion. Currently *Engkoo* is built for Chinese users who are learning English; however the technology itself is language independent and can be extended in the future. At a system level, *Engkoo* is an application platform that supports a multitude of NLP technologies such as cross language retrieval, alignment, sentence classification, and statistical machine translation. The data set that supports this system is primarily built from mining a massive set of bilingual terms and sentences from across the web. Specifically, web pages that contain both Chinese and English are discovered and analyzed for parallelism, extracted and formulated into clear term definitions and sample sentences. This approach allows us to build perhaps the world's largest lexicon linking both Chinese and English together - at the same time covering the most up-to-date terms as captured by the net.

1 Introduction

Learning and using a foreign language is a significant challenge for most people. Existing tools, though helpful, have several limitations. Firstly, they often depend on static contents compiled by experts, and therefore cannot cover fresh words or new usages of existing words. Secondly, their search functions are often limited, making it hard for users to effectively find information they are interested in. Lastly, existing tools tend to focus exclusively on dictionary, machine translation or language learning, losing out on synergy that can reduce inefficiencies in the user experience.

This paper presents *Engkoo*, a system for exploring and learning language. Different from existing tools, it discovers fresh and authentic translation knowledge from billions of web pages - using the Internet to catch language in motion, and offering novel search functions that allow users efficient access to massive knowledge resources. Additionally, the system unifies the scenarios of dictionary, machine translation, and language learning into a seamless and more productive user experience. *Engkoo* derives its data from a process that continuously culls bilingual term/sentence pairs from the web, filters noise and conducts a series of NLP processes including POS tagging, dependency parsing and classification. Meanwhile, statistical knowledge such as collocations is extracted. Next, the mined bilingual pairs, together with the extracted linguistic knowledge, are indexed. Finally, it exposes a set of web services through which users can: 1) look up the definition of a word/phrase; 2) retrieve example sentences using keywords, POS tags or collocations; and 3) get the translation of a word/phrase/sentence.

While *Engkoo* is currently built for Chinese users who are learning English, the technology itself is language independent and can be extended to support other language pairs in the future.

We have deployed *Engkoo* online to Chinese internet users and gathered log data that suggests its

[1] http://www.engkoo.com.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 44–49,
Portland, Oregon, USA, 21 June 2011.

utility. From the logs we can see on average 62.0% of daily users are return users and 71.0% are active users (make at least 1 query); active users make 8 queries per day on average. The service receives more than one million page views per day.

This paper is organized as follows. In the next section, we briefly introduce related work. In Section 3, we describe our system. Finally, Section 4 concludes and presents future work.

2 Related Work

Online Dictionary Lookup Services. Online dictionary lookup services can be divided into two categories. The first mainly relies on the dictionaries edited by experts, e.g., Oxford dictionaries [2] and Longman contemporary English dictionary [3]. Examples of these kinds of services include iCiba [4] and Lingoes [5]. The second depends mainly on mined bilingual term/sentence pairs, e.g., Youdao [6]. In contrast to those services, our system has a higher recall and fresher results, unique search functions (e.g., fuzzy POS-based search, classifier filtering), and an integrated language learning experience (e.g., translation with interactive word alignment, and photorealistic lip-synced video tutors).

Bilingual Corpus Mining and Postprocessing. Shi et al. (2006) uses document object model (DOM) tree mapping to extract bilingual sentence pairs from aligned bilingual web pages. Jiang et al. (2009b) exploits collective patterns to extract bilingual term/sentence pairs from one web page. Liu et al. (2010) proposes training a SVM-based classifier with multiple linguistic features to evaluate the quality of mined corpora. Some methods are proposed to detect/correct errors in English (Liu et al., 2010; Sun et al., 2007). Following this line of work, *Engkoo* implements its mining pipeline with a focus on robustness and speed, and is designed to work on a very large volume of web pages.

3 System Description

In this section, we first present the architecture followed by a discussion of the basic components; we then demonstrate the main scenarios.

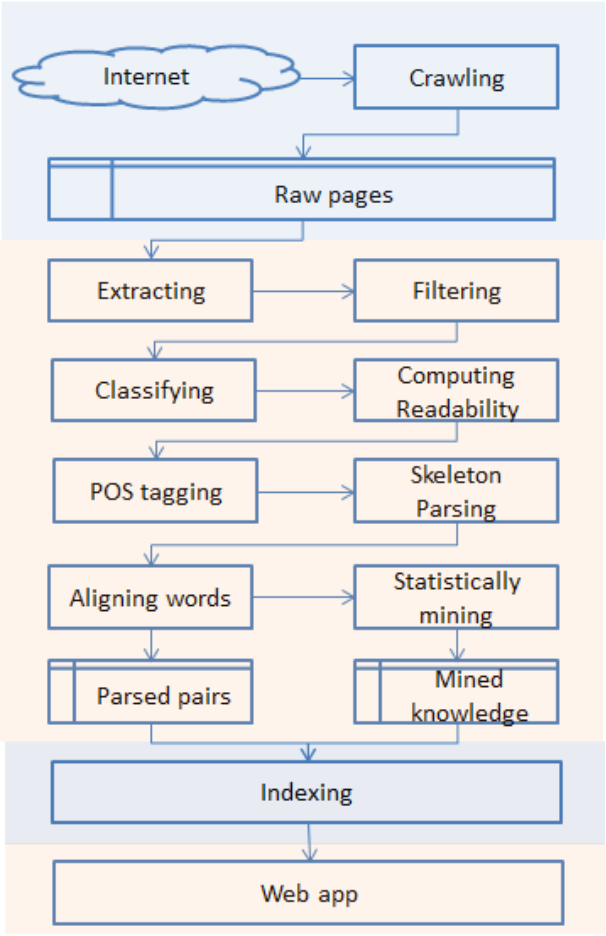


Figure 1: System architecture of *Engkoo*.

3.1 System Overview

Figure 1 presents the architecture of *Engkoo*. It can be seen that the components of *Engkoo* are organized into four layers. The first layer consists of the crawler and the raw web page storage. The crawler periodically downloads two kinds of web pages, which are put into the storage. The first kind of web pages are parallel web pages (describe the same contents but with different languages, often from bilingual sites, e.g., government sites), and the second are those containing bilingual contents. A list of seed URLs are maintained and updated after each round of the mining process.

The second layer consists of the extractor, the filter, the classifiers and the readability evaluator, which are applied sequentially. The extractor scans the raw web page storage and identifies bilingual

[2] http://oxforddictionaries.com
[3] http://www.ldoceonline.com/
[4] http://dict.en.iciba.com/
[5] http://www.lingoes.cn/
[6] http://dict.youdao.com

web page pairs using URL patterns. For example, two web pages are parallel if their URLs are in the form of "$\cdots$/zh/$\cdots$" and "$\cdots$/en/$\cdots$", respectively. Following the method of Shi et al. (2006) the extractor then extracts bilingual term/sentence pairs from parallel web pages. Meanwhile, it identifies web pages with bilingual contents, and mines bilingual term/sentence pairs from them using the method proposed by Jiang et al. (2009b). The filter removes repeated pairs, and uses the method introduced by Liu et al. (2010) to single out low quality pairs, which are further processed by a noisy-channel based sub-model that attempts to correct common spelling and grammar errors. If the quality is still unacceptable after correction, they will be dropped. The classifiers, i.e., oral/non-oral, technical/non-technical, title/non-title classifiers, are applied to each term/sentence pair. The readability evaluator assigns a score to each term/sentence pair according to Formula 1 [7].

$$206.835 - 1.015 \times \frac{\#words}{\#sentences} - 84.6 \times \frac{\#syllables}{\#words} \quad (1)$$

Two points are worth noting here. Firstly, a list of top sites from which a good number of high quality pairs are obtained, is figured out; these are used as seeds by the crawler. Secondly, bilingual term/sentence pairs extracted from traditional dictionaries are fed into this layer as well, but with the quality checking process ignored.

The third layer consists of a series of NLP components, which conduct POS tagging, dependency parsing, and word alignment, respectively. It also includes components that learn translation information and collocations from the parsed term/sentence pairs. Based on the learned statistical information, two phrase-based statistical machine translation (SMT) systems are trained, which can then translate sentences from one language to the other and vice versa. Finally, the mined bilingual term/sentence pairs, together with their parsed information, are stored and indexed with a multi-level indexing engine, a core component of this layer. The indexer is called multi-level since it uses not only keywords but also POS tags and dependency triples (e.g., "Tobj$\backsim$watch$\backsim$TV", which means "TV" is the object of "watch") as lookup entries.

The fourth layer consists of a set of services that expose the mined term/sentence pairs and the linguistic knowledge based on the built index. On top of these services, we construct a web application, supporting a wide range of functions, such as searching bilingual terms/sentences, translation and so on.

3.2 Main Components

Now we present the basic components of *Engkoo*, namely: 1) the crawler, 2) the extractor, 3) the filter, 4) the classifiers, 5) the SMT systems, and 6) the indexer.

Crawler. The crawler scans the Internet to get parallel and bilingual web pages. It employs a set of heuristic rules related to URLs and contents to filter unwanted pages. It uses a list of potential URLs to guide its crawling. That is, it uses these URLs as seeds, and then conducts a deep-first crawling with a maximum allowable depth of 5. While crawling, it maintains a cache of the URLs of the pages it has recently downloaded. It processes a URL if and only if it is not in the cache. In this way, the crawler tries to avoid repeatedly downloading the same web page. By now, about 2 billion pages have been scanned and about 0.1 parallel/bilingual pages have been downloaded.

Extractor. A bilingual term/sentence extractor is implemented following Shi et al. (2006) and Jiang et al. (2009b). It works in two modes, mining from parallel web pages and from bilingual web pages. Parallel web pages are identified recursively in the following way. Given a pair of parallel web pages, the URLs in two pages are extracted respectively, and are further aligned according to their positions in DOM trees, so that more parallel pages can be obtained. The method proposed by Jiang et al. (2007) is implemented as well to mine the definition of a given term using search engines. By now, we have obtained about 1,050 million bilingual term pairs and 100 million bilingual sentence pairs.

Filter. The filter takes three steps to drop low quality pairs. Firstly, it checks each pair if it contains any malicious word, say, a noisy symbol. Secondly, it adopts the method of Liu et al. (2010) to estimate the quality of mined pairs. Finally, following the work related to English as a second language (ESL) errors detection/correction (Liu et al., 2010; Sun et

[7] http://www.editcentral.com/gwt1/EditCentral.html

al., 2007), it implements a text normalization component based on the noisy-channel model to correct common spelling and grammar errors. That is, given a sentence $s^{'}$ possibly with noise, find the sentence $s^{*} = argmax_{s}\, p(s)p(s^{'}|s)$, where $p(s)$ and $p(s^{'}|s)$ are called the language model and the translation model, respectively. In *Engkoo*, the language model is a 5-gram language model trained on news articles using SRILM (Stolcke, 2002), while the translation model is based on a manually compiled translation table. We have got about 20 million bilingual term pairs and 15 million bilingual sentence pairs after filtering noise.

Classifiers. All classifiers adopt SVM as models, and bag of words, bi-grams as well as sentence length as features. For each classifier, about 10,000 sentence pairs are manually annotated for training/development/testing. Experimental results show that on average these classifiers can achieve an accuracy of more than 90.0%.

SMT Systems. Our SMT systems are phrase-based, trained on the web mined bilingual sentence pairs using the GIZA++ (Och and Ney, 2000) alignment package, with a collaborative decoder similar to Li et al. (2009). The Chinese-to-English/English-to-Chinese SMT system achieves a case-insensitive BLUE score of 29.6% / 47.1% on the NIST 2008 evaluation data set.

Indexer. At the heart of the indexer is the inverted lists, each of which contains an entry pointing to an ordered list of the related term/sentence pairs. Compared with its alternatives, the indexer has two unique features: 1) it contains various kinds of entries, including common keywords, POS taggers, dependency triples, collocations, readability scores and class labels; and 2) the term/sentence pairs related to the entry are ranked according to their qualities computed by the filter.

3.3 Using the System

Definition Lookup. Looking up a word or phrase on *Engkoo* is a core scenario. The traditional dictionary interface is extended with a blending of web-mined and ranked term definitions, sample sentences, synonyms, collocations, and phonetically similar terms. The result page user experience includes an intuitive comparable tabs interface described in Jiang et al. (2009a) that effectively exposes differences between similar terms. The search experience is augmented with a fuzzy auto completion experience, which besides traditional prefix matching is also robust against errors and allows for alternative inputs. All of these contain inline micro translations to help users narrow in on their intended search. Errors are resolved by a blend of edit-distance and phonetic search algorithms tuned for Chinese user behavior patterns identified by user study. Alternative input accepted includes Pinyin (Romanization of Chinese characters) which returns transliteration, as well as multiple wild card operators.

Take for example the query "tweet," illustrated in Figure 2(a). The definitions for the term derived from traditional dictionary sources are included in the main definition area and refer to the noise of a small bird. Augmenting the definition area are "Web translations," which include the contemporary use of the word standing for micro-blogging. Web-mined bilingual sample sentences are also presented and ranked by popularity metrics; this demonstrates the modern usage of the term.

Search of Example Sentences. *Engkoo* exposes a novel search and interactive exploration interface for the ever-growing web-mined bilingual sample sentences in its database. Emphasis is placed on sample sentences in *Engkoo* because of their crucial role in language learning. *Engkoo* offers new methods for the self-exploration of language based on the applied linguistic theories of "learning as discovery" and Data-Driven Learning (DDL) introduced by Johns (1991). One can search for sentences as they would in traditional search engines or concordancers. Extensions include allowing for mixed input of English and Chinese, and POS wild cards enabled by multi-level indexing. Further, sentences can be filtered based on classifiers such as oral, written, and technical styles, source, and language difficulty. Additionally sample sentences for terms can be filtered by their inflection and the semantics of a particular definition. Interactivity can be found in the word alignment between the languages as one moves his or her mouse over the words, which can also be clicked on for deeper exploration. And in addition to traditional text-to-speech, a visual representation of a human language tutor pronouncing each sentence is also included. Sample sentences between two similar words can be displayed side-by-side in a tabbed

(a) A screenshot of the definition and sample sentence areas of a *Engkoo* result page.

(b) A screenshot of samples sentences for the POS-wildcard query "v. tv" (meaning "verb TV").

(c) A screenshot of machine translation integrated into the dictionary experience, where the top pane shows results of machine translation while the bottom pane displays example sentences mined from the web.

Figure 2: Three scenarios of *Engkoo*.

user interface to easily expose the subtleties between usages.

In the example seen in Figure 2(b), a user has searched for the collocation verb+TV, represented by the query "v. TV" to find commonly used verbs describing actions for the noun "TV." In the results, we find fresh and authentic sample sentences mined from the web, the first of which contains "watch TV," the most common collocation, as the top result. Additionally, the corresponding keyword in Chinese is automatically highlighted using statistical alignment techniques.

Machine Translation. For many users, the difference between a machine translation (MT) system and a translation dictionary are not entirely clear. In *Engkoo*, if a term or phrase is out-of-vocabulary, a MT result is dynamically returned. For shorter MT queries, sample sentences might also be returned as one can see in Figure 2(c) which expands the search and also raises confidence in a translation as one can observe it used on the web. Like the sample sentences, word alignment is also exposed on the machine translation. As the alignment naturally serves as a word breaker, users can click the selection for a lookup which would open a new tab with the definition. This is especially useful in cases where a user might want to find alternatives to a particular part of a translation. Note that the seemingly single line dictionary search box is also adapted to MT behavior, allowing users to paste in multi-line text as it can detect and unfold itself to a larger text area as needed.

4 Conclusions and Future work

We have presented *Engkoo*, a novel online translation system which uniquely unifies the scenarios of dictionary, machine translation, and language learning. The features of the offering are based on an ever-expanding data set derived from state-of-the-art web mining and NLP techniques. The contribution of the work is a complete software system that maximizes the web's pedagogical potential by exploiting its massive language resources. Direct user feedback and implicit log data suggest that the service is effective for both translation utility and language learning, with advantages over existing services. In future work, we are examining extracting language knowledge from the real-time web for translation in news scenarios. Additionally, we are actively mining other language pairs to build a multi-language learning system.

Acknowledgments

We thank Cheng Niu, Dongdong Zhang, Frank Soong, Gang Chen, Henry Li, Hao Wei, Kan Wang, Long Jiang, Lijuan Wang, Mu Li, Tantan Feng, Weijiang Xu and Yuki Arase for their valuable contributions to this paper, and the anonymous reviewers for their valuable comments.

References

Long Jiang, Ming Zhou, Lee-Feng Chien, and Cheng Niu. 2007. Named entity translation with web mining and transliteration. In *IJCAI*, pages 1629–1634.

Gonglue Jiang, Chen Zhao, Matthew R. Scott, and Fang Zou. 2009a. Combinable tabs: An interactive method of information comparison using a combinable tabbed document interface. In *INTERACT*, pages 432–435.

Long Jiang, Shiquan Yang, Ming Zhou, Xiaohua Liu, and Qingsheng Zhu. 2009b. Mining bilingual data from the web with adaptively learnt patterns. In *ACL/AFNLP*, pages 870–878.

Tim Johns. 1991. From printout to handout: grammar and vocabulary teaching in the context of data driven learning. *Special issue of ELR Journal*, pages 27–45.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. Collaborative decoding: Partial hypothesis re-ranking using translation consensus between decoders. In *ACL/AFNLP*, pages 585–592.

Xiaohua Liu and Ming Zhou. 2010. Evaluating the quality of web-mined bilingual sentences using multiple linguistic features. In *IALP*, pages 281–284.

Xiaohua Liu, Bo Han, Kuan Li, Stephan Hyeonjun Stiller, and Ming Zhou. 2010. Srl-based verb selection for esl. In *EMNLP*, pages 1068–1076.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *ACL*.

Lei Shi, Cheng Niu, Ming Zhou, and Jianfeng Gao. 2006. A dom tree alignment model for mining parallel data from the web. In *ACL*, pages 489–496.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICSLP*, volume 2, pages 901–904.

Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, and Chin-Yew Lin. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *ACL*.

Dr Sentiment Knows Everything!

Amitava Das and **Sivaji Bandyopadhyay**
Department of Computer Science and Engineering
Jadavpur University
India
amitava.santu@gmail.com sivaji_cse_ju@yahoo.com

Abstract

Sentiment analysis is one of the hot demanding research areas since last few decades. Although a formidable amount of research have been done, the existing reported solutions or available systems are still far from perfect or do not meet the satisfaction level of end users'. The main issue is the various conceptual rules that govern sentiment and there are even more clues (possibly unlimited) that can convey these concepts from realization to verbalization of a human being. Human psychology directly relates to the unrevealed clues and governs the sentiment realization of us. Human psychology relates many things like social psychology, culture, pragmatics and many more endless intelligent aspects of civilization. Proper incorporation of human psychology into computational sentiment knowledge representation may solve the problem. In the present paper we propose a template based online interactive gaming technology, called ***Dr Sentiment*** to automatically create the ***PsychoSentiWordNet*** involving internet population. The PsychoSentiWordNet is an extension of SentiWordNet that presently holds human psychological knowledge on a few aspects along with sentiment knowledge.

1 Introduction

In order to identify sentiment from a text, lexical analysis plays a crucial role. For example, words like ***love, hate, good*** and ***favorite*** directly indicate sentiment or opinion. Previous works (Pang et al., 2002; Wiebe and Mihalcea, 2006; Baccianella et. al., 2010) have already proposed various techniques for making dictionaries for those sentiment words. But polarity assignment of such sentiment lexicons is a hard semantic disambiguation problem. The regulating aspects which govern the lexical level semantic orientation are natural language context (Pang et al., 2002), language properties (Wiebe and Mihalcea, 2006), domain pragmatic knowledge (Aue and Gamon, 2005), time dimension (Read, 2005), colors and culture (Strapparava and Ozbal, 2010) and many more unrevealed hidden aspects. Therefore it is a challenging and enigmatic research problem.

The current trend is to attach **prior polarity** to each entry at the sentiment lexicon level. Prior polarity is an approximation value based on heuristics based statistics collected from corpus and not exact. The probabilistic fixed point prior polarity scores do not solve the problem completely rather it places the problem into next level, called contextual polarity classification.

We start with the hypothesis that the summation of all the regulating aspects of sentiment orientation is human psychology and thus it is a multifaceted problem (Liu, 2010). More precisely what we mean by human psychology is the union of all known and unknown aspects that directly or indirectly govern the sentiment orientation knowledge of us. The regulating aspects wrapped in the present PsychoSentiWordNet are **Gender**, **Age**, **City**, **Country**, **Language and Profession**.

The PsychoSentiWordNet is an extension of the existing SentiWordNet 3.0 (Baccianella et. al., 2010) to hold the possible psychological ingredients and govern the sentiment understandability of us. The PsychoSentiWordNet holds variable prior polarity scores that could be fetched depending upon those psychological regulating aspects.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 50–55,
Portland, Oregon, USA, 21 June 2011.

An example with the input word '***High***' may illustrate the definition better:

Aspects (Profession)	Polarity
Null	Positive
Businessman	Negative
Share Broker	Positive

In this paper, we propose an interactive gaming (Dr Sentiment) technology to collect psycho-sentimental polarity for lexicons. This technology has proven itself as an excellent technique to collect psychological sentiment of human society even at multilingual level. Dr Sentiment presently supports 56 languages and therefore we may call it ***Global PsychoSentiWordNet***. The supported languages by Dr Sentiment are reported in Table 1.

In this section we have philosophically argued about the necessity of developing PsychoSentiWordNet. In the next section 2 we will describe the technical details of the proposed architecture for building the lexical resource. Section 3 explains about some exciting outcomes of PsychoSentiWordNet. The developed PsychoSentiWordNet(s) are expected to help automatic sentiment analysis research in many aspects and other disciplines as well and have been described in section 4.The data structure and the organization are described in section 5. The conclusion is drawn in section 6.

2 Dr Sentiment

Dr Sentiment[1] is a template based interactive online game, which collects player's sentiment by asking a set of simple template based questions and finally reveals a player's sentimental status. Dr Sentiment fetches random words from SentiWordNet synsets and asks every player to tell about his/her sentiment polarity understanding regarding the concept behind the word fetched by it.

There are several motivations behind developing the intuitive game to automatically collect human psycho-sentimental orientation information.

In the history of Information Retrieval research there is a milestone when ESP game[2] (Ahn et al., 2004) innovated the concept of a game to automatically label images available in the World Wide Web. It has been identified as the most reliable strategy to automatically annotate the online images. We are highly motivated by the success of the Image Labeler game.

A number of research endeavors could be found in the literature for creation of Sentiment Lexicon in several languages and domains. These techniques can be broadly categorized into two classes, one follows classical manual annotation techniques (Andreevskaia and Bergler, 2006);(Wiebe and Riloff, 2006) while the other follows various automatic techniques (Mohammad et al., 2008). Both types of techniques have few limitations. Manual annotation techniques are undoubtedly trustable but it generally takes time. Automatic techniques demand manual validations and are dependent on the corpus availability in the respective domain. Manual annotation techniques require a large number of annotators to balance one's sentimentality in order to reach agreement. But human annotators are quite unavailable and costly.

Sentiment is a property of human intelligence and is not entirely based on the features of a language. Thus people's involvement is required to capture the sentiment of the human society. We have developed an online game to attract internet population for the creation of PsychoSentiWordNet automatically. Involvement of Internet population is an effective approach as the population is very high in number and ever growing (approx. 360,985,492) [3] . Internet population consists of people with various languages, cultures, age etc and thus not biased towards any domain, language or particular society. A detailed statistics on the Internet usage and population has been reported in the Table 2.

The lexicons tagged by this system are credible as it is tagged by human beings. It is not a static sentiment lexicon set [polarity changes with time (Read, 2005)] as it is updated regularly. Around 10-20 players each day are playing it throughout the world in different languages. The average number of tagging per word is about 7.47 till date.

The Sign Up form of the "Dr Sentiment" game asks the player to provide personal information such as Sex, Age, City, Country, Language and Profession. These collected personal details of a player are kept as a log record in the database.

The gaming interface has four types of question templates. The question templates are named as Q1, Q2, Q3 and Q4.

[1] http://www.amitavadas.com/Sentiment%20Game/index.php
[2] http://www.espgame.org/
[3] http://www.internetworldstats.com/stats.htm

Languages							
Afrikaans	Bulgarian	Dutch	German	Irish	Malay	Russian	Thai
Albanian	Catalan	Estonian	Greek	Italian	Maltese	Serbian	Turkish
Arabic	Chinese	Filipino	Haitian	Japanese	Norwegian	Slovak	Ukrainian
Armenian	Croatian	Finnish	Hebrew	Korean	Persian	Slovenian	Urdu
Azerbaijani	Creole	French	Hungarian	Latvian	Polish	Spanish	Vietnamese
Basque	Czech	Galician	Icelandic	Lithuanian	Portuguese	Swahili	Welsh
Belarusian	Danish	Georgian	Indonesian	Macedonian	Romanian	Swedish	Yiddish

Table 1: Languages

WORLD INTERNET USAGE AND POPULATION STATISTICS						
World Regions	**Population (2010 Est.)**	**Internet Users Dec. 31, 2000**	**Internet Users Latest Data**	**Penetration (Population)**	**Growth 2000-2010**	**Users % of Table**
Africa	1,013,779,050	4,514,400	110,931,700	10.9 %	2,357.3 %	5.6 %
Asia	3,834,792,852	114,304,000	825,094,396	21.5 %	621.8 %	42.0 %
Europe	813,319,511	105,096,093	475,069,448	58.4 %	352.0 %	24.2 %
Middle East	212,336,924	3,284,800	63,240,946	29.8 %	1,825.3 %	3.2 %
North America	344,124,450	108,096,800	266,224,500	77.4 %	146.3 %	13.5 %
Latin America/Caribbean	592,556,972	18,068,919	204,689,836	34.5 %	1,032.8 %	10.4 %
Oceania / Australia	34,700,201	7,620,480	21,263,990	61.3 %	179.0 %	1.1 %
WORLD TOTAL	6,845,609,960	360,985,492	1,966,514,816	28.7 %	444.8 %	100.0 %

Table 2: Internet Usage and Population Statistics

To make the gaming interface more interesting images have been added. These images have been retrieved by Google image search API[4] and to avoid biasness we have randomized among the first ten images retrieved by Google.

2.1 Gaming Strategy

Dr Sentiment asks 30 questions to each player. There are predefined distributions of each question type as 11 for Q1, 11 for Q2, 4 for Q3 and 4 for Q4. These numbers are arbitrarily chosen and randomly changed for experimentation. The questions are randomly asked to keep the game more interesting. For word based translation Google translation[5] service has been used. At each Question (Q) level translation service has been used to display the sentiment word into player's own language. Google API provides multiple senses for word level translation and currently only the first sense has been picked automatically.

2.2 Q1

An English word from the English SentiWordNet synset is randomly chosen. The Google image search API is fired with the word as a query. An image along with the word itself is shown in the Q1 page of the game.

Players press the different emoticons (Figure 1) to express their sentimentality. The interface keeps log records of each interaction.

Figure 1: Emoticons to Express Player's Sentiment

2.3 Q2

This question type is specially designed for relative scoring technique. For example: ***good*** and ***better*** both are positive but we need to know which one is more positive than other. Table 3 shows how in SentiWordNet relative scoring has been made. With the present gaming technology relative polarity scoring has been assigned to each ***n-n*** word pair combination.

Randomly ***n*** (presently 2-4) words have been chosen from the source SentiWordNet synsets along with their images as retrieved by Google API. Each player is then asked to select one of them that he/she likes most. The relative score is calculated and stored in the corresponding log table.

Word	Positivity	Negativity
Good	0.625	0.0
Better	0.875	0.0
Best	0.980	0.0

Table 3: Relative Sentiment Scores in SentiWordNet

[4] http://code.google.com/apis/imagesearch/
[5] http://translate.google.com/

2.4 Q3

The player is asked for any positive word in his/her mind. This technique helps to increase the coverage of existing SentiWordNet. The word is then added to the existing PsychoSentiWordNet and further used in Q1 to other users to note their sentimentality about the particular word.

2.5 Q4

A player is asked by Dr Sentiment about any negative word. The word is then added to the existing PsychoSentiWordNet and further used in Q1 to other users to note their sentimentality about the particular word.

2.6 Comment Architecture

There are three types of Comments, Comment type 1 (CMNT1), Comment type 2 (CMNT2) and the final comment as Dr Sentiment's prescription. CMNT1 type and CMNT2 type comments are associated with question types Q1 and Q2 respectively.

2.6.1 CMNT1

Comment type 1 has 5 variations as shown in the Comment table in Table 4. Comments are randomly retrieved from comment type table according to their category:

- Positive word has been tagged as negative (PN)
- Positive word has been tagged as positive (PP)
- Negative word has been tagged as positive (NP)
- Negative word has been tagged as negative (NN)
- Neutral. (NU)

2.6.2 CMNT2

The strategy here is as same as the CMNT 1. Comment type 2 has only two variations as.

- Positive word has been tagged as negative (PN)
- Negative word has been tagged as positive (NP)

2.7 Dr Sentiment's Prescription

The final prescription depends on various factors such as total number of positive, negative or neutral comments and the total time taken by any player. The final prescription also depends on the range of the accumulated values of all the above factors.

This is the most important appealing factor to a player. The motivating message for players is that Dr Sentiment can reveal their sentimental status: whether they are extreme negative or positive or very much neutral or diplomatic etc. It is not claimed that the revealed status of a player by Dr Sentiment is exact or ideal. It is only to make the players motivated but the outcomes of the game effectively helps to store human sentimental psychology in terms of computational lexicon.

A word previously tagged by a player is avoided by the tracking system during subsequent turns by the same player. The intension is to tag more and more words involving Internet population. We observe that the strategy helps to keep the game interesting as a large number of players return to play the game after this strategy was implemented.

3 Senti-Mentality

PsychoSentiWordNet gives a good sketch to understand the psycho-sentimental behavior of the human society depending upon proposed psychological dimensions. The PsychoSentiWordNet is basically the log records of every player's tagged words.

3.1 Concept-Culture-Wise Analysis

The word "blue" gets tagged by different players around the world. But surprisingly it has been tagged as positive from one part of the world and negative from another part of the world. The graphical illustration in Figure 2 may explain the situation better. The observation is that most of the negative tags are coming from the middle-east and especially from the Islamic countries.

PN	PP	NP	NN	NU
You don't like <word>!	Good you have a good choice!	Is <word> good!	Yes <word> is too bad!	You should speak out frankly!
You should like <word>!	I love <word> too!	I hope it is a bad choice!	You are quite right!	You are too diplomatic!
But <word> is a good itself!	I support your view!	I don't agree with you!	I also don't like <word>!	Why you hiding from me? I am Dr Sentiment.

Table 4: Comments

We found a line in Wiki[6] (see in Religion Section) that may provide a good explanation: "Blue in Islam: In verse 20:102 of the Qur'an, the word زرق zurq (plural of azraq 'blue') is used metaphorically for evil doers whose eyes are glazed with fear". But other explanations may be there for this situation. This is an interesting observation that supports the effectiveness of the developed PsychoSentiWordNet. This information could be further retrieved from the developed source by giving information like (blue, Italy), (blue, Iraq) or (blue, USA) etc.

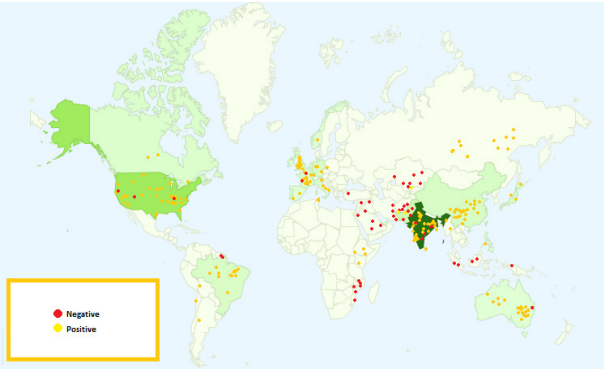


Figure 2: Geospatial Senti-Mentality

3.2 Age-Wise Analysis

Another interesting observation is that sentimentality may vary age-wise. For better understanding we look at the total statistics and the age wise distribution of all the players. Total 533 players have taken part till date. The total number of players for each range of age is shown at the top of every bar.

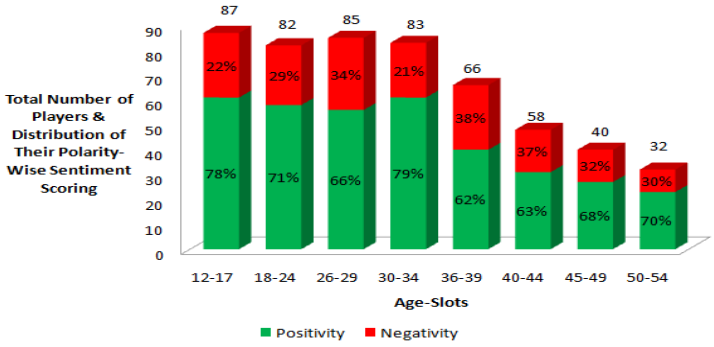


Figure 3: Age-Wise Senti-Mentality

In Figure 3 the horizontal bars are divided into two colors (Green depicts the Positivity and Red depicts the negativity) according to the total positivity and negativity scores, gathered during playing. This sociological study gives an idea on the variation of sentimentality with age. This information may be retrieved from the developed source by giving information like (X, 36-39) or (X, 45-49) etc where X denotes any arbitrary lexicon synset.

3.3 Gender-Wise Analysis

It is observed from the collected statistics that women are more positive than men! The variations in sentimentality among men and women are shown in the following Figure 4.

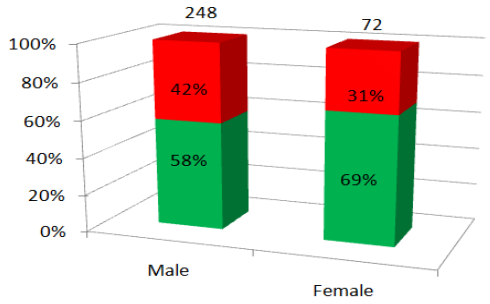


Figure 4: Gender Specific Senti-Mentality

3.4 Other-Wise

We have described several important observations in the previous sections and there are other important observations as well. Studies on the combinations of the proposed psychological dimensions, such as, location-age, location-profession and gender-location may reveal some interesting results.

4 Expected Impact of the Resource

Undoubtedly the generated PsychoSentiWordNet(s) are important resources for sentiment/opinion or emotion analysis task. Moreover the other non linguistic psychological dimensions are very much important for further analysis as well as for several newly discovered sub-disciplines such as: Geospatial Information retrieval (Egenhofer, 2002), Personalized search (Gaucha et al., 2003), Recommender System (Adomavicius and Tuzhilin, 2005), Sentiment Tracking (Tong, 2001) etc.

5 The Data Structure and Organization

Deciding on the data structure for the PsychoSentiWordNet was not trivial. Presently RDBMS (Relational Database Management System) has been

[6] http://en.wikipedia.org/wiki/Blue

used. Several tables are being used to keep user's clicking log and their personal information.

As one of the research motivations was to generate up-to-date prior polarity scores across various dimensions, we decided to generate web service API through which the people can access latest prior polarity scores. The developed PsychoSentiWordNet is expected to perform better than a static sentiment lexicon.

6 Conclusion and Future Directions

In the present paper the development of the ***PsychoSentiWordNet*** for 56 languages has been described. No evaluation has been done yet as there is no data available for this kind of experimentation and to the best of our knowledge this is the first endeavor where sentiment analysis meets AI and psychology.

Our present goal is to collect such corpus and carry out experiments to check whether variable prior polarity scores of PsychoSentiWordNet excel over the fixed point prior polarity score of SentiWordNet.

Automatically picked first sense from Google translation API may cause difficulties for cross lingual projection of sentiment synsets. Erroneous outputs from API may also cause some problems. But these problems lead to another research issue that may be termed as cross lingual sentiment synset linking. Presently we are giving a closer look to the qualitative analysis of developed multilingual psycho-sentiment lexicons.

Acknowledgment

The work reported in this paper was supported by a grant from the India-Japan Cooperative Program (DST-JST) Research project entitled "***Sentiment Analysis where AI meets Psychology***" funded by Department of Science and Technology (DST), Government of India.

References

Adomavicius Gediminas and Alexander Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. In the Proc. of IEEE Transactions on Knowledge and Data Engineering, VOL. 17, NO. 6, June 2005. ISSN 1041-4347/05. Pages 734-749.

Ahn Luis von and Laura Dabbish. Labeling Images with a Computer Game.In the Proc. of ACM CHI 2004.

Andreevskaia Alina and Bergler Sabine. CLaC and CLaC-NB: Knowledge-based and corpus-based approaches to sentiment tagging. In the Proc. of the 4th SemEval-2007, Pages 117–120, Prague, June 2007.

Aue A. and Gamon M., Customizing sentiment classifiers to new domains: A case study. In the Proc. Of RANLP, 2005.

Baccianella Stefano, Andrea Esuli, and Fabrizio Sebastiani. SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In the Proc. of LREC-10.

Bo Pang, Lee Lillian, and Vaithyanathan Shivakumar. Thumbs up? Sentiment classification using machine learning techniques. In the Proc. of EMNLP, Pages 79–86, 2002.

Egenhofer M.. Toward the Semantic Geospatial Web. ACM-GIS 2002, McLean, VI A. Voisard and S.-C. Chen (eds.), Pages. 1-4, November 2002.

Gaucha Susan, Jason Chaffeeb and Alexander Pretschnerc. Ontology-based personalized search and browsing. In Proc. of Web Intelligence and Agent Systems: An international journal. 2003. Pages 219–234. ISSN 1570-1263/03.

Liu Bing . Sentiment Analysis: A Multi-Faceted Problem.In the IEEE Intelligent Systems, 2010.

Read Jonathon. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In the Proc. of the ACL Student Research Workshop, 2005.

Richard M. Tong. An operational system for detecting and tracking opinions in online discussion. In the Proc. of the Workshop on Operational Text Classification (OTC), 2001.

Saif Mohammad, Dorr Bonnie and Hirst Graeme. Computing Word-Pair Antonymy. In the Proc. of EMNLP-2008.

Strapparava, C. and Valitutti, A. WordNet-Affect: an affective extension of WordNet. In Proc. of LREC 2004, Pages 1083 – 1086

Wiebe Janyce and Mihalcea Rada. Word sense and subjectivity. In the Proc. of COLING/ACL-06. Pages 1065-1072.

BLAST: A Tool for Error Analysis of Machine Translation Output

Sara Stymne
Department of Computer and Information Science
Linköping University, Linköping, Sweden
sara.stymne@liu.se

Abstract

We present BLAST, an open source tool for error analysis of machine translation (MT) output. We believe that error analysis, i.e., to identify and classify MT errors, should be an integral part of MT development, since it gives a qualitative view, which is not obtained by standard evaluation methods. BLAST can aid MT researchers and users in this process, by providing an easy-to-use graphical user interface. It is designed to be flexible, and can be used with any MT system, language pair, and error typology. The annotation task can be aided by highlighting similarities with a reference translation.

1 Introduction

Machine translation evaluation is a difficult task, since there is not only one correct translation of a sentence, but many equally good translation options. Often, machine translation (MT) systems are only evaluated quantitatively, e.g. by the use of automatic metrics, which is fast and cheap, but does not give any indication of the specific problems of a MT system. Thus, we advocate human error analysis of MT output, where humans identify and classify the problems in machine translated sentences.

In this paper we present BLAST,[1] a graphical tool for performing human error analysis, from any MT system and for any language pair. BLAST has a graphical user interface, and is designed to be easy and intuitive to work with. It can aid the user by highlighting similarities with a reference sentence. BLAST is flexible in that it can be used with output from any MT system, and with any hierarchical error typology. It has a modular design, allowing easy extension with new modules. To the best of our knowledge, there is no other publicly available tool for MT error annotation. Since we believe that error analysis is a vital complement to MT evaluation, we think that BLAST can be useful for many other MT researchers and developers.

[1]The BiLingual Annotation/Annotator/Analysis Support Tool, available for download at http://www.ida.liu.se/~sarst/blast/

2 MT Evaluation and Error Analysis

Hovy et al. (2002) discussed the complexity of MT evaluation, and stressed the importance of adjusting evaluation to the purpose and context of the translation. However, MT is very often only evaluated quantitatively using a single metric, especially in research papers. Quantitative evaluations can be automatic, using metrics such as Bleu (Papineni et al., 2002) or Meteor (Denkowski and Lavie, 2010), where the MT output is compared to one or more human reference translations. Metrics, however, only give a single quantitative score, and do not give any information about the strengths and weaknesses of the system. Comparing scores from different metrics can give a very rough indication of some major problems, especially in combination with a part-of-speech analysis (Popović et al., 2006).

Human evaluation is also often quantitative, for instance in the form of estimates of values such as adequacy and fluency, or by ranking sentences from different systems (e.g. Callison-Burch et al. (2007)). A combination of human and automatic metrics is

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 56–61,
Portland, Oregon, USA, 21 June 2011.

human-targeted metrics such as HTER, where a human corrects the output of a system to the closest correct translation, on which standard metrics such as TER is then computed (Snover et al., 2006). While these types of evaluation are certainly useful, they are expensive and time-consuming, and still do not tell us anything about the particular errors of a system.[2]

Thus, we think that qualitative evaluation is an important complement, and that error analysis, the identification and classification of MT errors, is an important task. There have been several suggestions for general MT error typologies (Flanagan, 1994; Vilar et al., 2006; Farrús et al., 2010), targeted at different user groups and purposes, focused on either evaluation of single systems, or comparison between systems. It is also possible to focus error analysis at a specific problem, such as verb form errors (Murata et al., 2005).

We have not been able to find any other freely available tool for error analysis of MT. Vilar et al. (2006) mentioned in a footnote that "a tool for highlighting the differences [between the MT system and a correct translation] also proved to be quite useful" for error analysis. They do not describe this tool any further, and do not discuss if it was also used to mark and store the error annotations themselves.

Some tools for post-editing of MT output, a related activity to error analysis, have been described in the literature. Font Llitjós and Carbonell (2004) presented an online tool for eliciting information from the user when post-editing sentences, in order to improve a rule-based translation system. The post-edit operations were labeled with error categories, making it a type of error analysis. This tool was highly connected to their translation system, and it required users to post-edit sentences by modifying word alignments, something that many users found difficult. Glenn et al. (2008) described a post-editing tool used for HTER calculation, which has been used in large evaluation campaigns. The tool is a pure post-editing tool and the edits are not classified. Graphical tools have also successfully been used to aid humans in other MT-related tasks, such as human MT evaluation of adequacy, fluency and system comparison (Callison-Burch et al., 2007), and word alignment (Ahrenberg et al., 2003).

[2]Though it does, at least in principle, seem possible to mine HTER annotations for more information

3 System Overview

BLAST is a tool for human annotations of bilingual material. Its main purpose is error analysis for machine translation. BLAST is designed for use in any MT evaluation project. It is not tied to the information provided by specific MT systems, or to specific languages, and it can be used with any hierarchical error typology. It has a preprocessing module for automatically aiding the annotator by highlighting similarities between the MT output and a reference. Its modular design allows easy integration of new modules for preprocessing. BLAST has three working modes for handling error annotations: for adding new annotations, for editing existing annotations, and for searching among annotations.

BLAST can handle two types of annotations: error annotations and support annotations. Error annotations are based on a hierarchical error typology, and are used to annotate errors in MT output. Error annotations are added by the users of BLAST. Support annotations are used as a support to the user, currently to mark similarities in the system and reference sentences. The support annotations are normally created automatically by BLAST, but they can also be modified by the user. Both annotation types are stored with the indices of the words they apply to.

Figure 1 shows a screenshot of BLAST. The MT output is shown to the annotator one segment at a time, in the upper part of the screen. A segment normally consists of a sentence and the MT output can be accompanied by a source sentence, a reference sentence, or both. Error annotations are marked in the segments by bold, underlined, colored text, and support annotations are marked by light background colors. The bottom part of the tool, contains the error typology, and controls for updating annotations and navigation. The error typology is shown using a menu structure, where submenus are activated by the user clicking on higher levels.

3.1 Design goals

We created BLAST with the goal that it should be flexible, and allow maximum freedom for the user,

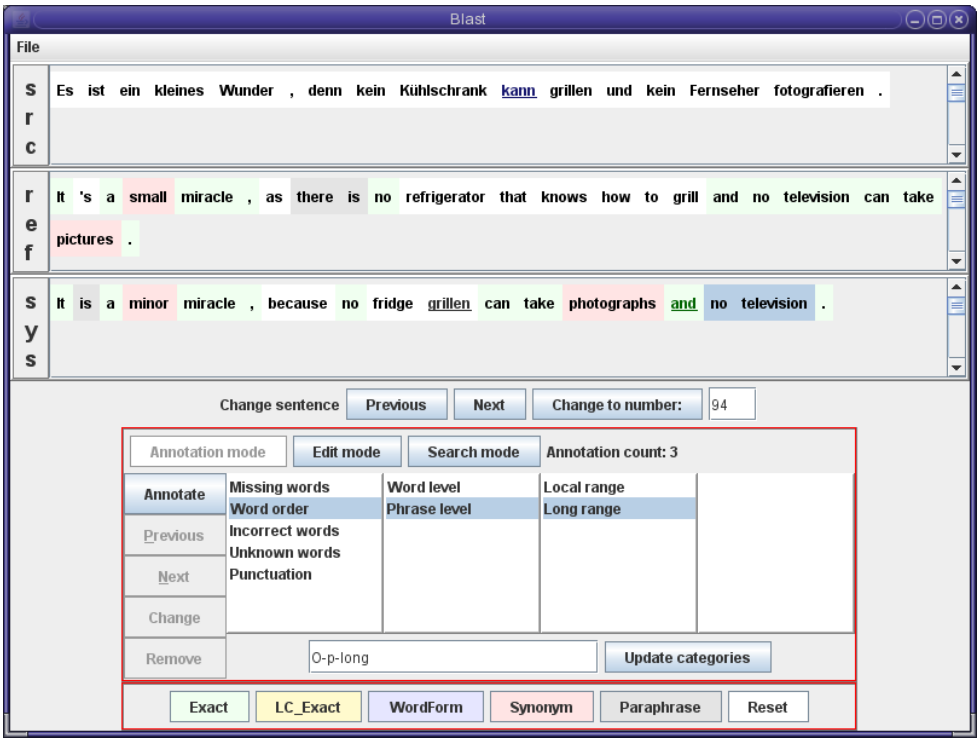


Figure 1: Screenshot of BLAST

based on the following goals:

- Independent of the MT system being analyzed, particularly not dependent on specific information given by a particular MT system, such as alignment information
- Compatible with any error typology
- Language pair independent
- Possible to mark where in a sentence an error occurs
- Possible to view either source or reference sentences, or both
- Possible to automatically highlight similarities between the system and the reference sentences
- Containing a search function for errors
- Simple to understand and use

The current implementation of BLAST fulfils all these goals, with the possible small limitation that the error typology has to be hierarchical. We believe this limitation is minor, however, since it is possible to have a relatively flat structure if desired, and to re-use the same submenu in many places, allowing cross-classification within a hierarchical typology.

The flexibility of the tool gives users a lot of freedom in how to use it in their evaluation projects. However, we believe that it is important within every error annotation project to use a set error typology and guidelines for annotation, but the annotation tool should not limit users in making these choices.

3.2 Error Typologies

As described above, BLAST is easily configurable with new typologies for annotation, with the only restriction that the typology is hierarchical. BLAST currently comes with the following implemented typologies, some of which are general, and some of which are targeted at specific language (pairs):

- Vilar et al. (2006)
  - General
  - Chinese
  - Spanish
- Farrús et al. (2010)
  - Catalan–Spanish
- Flanagan (1994) (slightly modified into a hierarchical structure)
  - French

  - German
- Our own tentative fine-grained typology
  - General
  - Swedish

The error typologies can be very big, and it is hard to fit an arbitrarily large typology into a graphical tool. BLAST thus uses a menu structure which always shows the categories in the first level of the typology. Lower subtypologies are only shown when they are activated by the user clicking on a higher level. In Figure 1, the subtypologies to *Word order* were activated by the user first clicking on *Word order*, then on *Phrase level*.

It is important that typologies are easy to extend and modify, especially in order to cover new target languages, since the translation problems to some extent will be dependent on the target language, for instance with regard to the different agreement phenomena in languages. The typologies that come with BLAST can serve as a starting point for adjusting typologies, especially to new target languages.

3.3 Implementation

BLAST is implemented as a Java application using Swing for the graphical user interface. Using Java makes it platform independent, and it is currently tested on Unix, Linux, Mac, and Windows. BLAST has an object-oriented design, with a particular focus on modular design, to allow it to be easily extendible with new modules for preprocessing, reading and writing to different file formats, and presenting statistics. Unicode is used in order to allow a high number of languages, and sentences can be displayed both right to left, and left to right. BLAST is open source and is released under the LGPL license.[3]

3.4 File formats

The main file types used in BLAST is the annotation file, containing the translation segments and annotations, and the typology file. These files are stored in a simple text file format. There is also a configuration file, which can be used for program settings, besides using command line options, for instance to configure color schemes, and to change preprocessing settings. The statistics of an annotation project are printed in a text file in a human-readable format (see Section 4.5).

The annotation file contains the translation segments for the MT system, and possibly for the source and reference sentences, and all error and support annotations. The annotations are stored with the indices of the word(s) in the segments that were marked, and a label identifying the error type. The annotation file is initially created automatically by BLAST based on sentence aligned files. It is then updated by BLAST with the annotations added by the user.

The typology file has a header with main information, and then an item for each menu containing:

- The name of the menu
- A list of menu items, containing:
  - Display name
  - Internal name (used in annotation file, and internally in BLAST)
  - The name of its submenu (if any)

The typology files have to be specified by the user, but BLAST comes with several typology files, as described in Section 3.2.

4 Working with BLAST

BLAST has three different working modes: annotation, edit and search. The main mode is annotation, which allows the user to add new error annotations. The edit mode allows the user to edit and remove error annotations. The search mode allows the user to search for errors of different types. BLAST can also create support annotations, that can later be updated by the user, and calculate and print statistics of an annotation project.

4.1 Annotation

The annotation mode is the main working mode in BLAST, and it is active in Figure 1. In annotation mode a segment is shown with all its current error annotations. The annotations are marked with bold and colored text, where the color depends on the main type of the error. For each new annotation the user selects the word or words that are wrong, and selects an error type. In figure 1, the words *no television*, and the error type *Word order→Phrase level→Long* are selected in order to add a new error

[3] http://www.gnu.org/copyleft/lesser.html

annotation. BLAST ignores identical annotations, and warns the user if they try to add an annotation for the exact same words as another annotation.

4.2 Edit

In edit mode the user can change existing error annotations. In this mode only one annotation at a time is shown, and the user can switch between them. For each annotation affected words are highlighted, and the error typology area shows the type of the error. The currently shown error can be changed to a different error type, or it can be removed. The edit mode is useful for revising annotations, and for correcting annotation errors.

4.3 Search

In search mode, it is possible to search for errors of a certain type. To search, users choose the error type they want to search for in the error typology, and then search backwards or forwards for error annotations of that type. It is possible both to search for specific errors deep in the typology, and to search for all errors of a type higher in the typology, for instance, to search for all word order errors, regardless of subclassification. Search is active between all segments, not only for the currently shown segment. Search is useful for controlling the consistency of annotations, and for finding instances of specific errors.

4.4 Support annotations

Error annotation is a hard task for humans, and thus we try to aid it by including automatic preprocessing, where similarities between the system and reference sentences are marked at different levels of similarity. Even if the goal of the error analysis often is not to compare the MT output to a single reference, but to the closest correct equivalent, it can still be useful to be able to see the similarities to one reference sentence, to be able to identify problematic parts easier.

For this module we have adapted the code for alignment used in the Meteor-NEXT metric (Denkowski and Lavie, 2010) to BLAST. In Meteor-NEXT the system and reference sentences are aligned at the levels of exact matching, stemmed matching, synonyms, and paraphrases. All these modules work on lower-cased data, so we added a module for exact matching with the original casing kept. The exact and lower-cased matching works for most languages, and stemming for 15 languages. The synonym module uses WordNet, and is only available for English. The paraphrase module is based on an automatic paraphrase induction method (Bannard and Callison-Burch, 2005), it is currently trained for five languages, but the Meteor-NEXT code for training it for additional languages is included.

Support annotations are normally only created automatically, but BLAST allows the user to edit them. The mechanism for adding, removing or changing support annotations is separate from error annotations, and can be used regardless of mode.

4.5 Create Statistics

The statistics module prints statistics about the currently loaded annotation project. The statistics are printed to a file, in a human-readable format. It contains information about the number of sentences and errors in the project, average number of errors per sentence, and how many sentences there are with certain numbers of errors. The main part of the statistics is the number and percentage of errors for each node in the error typology. It is also possible to get the number of errors for cross-classifications, by specifying regular expressions for the categories to cross-classify in the configuration file.

5 Future Extensions

BLAST is under active development, and we plan to add new features. Most importantly we want to add the possibility to annotate two MT systems in parallel, which can be useful if the purpose of the annotation is to compare MT systems. We are also working on refining and developing the existing proposals for error typologies, which is an important complement to the tool itself. We intend to define a new fine-grained general error typology, with extensions to a number of target languages.

The modularity of BLAST also makes it possible to add new modules, for instance for preprocessing and to support other file formats. One example would be to support error annotation of only specific phenomena, such as verb errors, by adding a preprocessing module for highlighting verbs with support

annotations, and a suitable verb-focused error typology. We are also working on a preprocessing module based on grammar checker techniques (Stymne and Ahrenberg, 2010), that highlights parts of the MT output that it suspects are non-grammatical.

Even though the main purpose of BLAST is for error annotation of machine translation output, the freedom in the use of error typologies and support annotations also makes it suitable for other tasks where bilingual material is used, such as for annotations of named entities in bilingual texts, or for analyzing human translations, e.g. giving feedback to second language learners, with only the addition of a suitable typology, and possibly a preprocessing module.

6 Conclusion

We presented BLAST; a flexible tool for annotation of bilingual segments, specifically intended for error analysis of MT. BLAST facilitates the error analysis task, which we believe is vital for MT researchers, and could also be useful for other users of MT. Its flexibility makes it possible to annotate translations from any MT system and between any language pairs, using any hierarchical error typology.

References

Lars Ahrenberg, Magnus Merkel, and Michael Petterstedt. 2003. Interactive word alignment for language engineering. In *Proceedings of EACL*, pages 49–52, Budapest, Hungary.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604, Ann Arbor, Michigan, USA.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Proceedings of WMT*, pages 136–158, Prague, Czech Republic, June.

Michael Denkowski and Alon Lavie. 2010. METEOR-NEXT and the METEOR paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of WMT and MetricsMATR*, pages 339–342, Uppsala, Sweden.

Mireia Farrús, Marta R. Costa-jussà, José B. Mariño, and José A. R. Fonollosa. 2010. Linguistic-based evaluation criteria to identify statistical machine translation errors. In *Proceedings of EAMT*, pages 52–57, Saint Raphaël, France.

Mary Flanagan. 1994. Error classification for MT evaluation. In *Proceedings of AMTA*, pages 65–72, Columbia, Maryland, USA.

Ariadna Font Llitjós and Jaime Carbonell. 2004. The translation correction tool: English-Spanish user studies. In *Proceedings of LREC*, pages 347–350, Lisbon, Portugal.

Meghan Lammie Glenn, Stephanie Strassel, Lauren Friedman, and Haejoong Lee. 2008. Management of large annotation projects involving multiple human judges: a case study of GALE machine translation post-editing. In *Proceedings of LREC*, pages 2957–2960, Marrakech, Morocco.

Eduard Hovy, Margaret King, and Andrei Popescu-Belis. 2002. Principles of context-based machine translation evaluation. *Machine Translation*, 17(1):43–75.

Masaki Murata, Kiyotaka Uchimoto, Qing Ma, Toshiyuki Kanamaru, and Hitoshi Isahara. 2005. Analysis of machine translation systems' errors in tense, aspect, and modality. In *Proceedings of PACLIC 19*, pages 155–166, Taipei, Taiwan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Maja Popović, Adrià de Gisper, Deepa Gupta, Patrik Lambert, Hermann Ney, José Mariño, and Rafael Banchs. 2006. Morpho-syntactic information for automatic error analysis of statistical machine translation output. In *Proceedings of WMT*, pages 1–6, New York City, New York, USA.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human notation. In *Proceedings of AMTA*, pages 223–231, Cambridge, Massachusetts, USA.

Sara Stymne and Lars Ahrenberg. 2010. Using a grammar checker for evaluation and postprocessing of statistical machine translation. In *Proceedings of LREC*, pages 2175–2181, Valetta, Malta.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error analysis of machine translation output. In *Proceedings of LREC*, pages 697–702, Genoa, Italy.

Prototyping virtual instructors from human-human corpora

Luciana Benotti
PLN Group, FAMAF
National University of Córdoba
Córdoba, Argentina
luciana.benotti@gmail.com

Alexandre Denis
TALARIS team, LORIA/CNRS
Lorraine. Campus scientifique, BP 239
Vandoeuvre-lès-Nancy, France
alexandre.denis@loria.fr

Abstract

Virtual instructors can be used in several applications, ranging from trainers in simulated worlds to non player characters for virtual games. In this paper we present a novel algorithm for rapidly prototyping virtual instructors from human-human corpora without manual annotation. Automatically prototyping full-fledged dialogue systems from corpora is far from being a reality nowadays. Our algorithm is restricted in that only the virtual instructor can perform speech acts while the user responses are limited to physical actions in the virtual world. We evaluate a virtual instructor, generated using this algorithm, with human users. We compare our results both with human instructors and rule-based virtual instructors hand-coded for the same task.

1 Introduction

Virtual human characters constitute a promising contribution to many fields, including simulation, training and interactive games (Kenny et al., 2007; Jan et al., 2009). The ability to communicate using natural language is important for believable and effective virtual humans. Such ability has to be good enough to engage the trainee or the gamer in the activity. Nowadays, most conversational systems operate on a dialogue-act level and require extensive annotation efforts in order to be fit for their task (Rieser and Lemon, 2010). Semantic annotation and rule authoring have long been known as bottlenecks for developing conversational systems for new domains.

In this paper, we present novel a algorithm for generating virtual instructors from automatically annotated human-human corpora. Our algorithm, when given a task-based corpus situated in a virtual world, generates an instructor that robustly helps a user achieve a given task in the virtual world of the corpus. There are two main approaches toward automatically producing dialogue utterances. One is the selection approach, in which the task is to pick the appropriate output from a corpus of possible outputs. The other is the generation approach, in which the output is dynamically assembled using some composition procedure, e.g. grammar rules. The selection approach to generation has only been used in conversational systems that are not task-oriented such as negotiating agents (Gandhe and Traum, 2007), question answering characters (Kenny et al., 2007), and virtual patients (Leuski et al., 2006). Our algorithm can be seen as a novel way of doing robust generation by selection and interaction management for task-oriented systems.

In the next section we introduce the corpora used in this paper. Section 3 presents the two phases of our algorithm, namely automatic annotation and dialogue management through selection. In Section 4 we present a fragment of an interaction with a virtual instructor generated using the corpus and the algorithm introduced in the previous sections. We evaluate the virtual instructor in interactions with human subjects using objective as well as subjective metrics. We present the results of the evaluation in Section 5. We compare our results with both human and rule-based virtual instructors hand-coded for the same task. Finally, Section 6 concludes the paper proposing an improved virtual instructor designed as a result of our error analysis.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 62–67,
Portland, Oregon, USA, 21 June 2011.

2 The GIVE corpus

The Challenge on Generating Instructions in Virtual Environments (GIVE; Koller et al. (2010)) is a shared task in which Natural Language Generation systems must generate real-time instructions that guide a user in a virtual world. In this paper, we use the GIVE-2 Corpus (Gargett et al., 2010), a corpus of human instruction giving in virtual environments. We use the English part of the corpus which consists of 63 American English written discourses in which one subject guided another in a treasure hunting task in 3 different 3D worlds.

The task setup involved pairs of human partners, each of whom played one of two different roles. The "direction follower" (DF) moved about in the virtual world with the goal of completing a treasure hunting task, but had no knowledge of the map of the world or the specific behavior of objects within that world (such as, which buttons to press to open doors). The other partner acted as the "direction giver" (DG), who was given complete knowledge of the world and had to give instructions to the DF to guide him/her to accomplish the task.

The GIVE-2 corpus is a multimodal corpus which consists of all the instructions uttered by the DG, and all the object manipulations done by the DF with the corresponding timestamp. Furthermore, the DF's position and orientation is logged every 200 milliseconds, making it possible to extract information about his/her movements.

3 The unsupervised conversational model

Our algorithm consists of two phases: an annotation phase and a selection phase. The *annotation phase* is performed only once and consists of automatically associating the DG instruction to the DF reaction. The *selection phase* is performed every time the virtual instructor generates an instruction and consists of picking out from the annotated corpus the most appropriate instruction at a given point.

3.1 The automatic annotation

The basic idea of the annotation is straightforward: associate each *utterance* with its corresponding *reaction*. We assume that a reaction captures the semantics of its associated instruction. Defining reaction involves two subtle issues, namely *boundary* determination and *discretization*. We discuss these issues in turn and then give a formal definition of reaction.

We define the *boundaries* of a reaction as follows. A reaction r_k to an instruction u_k begins right after the instruction u_k is uttered and ends right before the next instruction u_{k+1} is uttered. In the following example, instruction 1 corresponds to the reaction $\langle 2, 3, 4 \rangle$, instruction 5 corresponds to $\langle 6 \rangle$, and instruction 7 to $\langle 8 \rangle$.

DG(1): hit the red you see in the far room
DF(2): [enters the far room]
DF(3): [pushes the red button]
DF(4): [turns right]
DG(5): hit far side green
DF(6): [moves next to the wrong green]
DG(7): no
DF(8): [moves to the right green and pushes it]

As the example shows, our definition of boundaries is not always semantically correct. For instance, it can be argued that it includes too much because 4 is not strictly part of the semantics of 1. Furthermore, misinterpreted instructions (as 5) and corrections (e.g., 7) result in clearly inappropriate instruction-reaction associations. Since we want to avoid any manual annotation, we decided to use this naive definition of boundaries anyway. We discuss in Section 5 the impact that inappropriate associations have on the performance of a virtual instructor.

The second issue that we address here is *discretization* of the reaction. It is well known that there is not a unique way to discretize an action into subactions. For example, we could decompose action 2 into 'enter the room' or into 'get close to the door and pass the door'. Our algorithm is not dependent on a particular discretization. However, the same discretization mechanism used for annotation has to be used during selection, for the dialogue manager to work properly. For selection (i.e., in order to decide what to say next) any virtual instructor needs to have a *planner* and a *planning domain representation*, i.e., a specification of how the virtual world works and a way to represent the state of the virtual world. Therefore, we decided to use them in order to discretize the reaction.

Now we are ready to define *reaction* formally. Let S_k be the state of the virtual world when uttering in-

struction u_k, S_{k+1} be the state of the world when uttering the next utterance u_{k+1} and D be the planning domain representation. The *reaction* to u_k is defined as the sequence of actions returned by the planner with S_k as initial state, S_{k+1} as goal state and D as planning domain.

The annotation of the corpus then consists of automatically associating each utterance to its (discretized) reaction.

3.2 Selecting what to say next

In this section we describe how the selection phase is performed every time the virtual instructor generates an instruction.

The instruction selection algorithm consists in finding in the corpus the set of candidate utterances C for the current task plan P; P being the sequence of actions returned by the same planner and planning domain used for discretization. We define $C = \{U \in \text{Corpus} \mid U.\textit{Reaction}$ is a prefix of $P\}$. In other words, an utterance U belongs to C if the first actions of the current plan P exactly match the reaction associated to the utterance. All the utterances that pass this test are considered paraphrases and hence suitable in the current context.

While P does not change, the virtual instructor iterates through the set C, verbalizing a different utterance at fixed time intervals (e.g., every 3 seconds). In other words, the virtual instructor offers alternative paraphrases of the intended instruction. When P changes as a result of the actions of the DF, C is recalculated.

It is important to notice that the discretization used for annotation and selection directly impacts the behavior of the virtual instructor. It is crucial then to find an appropriate granularity of the discretization. If the granularity is too coarse, many instructions in the corpus will have an empty associated reaction. For instance, in the absence of the representation of the user orientation in the planning domain (as is the case for the virtual instructor we evaluate in Section 5), instructions like "turn left" and "turn right" will have empty reactions making them indistinguishable during selection. However, if the granularity is too fine the user may get into situations that do not occur in the corpus, causing the selection algorithm to return an empty set of candidate utterances. It is the responsibility of the virtual instructor developer to find a granularity sufficient to capture the diversity of the instructions he wants to distinguish during selection.

4 A virtual instructor for a virtual world

We implemented an English virtual instructor for one of the worlds used in the corpus collection we presented in Section 2. The English fragment of the corpus that we used has 21 interactions and a total of 1136 instructions. Games consisted on average of 54.2 instructions from the human DG, and took about 543 seconds on average for the human DF to complete the task.

On Figures 1 to 4 we show an excerpt of an interaction between the system and a real user that we collected during the evaluation. The figures show a 2D map from top view and the 3D in-game view. In Figure 1, the user, represented by a blue character, has just entered the upper left room. He has to push the button close to the chair. The first candidate utterance selected is "red closest to the chair in front of you". Notice that the referring expression uniquely identifies the target object using the spatial proximity of the target to the chair. This referring expression is generated without any reasoning on the target distractors, just by considering the current state of the task plan and the user position.

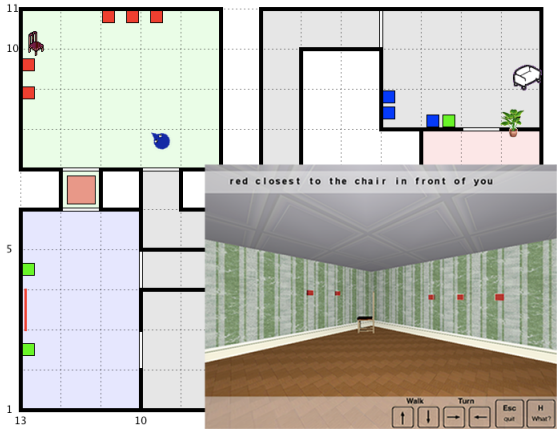


Figure 1: "red closest to the chair in front of you"

After receiving the instruction the user gets closer to the button as shown in Figure 2. As a result of the new user position, a new task plan exists, the set of candidate utterances is recalculated and the system selects a new utterance, namely "the closet one".

The generation of the ellipsis of the button or the

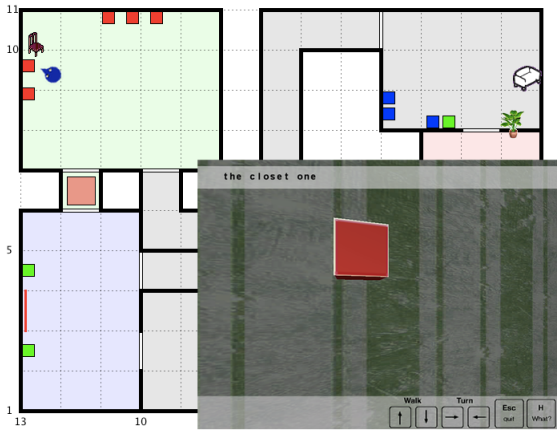


Figure 2: “the closet one”

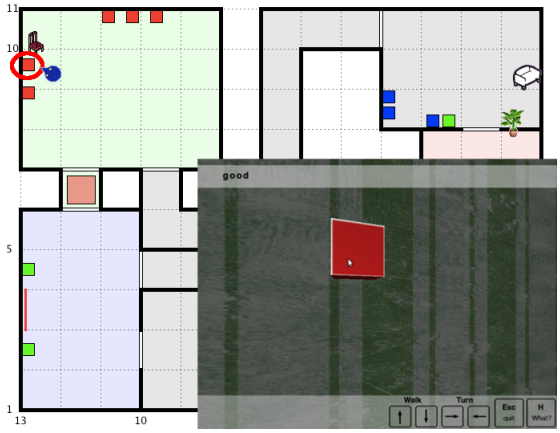


Figure 3: “good”

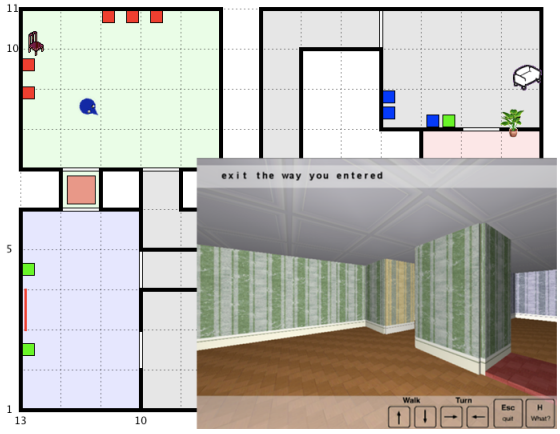


Figure 4: “exit the way you entered”

chair is a direct consequence of the utterances normally said in the corpus at this stage of the task plan (that is, when the user is about to manipulate this object). From the point of view of referring expression algorithms, the referring expression may not be optimal because it is over-specified (a pronoun would be preferred as in “click it”), Furthermore, the instruction contains a spelling error (‘closet’ instead of ‘closest’). In spite of this non optimality, the instruction led our user to execute the intended reaction, namely pushing the button.

Right after the user clicks on the button (Figure 3), the system selects an utterance corresponding to the new task plan. The player position stayed the same so the only change in the plan is that the button no longer needs to be pushed. In this task state, DGs usually give acknowledgements and this then what our selection algorithm selects: “good”.

After receiving the acknowledgement, the user turns around and walks forward, and the next action in the plan is to leave the room (Figure 4). The system selects the utterance “exit the way you entered” which refers to the previous interaction. Again, the system keeps no representation of the past actions of the user, but such utterances are the ones that are found at this stage of the task plan.

5 Evaluation and error analysis

In this section we present the results of the evaluation we carried out on the virtual instructor presented in Section 4 which was generated using the dialogue model algorithm introduced in Section 3.

We collected data from 13 subjects. The participants were mostly graduate students; 7 female and 6 male. They were not English native speakers but rated their English skills as near-native or very good.

The evaluation contains both objective measures which we discuss in Section 5.1 and subjective measures which we discuss in Section 5.2.

5.1 Objective metrics

The objective metrics we extracted from the logs of interaction are summarized in Table 1. The table compares our results with both human instructors and the three rule-based virtual instructors that were top rated in the GIVE-2 Challenge. Their results correspond to those published in (Koller et al., 2010) which were collected not in a laboratory but connecting the systems to users over the Internet. These hand-coded systems are called NA, NM and Saar. We refer to our system as OUR.

	Human	NA	Saar	NM	OUR
Task success	100%	47%	40%	30%	70%
Canceled	0%	24%	n/a	35%	7%
Lost	0%	29%	n/a	35%	23%
Time (sec)	543	344	467	435	692
Mouse actions	12	17	17	18	14
Utterances	53	224	244	244	194

Table 1: Results for the *objective* metrics

In the table we show the percentage of games that users completed successfully with the different instructors. Unsuccessful games can be either canceled or lost. To ensure comparability, time until task completion, number of instructions received by users, and mouse actions are only counted on successfully completed games.

In terms of task success, our system performs better than all hand-coded systems. We duly notice that, for the GIVE Challenge in particular (and probably for human evaluations in general) the success rates in the laboratory tend to be higher than the success rate online (this is also the case for completion times) (Koller et al., 2009).

In any case, our results are preliminary given the amount of subjects that we tested (13 versus around 290 for GIVE-2), but they are indeed encouraging. In particular, our system helped users to identify better the objects that they needed to manipulate in the virtual world, as shown by the low number of mouse actions required to complete the task (a high number indicates that the user must have manipulated wrong objects). This correlates with the subjective evaluation of referring expression quality (see next section).

We performed a detailed analysis of the instructions uttered by our system that were unsuccessful, that is, all the instructions that did not cause the intended reaction as annotated in the corpus. From the 2081 instructions uttered in the 13 interactions, 1304 (63%) of them were successful and 777 (37%) were unsuccessful.

Given the limitations of the annotation discussed in Section 3.1 (wrong annotation of correction utterances and no representation of user orientation) we classified the unsuccessful utterances using lexical cues into 1) correction ('no','don't','keep', etc.), 2) orientation instruction ('left', 'straight', 'behind', etc.) and 3) other. We found that 25% of the unsuccessful utterances are of type 1, 40% are type 2, 34% are type 3 (1% corresponds to the default utterance "go" that our system utters when the set of candidate utterances is empty). Frequently, these errors led to contradictions confusing the player and significantly affecting the completion time of the task as shown in Table 1. In Section 6 we propose an improved virtual instructor designed as a result of this error analysis.

5.2 Subjective metrics

The subjective measures were obtained from responses to the GIVE-2 questionnaire that was presented to users after each game. It asked users to rate different statements about the system using a continuous slider. The slider position was translated to a number between -100 and 100. As done in GIVE-2, for negative statements, we report the reversed scores, so that in Tables 2 and 3 greater numbers are always better. In this section we compare our results with the systems NA, Saar and NM as we did in Section 5.1, we cannot compare against human instructors because these subjective metrics were not collected in (Gargett et al., 2010).

The GIVE-2 Challenge questionnaire includes twenty-two subjective metrics. Metrics Q1 to Q13 and Q22 assess the effectiveness and reliability of instructions. For almost all of these metrics we got similar or slightly lower results than those obtained by the three hand-coded systems, except for three metrics which we show in Table 2. We suspect that the low results obtained for Q5 and Q22 relate to the unsuccessful utterances identified and discussed in Section 5.1. The high unexpected result in Q6 is probably correlated with the low number of mouse actions mentioned in Section 5.1.

NA	Saar	NM	OUR
Q5: I was confused about which direction to go in			
29	5	9	-12
Q6: I had no difficulty with identifying the objects the system described for me			
18	20	13	40
Q22: I felt I could trust the system's instructions			
37	21	23	0

Table 2: Results for the *subjective* measures assessing the efficiency and effectiveness of the instructions

Metrics Q14 to Q20 are intended to assess the nat-

uralness of the instructions, as well as the immersion and engagement of the interaction. As Table 3 shows, in spite of the unsuccessful utterances, our system is rated as more natural and more engaging (in general) than the best systems that competed in the GIVE-2 Challenge.

NA	Saar	NM	OUR
Q14: The system's instructions sounded robotic			
-4	5	-1	28
Q15: The system's instructions were repetitive			
-31	-26	-28	-8
Q16: I really wanted to find that trophy			
-11	-7	-8	7
Q17: I lost track of time while solving the task			
-16	-11	-18	16
Q18: I enjoyed solving the task			
-8	-5	-4	4
Q19: Interacting with the system was really annoying			
8	-2	-2	4
Q20: I would recommend this game to a friend			
-30	-25	-24	-28

Table 3: Results for the *subjective* measures assessing the naturalness and engagement of the instructions

6 Conclusions and future work

In this paper we presented a novel algorithm for rapidly prototyping virtual instructors from human-human corpora without manual annotation. Using our algorithm and the GIVE corpus we have generated a virtual instructor[1] for a game-like virtual environment. We obtained encouraging results in the evaluation with human users that we did on the virtual instructor. Our system outperforms rule-based virtual instructors hand-coded for the same task both in terms of objective and subjective metrics. It is important to mention that the GIVE-2 hand-coded systems do not need a corpus but are tightly linked to the GIVE task. Our algorithm requires human-human corpora collected on the target task and environment, but it is independent of the particular instruction giving task. For instance, it could be used for implementing game tutorials, real world navigation systems or task-based language teaching.

In the near future we plan to build a new version of the system that improves based on the error analysis that we did. For instance, we plan to change our discretization mechanism in order to take orientation into account. This is supported by our algorithm although we may need to enlarge the corpus we used so as not to increase the number of situations in which the system does not find anything to say. Finally, if we could identify corrections automatically, as suggested in (Raux and Nakano, 2010), we could get another increase in performance, because we would be able to treat them as corrections and not as instructions as we do now.

In sum, this paper presents a novel way of automatically prototyping task-oriented virtual agents from corpora who are able to effectively and naturally help a user complete a task in a virtual world.

[1]Demo at `cs.famaf.unc.edu.ar/~luciana/give-OUR`

References

Sudeep Gandhe and David Traum. 2007. Creating spoken dialogue characters from corpora without annotations. In *Proceedings of Interspeech*, Belgium.

Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. 2010. The GIVE-2 corpus of giving instructions in virtual environments. In *Proc. of the LREC*, Malta.

Dusan Jan, Antonio Roque, Anton Leuski, Jacki Morie, and David Traum. 2009. A virtual tour guide for virtual worlds. In *Proc. of IVA*, pages 372–378, The Netherlands. Springer-Verlag.

Patrick Kenny, Thomas D. Parsons, Jonathan Gratch, Anton Leuski, and Albert A. Rizzo. 2007. Virtual patients for clinical therapist skills training. In *Proc. of IVA*, pages 197–210, France. Springer-Verlag.

Alexander Koller, Kristina Striegnitz, Donna Byron, Justine Cassell, Robert Dale, Sara Dalzel-Job, Johanna Moore, and Jon Oberlander. 2009. Validating the web-based evaluation of nlg systems. In *Proc. of ACL-IJCNLP*, Singapore.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second challenge on generating instructions in virtual environments (GIVE-2). In *Proc. of INLG*, Dublin.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2006. Building effective question answering characters. In *Proc. of SIGDIAL*, pages 18–27, Australia. ACL.

Antoine Raux and Mikio Nakano. 2010. The dynamics of action corrections in situated interaction. In *Proc. of SIGDIAL*, pages 165–174, Japan. ACL.

Verena Rieser and Oliver Lemon. 2010. Learning human multimodal dialogue strategies. *Natural Language Engineering*, 16:3–23.

An Interactive Machine Translation System with Online Learning

Daniel Ortiz-Martínez, Luis A. Leiva, Vicent Alabau,
Ismael García-Varea†, Francisco Casacuberta
ITI - Institut Tecnològic d'Informàtica, Universitat Politècnica de València
† Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha
{dortiz,luileito,valabau,fcn}@iti.upv.es, †ismael.garcia@uclm.es

Abstract

State-of-the-art Machine Translation (MT) systems are still far from being perfect. An alternative is the so-called Interactive Machine Translation (IMT) framework, where the knowledge of a human translator is combined with the MT system. We present a statistical IMT system able to learn from user feedback by means of the application of online learning techniques. These techniques allow the MT system to update the parameters of the underlying models in real time. According to empirical results, our system outperforms the results of conventional IMT systems. To the best of our knowledge, this online learning capability has never been provided by previous IMT systems. Our IMT system is implemented in C++, JavaScript, and ActionScript; and is publicly available on the Web.

1 Introduction

The research in the field of machine translation (MT) aims to develop computer systems which are able to translate text or speech without human intervention. However, current translation technology has not been able to deliver full automated high-quality translations. Typical solutions to improve the quality of the translations supplied by an MT system require manual post-editing. This serial process prevents the MT system from integrating the knowledge of the human translator.

An alternative way to take advantage of the existing MT technologies is to use them in collaboration with human translators within a computer-assisted translation (CAT) or interactive framework (Isabelle and Church, 1997). Interactivity in CAT has been explored for a long time. Systems have been designed to interact with linguists to solve ambiguities or update user dictionaries.

An important contribution to CAT technology was pioneered by the *TransType* project (Foster et al., 1997; Langlais et al., 2002). The idea proposed in that work was to embed data driven MT techniques within the interactive translation environment. Following the TransType ideas, Barrachina et al. (2009) proposed the so-called IMT framework, in which fully-fledged statistical MT (SMT) systems are used to produce full target sentences hypotheses, or portions thereof, which can be accepted or amended by a human translator. Each corrected text segment is then used by the MT system as additional information to achieve improved suggestions. Figure 1 shows an example of a typical IMT session.

The vast majority of the existing work on IMT makes use of the well-known *batch learning* paradigm. In the batch learning paradigm, the training of the IMT system and the interactive translation process are carried out in separate stages. This paradigm is not able to take advantage of the new knowledge produced by the user of the IMT system. In this paper, we present an application of the *online learning* paradigm to the IMT framework. In the online learning paradigm, the training and prediction stages are no longer separated. This feature is particularly useful in IMT since it allows to take into account the user feedback. Specifically, our proposed IMT system can be extended with the new training samples that are generated each time the user validates the translation of a given source sentence. The online learning techniques implemented in our IMT system incrementally update the statistical models involved in the translation process.

2 Related work

There are some works on IMT in the literature that try to take advantage of user feedback. One example is the work by Nepveu et al. (2004), where dynamic adaptation of an IMT system via cache-based model extensions to language and translation models is proposed. One major drawback of such proposal is its inability to learn new words.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 68–73,
Portland, Oregon, USA, 21 June 2011.

source($\mathbf{f}$):		Para ver la lista de recursos
reference($\hat{\mathbf{e}}$):		To view a listing of resources
interaction-0	$\mathbf{e}_p$	
	$\mathbf{e}_s$	*To view the resources list*
interaction-1	$\mathbf{e}_p$	To view
	k	**a**
	$\mathbf{e}_s$	*list of resources*
interaction-2	$\mathbf{e}_p$	To view a list
	k	**i**
	$\mathbf{e}_s$	*ng resources*
interaction-3	$\mathbf{e}_p$	To view a listing
	k	**o**
	$\mathbf{e}_s$	*f resources*
accept	$\mathbf{e}_p$	To view a listing of resources

Figure 1: IMT session to translate a Spanish sentence into English. In interaction-0, the system suggests a translation ($\mathbf{e}_s$). In interaction-1, the user moves the mouse to accept the first eight characters "To view " and presses the **a** key (k), then the system suggests completing the sentence with "*list of resources*" (a new $\mathbf{e}_s$). Interactions 2 and 3 are similar. In the final interaction, the user accepts the current suggestion.

Recent research on IMT has proposed the use of online learning as one possible way to successfully incorporate user feedback in IMT systems (Ortiz-Martínez et al., 2010). In the online learning setting, models are trained sample by sample. For this reason, such learning paradigm is appropriate for its use in the IMT framework. The work by Ortiz-Martínez et al. (2010) implements online learning as incremental learning. Specifically, an IMT system able to incrementally update the parameters of all of the different models involved in the interactive translation process is proposed. One previous attempt to implement online learning in IMT is the work by Cesa-Bianchi et al. (2008). In that work, the authors present a very constrained version of online learning, which is not able to extend the translation models due to the high time cost of the learning process.

We have adopted the online learning techniques proposed in (Ortiz-Martínez et al., 2010) to implement our IMT system. We are not aware of other IMT tools that include such functionality. For instance, a prototype system for text prediction to help translators is shown in (Foster et al., 2002). Additionally, Koehn (2009) presents the *Caitra* translation tool. *Caitra* aids linguists suggesting sentence completions, alternative words or allowing users to post-edit machine translation output. However, neither of these systems are able to take advantage of the user validated translations.

3 Interactive Machine Translation

IMT can be seen as an evolution of the statistical machine translation (SMT) framework. In SMT, given source string $\mathbf{f}$, we seek for the target string $\mathbf{e}$ which maximizes the posterior probability:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{e}|\mathbf{f}) \tag{1}$$

Within the IMT framework, a state-of-the-art SMT system is employed in the following way. For a given source sentence, the SMT system automatically generates an initial translation. A human translator checks this translation from left to right, correcting the first error. The SMT system then proposes a new extension, taking the correct prefix $\mathbf{e}_p$ into account. These steps are repeated until the whole input sentence has been correctly translated. In the resulting decision rule, we maximize over all possible extensions $\mathbf{e}_s$ of $\mathbf{e}_p$:

$$\hat{\mathbf{e}}_s = \underset{\mathbf{e}_s}{\operatorname{argmax}}\, Pr(\mathbf{e}_s|\mathbf{e}_p, \mathbf{f}) \tag{2}$$

It is worth to note that the user interactions are at character level, that is, for each submitted keystroke the system provides a new extension (or suffix) to the current hypothesis. A typical IMT session for a given source sentence is depicted in Figure 1.

State-of-the-art SMT systems follow a log-linear approach (Och and Ney, 2002), where the posterior

probability $Pr(\mathbf{e}\,|\,\mathbf{f})$ of Eq. (1) is used. Such log-linear approach can be easily adapted for its use in the IMT framework as follows:

$$\hat{\mathbf{e}}_s = \underset{\mathbf{e}_s}{\operatorname{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}_p, \mathbf{e}_s, \mathbf{f}) \right\} \quad (3)$$

where each $h_m(\mathbf{e}_p, \mathbf{e}_s, \mathbf{f})$ is a feature function representing a statistical model and λ_m its corresponding weight. Typically, a set of statistical generative models are used as feature functions. Among this feature functions, the most relevant are the language and translation models. The language model is implemented using statistical n-gram language models and the translation model is implemented using phrase-based models.

The IMT system proposed here is based on a log-linear SMT system which includes a total of seven feature functions: an n-gram language model, a target sentence length model, inverse and direct phrase-based models, source and target phrase length models and a reordering model.

4 Online Learning

In the online learning paradigm, *learning* proceeds as a sequence of trials. In each trial, a sample is presented to the learning algorithm to be classified. Once the sample is classified, its correct label is told to the learning algorithm.

The online learning paradigm fits nicely in the IMT framework, since the interactive translation of the source sentences generates new user-validated training samples that can be used to extend the statistical models involved in the translation process.

One key aspect in online learning is the time required by the learning algorithm to process the new training samples. One way to satisfy this constraint is to obtain incrementally updateable versions of the algorithms that are executed to train the statistical models involved in the translation process. We have adopted this approach to implement our IMT system. Specifically, our proposed IMT system implements the set of training algorithms that are required to incrementally update each component of the log-linear model. Such log-linear model is composed of seven components (see section 3). One key aspect of the required training algorithms is the necessity to replace the conventional expectation-maximization (EM) algorithm by its incremental version (Neal and Hinton, 1998). The complete details can be found in (Ortiz-Martínez et al., 2010).

5 System Overview

In this section the main features of our prototype are shown, including prototype design, interaction protocol, prototype functionalities and demo usage.

5.1 Prototype Design

Prototype architecture has been built on two main aspects, namely, accessibility and flexibility. The former is necessary to reach a larger number of potential users. The latter allows researchers to test different techniques and interaction protocols.

For that reason, we developed an CAT Application Programming Interface (API) between the client and the actual translation engine, by using a network communication protocol and exposing a well-defined set of functions.

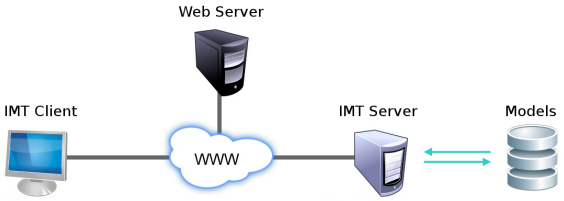


Figure 2: IMT system architecture.

A diagram of the architecture is shown in Figure 2. On the one hand, the IMT client provides a User Interface (UI) which uses the API to communicate with the IMT server through the Web. The hardware requirements in the client are very low, as the translation process is carried out remotely on the server, so virtually any computer (including netbooks, tablets or 3G mobile phones) should be fairly enough. On the other hand, the server, which is unaware of the implementation details of the IMT client, uses and adapts the statistical models that are used to perform the translation.

5.2 User Interaction Protocol

The protocol that rules the IMT process has the following steps:

1. The system proposes a full translation of the selected text segment.

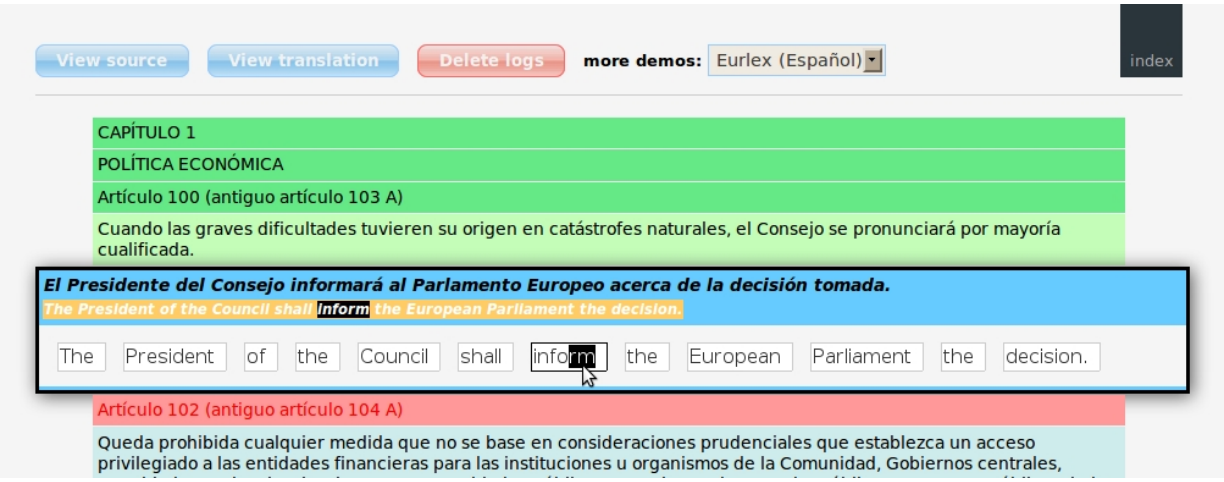


Figure 3: Demo interface. The source text segments are automatically extracted from source document. Such segments are marked as pending (light blue), validated (dark green), partially translated (light green), and locked (light red). The translation engine can work either at full-word or character level.

2. The user validates the longest prefix of the translation which is error-free and/or corrects the first error in the suffix. Corrections are entered by amendment keystrokes or mouse clicks/wheel operations.
3. In this way, a new extended consolidated prefix is produced based on the previous validated prefix and the interaction amendments. Using this new prefix, the system suggests a suitable continuation of it.
4. Steps 2 and 3 are iterated until the user-desired translation is produced.
5. The system adapts the models to the new validated pair of sentences.

5.3 Prototype Functionality

The following is a list of the main features that the prototype supports:

- When the user corrects the solution proposed by the system, a new improved suffix is presented to the user.
- The system is able to *learn* from user-validated translations.
- The user is able to perform *actions* by means of keyboard shortcuts or mouse gestures. The supported actions on the proposed suffix are:

  Substitution Substitute the first word or character of the suffix.

  Deletion Delete the first word of the suffix.

  Insertion Insert a word before the suffix.

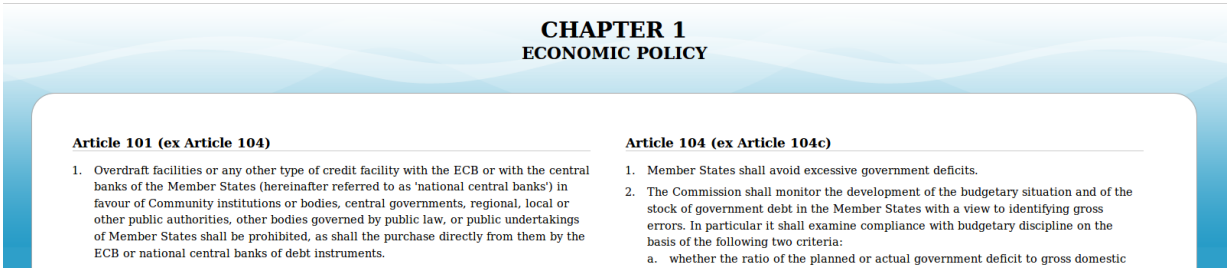
  Rejection The rejected word will not appear in the following proposals.

  Acceptance Assume that the current translation is correct and *adapt* the models.
- At any time, the user is able to visualize the original document (Figure 4(a)), as well as a properly formated draft of the current translation (Figure 4(b)).
- Users can select the document to be translated from a list or upload their own documents.

5.4 Demo Description and Usage

This demo exploits the WWW to enable the connection of simultaneous accesses across the globe, coordinating client-side scripting with server-side technologies. The interface uses web technologies such as XHTML, JavaScript, and ActionScript; while the IMT engine is written in C++.

The prototype is publicly available on the Web (`http://cat.iti.upv.es/imt/`). To begin with, the UI loads an index of all available translation corpora. Currently, the prototype can be tested with the well-known Europarl corpora (Koehn, 2005). The user chooses a corpus and navigates to the main interface page (Figure 3), where she interactively translates the text segments one by one. User's feedback is then processed by the IMT server.



CHAPTER 1
ECONOMIC POLICY

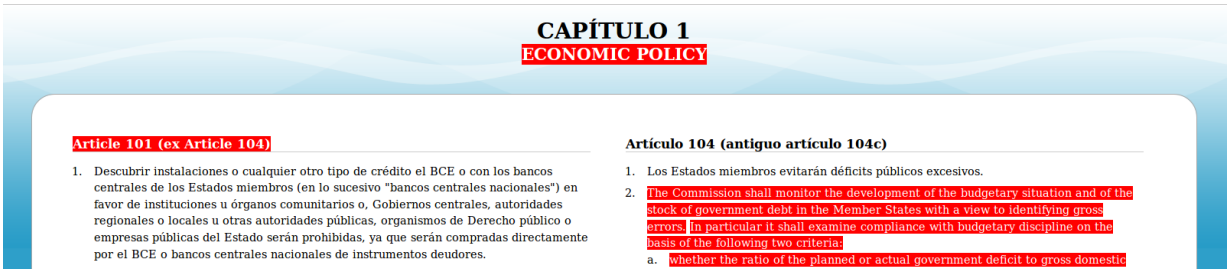
Article 101 (ex Article 104)

1. Overdraft facilities or any other type of credit facility with the ECB or with the central banks of the Member States (hereinafter referred to as 'national central banks') in favour of Community institutions or bodies, central governments, regional, local or other public authorities, other bodies governed by public law, or public undertakings of Member States shall be prohibited, as shall the purchase directly from them by the ECB or national central banks of debt instruments.

Article 104 (ex Article 104c)

1. Member States shall avoid excessive government deficits.
2. The Commission shall monitor the development of the budgetary situation and of the stock of government debt in the Member States with a view to identifying gross errors. In particular it shall examine compliance with budgetary discipline on the basis of the following two criteria:
   a. whether the ratio of the planned or actual government deficit to gross domestic

(a) Source document example, created from EuroParl corpus.



CAPÍTULO 1
ECONOMIC POLICY

Article 101 (ex Article 104)

1. Descubrir instalaciones o cualquier otro tipo de crédito el BCE o con los bancos centrales de los Estados miembros (en lo sucesivo "bancos centrales nacionales") en favor de instituciones u órganos comunitarios o, Gobiernos centrales, autoridades regionales o locales u otras autoridades públicas, organismos de Derecho público o empresas públicas del Estado serán prohibidas, ya que serán compradas directamente por el BCE o bancos centrales nacionales de instrumentos deudores.

Artículo 104 (antiguo artículo 104c)

1. Los Estados miembros evitarán déficits públicos excesivos.
2. The Commission shall monitor the development of the budgetary situation and of the stock of government debt in the Member States with a view to identifying gross errors. In particular it shall examine compliance with budgetary discipline on the basis of the following two criteria:
   a. whether the ratio of the planned or actual government deficit to gross domestic

(b) Translated example document, preserving original format and highlighting non-translated sentences.

Figure 4: Translating documents with the proposed system.

All corrections are stored in plain text logs on the server, so the user can retake them in any moment, also allowing collaborative translations between users. On the other hand, this prototype allows uploading custom documents in text format.

Since the users operate within a web browser, the system also provides crossplatform compatibility and requires neither computational power nor disk space on the client's machine. The communication between application and web server is based on asynchronous HTTP connections, providing thus a richer interactive experience (no page refreshes are required.) Moreover, the Web server communicates with the IMT engine through binary TCP sockets, ensuring really fast response times.

6 Experimental Results

Experimental results were carried out using the Xerox corpus (Barrachina et al., 2009), which consists of translation of Xerox printer manual involving three different language pairs: French-English, Spanish-English, and German-English. This corpus has been extensively used in the literature to report IMT results. The corpus consists of approximately 50,000 sentences pairs for training, 1,000 for development, and 1,000 for test.

The evaluation criteria used in the experiments are the *key-stroke and mouse-action ratio* (KSMR) metric (Barrachina et al., 2009), which measures the user effort required to generate error-free translations, and the well-known BLEU score, which constitutes a measure of the translation quality.

The test corpora were interactively translated from English to the other three languages, comparing the performance of a batch IMT (baseline) and the online IMT systems. The batch IMT system is a conventional IMT system which is not able to take advantage of user feedback after each translation is performed. The online IMT system uses the translations validated by the user to adapt the translation models at runtime. Both systems were initialized with a log-linear model trained in batch mode using the training corpus. Table 1 shows the BLEU score and the KSMR for the batch and the online IMT systems (95% confidence intervals are shown). The BLEU score was calculated from the first translation hypothesis produced by the IMT system for each source sentence. All the obtained improvements with the online IMT system were statistically significant. The average online training time for each new sample presented to the system, and the average response time for each user interaction

(that is, time that the system uses to propose new extensions for corrected prefixes) are also shown in Table 1, which are less than a tenth of a second and around two tenths of a second respectively[1]. According to the reported response and online training times, we can argue that the system proposed here is able to be used on real time scenarios.

	System	BLEU	KSMR	LT/RT (s)
En-Sp	batch	55.1± 2.3	18.2± 1.1	– /0.09
	online	60.6± 2.3	15.8± 1.0	0.04 /0.09
En-Fr	batch	33.7± 2.0	33.9± 1.3	– /0.14
	online	42.2± 2.2	27.9± 1.3	0.09 /0.14
En-Ge	batch	20.4± 1.8	40.3± 1.2	– /0.15
	online	28.0± 2.0	35.0± 1.3	0.07 /0.15

Table 1: BLEU and KSMR results for the XEROX test corpora using the batch and the online IMT systems, reporting the average online learning (LT) and the interaction response times (RP) in seconds.

It is worth mentioning that the results presented here significantly improve those presented in (Barrachina et al., 2009) for other state-of-the-art IMT systems using the same corpora.

7 Conclusions

We have described an IMT system with online learning which is able to learn from user feedback in real time. As far as we know, to our knowledge, this feature have never been provided by previously presented IMT prototypes.

The proposed IMT tool is publicly available through the Web (`http://cat.iti.upv.es/imt/`). Currently, the system can be used to interactively translate the well-known Europarl corpus. We have also carried out experiments with simulated users. According to such experiments, our IMT system is able to outperform the results obtained by conventional IMT systems implementing batch learning. Future work includes researching further on the benefits provided by our online learning techniques with experiments involving real users.

Acknowledgments

Work supported by the EC (FEDER/FSE), the Spanish Government (MEC, MICINN, MITyC, MAEC, "Plan E", under grants MIPRCV "Consolider Ingenio 2010" CSD2007-00018, iTrans2 TIN2009-14511, erudito.com TSI-020110-2009-439), the Generalitat Valenciana (grant Prometeo/2009/014, grant GV/2010/067), the Universitat Politècnica de València (grant 20091027), and the Spanish JCCM (grant PBI08-0210-7127).

[1] All the experiments were executed in a PC with 2.40 GHz Intel Xeon processor and 1GB of memory.

References

S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. Lagarda, H. Ney, J. Tomás, and E. Vidal. 2009. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

N. Cesa-Bianchi, G. Reverberi, and S. Szedmak. 2008. Online learning algorithms for computer-assisted translation. Deliverable D4.2, SMART: Stat. Multilingual Analysis for Retrieval and Translation.

G. Foster, P. Isabelle, and P. Plamondon. 1997. Target-text mediated interactive machine translation. *Machine Translation*, 12(1):175–194.

G. Foster, P. Langlais, and G. Lapalme. 2002. Transtype: text prediction for translators. In *Proc. HLT*, pages 372–374.

P. Isabelle and K. Church. 1997. Special issue on new tools for human translators. *Machine Translation*, 12(1–2).

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of the MT Summit X*, pages 79–86, September.

P. Koehn. 2009. A web-based interactive computer aided translation tool. In *Proc. ACL-IJCNLP*, ACLDemos, pages 17–20.

P. Langlais, G. Lapalme, and M. Loranger. 2002. Transtype: Development-evaluation cycles to boost translator's productivity. *Machine Translation*, 15(4):77–98.

R.M. Neal and G.E. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Proc. of the NATO-ASI on Learning in graphical models*, pages 355–368, Norwell, MA, USA.

L. Nepveu, G. Lapalme, P. Langlais, and G. Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Proc. EMNLP*, pages 190–197.

F. J. Och and H. Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. ACL*, pages 295–302.

D. Ortiz-Martínez, I. García-Varea, and F. Casacuberta. 2010. Online learning for interactive statistical machine translation. In *Proc. NAACL/HLT*, pages 546–554.

Wikulu: An Extensible Architecture for Integrating Natural Language Processing Techniques with Wikis

Daniel Bär, Nicolai Erbs, Torsten Zesch, and Iryna Gurevych
Ubiquitous Knowledge Processing Lab
Computer Science Department, Technische Universität Darmstadt
Hochschulstrasse 10, D-64289 Darmstadt, Germany
www.ukp.tu-darmstadt.de

Abstract

We present *Wikulu*[1], a system focusing on supporting wiki users with their everyday tasks by means of an intelligent interface. Wikulu is implemented as an extensible architecture which transparently integrates natural language processing (NLP) techniques with wikis. It is designed to be deployed with any wiki platform, and the current prototype integrates a wide range of NLP algorithms such as keyphrase extraction, link discovery, text segmentation, summarization, or text similarity. Additionally, we show how Wikulu can be applied for visually analyzing the results of NLP algorithms, educational purposes, and enabling semantic wikis.

1 Introduction

Wikis are web-based, collaborative content authoring systems (Leuf and Cunningham, 2001). As they offer fast and simple means for adding and editing content, they are used for various purposes such as creating encyclopedias (e.g. *Wikipedia*[2]), constructing dictionaries (e.g. *Wiktionary*[3]), or hosting online communities (e.g. *ACLWiki*[4]). However, as wikis do not enforce their users to structure pages or add complementary metadata, wikis often end up as a mass of unmanageable pages with meaningless page titles and no usable link structure (Buffa, 2006).

To solve this issue, we present the *Wikulu* system which uses natural language processing to support wiki users with their typical tasks of adding, organizing, and finding content. For example, Wikulu supports users with reading longer texts by *highlighting keyphrases* using keyphrase extraction methods such as *TextRank* (Mihalcea and Tarau, 2004). Support integrated in Wikulu also includes *text segmentation* to segment long pages, *text similarity* for detecting potential duplicates, or *text summarization* to facilitate reading of lengthy pages. Generally, Wikulu allows to integrate any NLP component which conforms to the standards of *Apache UIMA* (Ferrucci and Lally, 2004).

Wikulu is designed to integrate seamlessly with any wiki. Our system is implemented as an HTTP proxy server which intercepts the communication between the web browser and the underlying wiki engine. No further modifications to the original wiki installation are necessary. Currently, our system prototype contains adaptors for two widely used wiki engines: *MediaWiki*[5] and *TWiki*[6]. Adaptors for other wiki engines can be added with minimal effort. Generally, Wikulu could also be applied to any web-based system other than wikis with only slight modifications to its architecture.

In Figure 1, we show the integration of Wikulu with Wikipedia.[7] The additional user interface components are integrated into the default toolbar (highlighted by a red box in the screenshot). In this example, the user has requested keyphrase highlighting in order to quickly get an idea about the main content of the wiki article. Wikulu then invokes the

[1] Portmanteau of the Hawaiian terms *wiki* ("fast") and *kukulu* ("to organize")

[2] http://www.wikipedia.org

[3] http://www.wiktionary.org

[4] http://aclweb.org/aclwiki

[5] http://mediawiki.org (e.g. used by Wikipedia)

[6] http://twiki.org (often used for corporate wikis)

[7] As screenshots only provide a limited overview of Wikulu's capabilities, we refer the reader to a screencast: http://www.ukp.tu-darmstadt.de/research/projects/wikulu

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 74–79,
Portland, Oregon, USA, 21 June 2011.

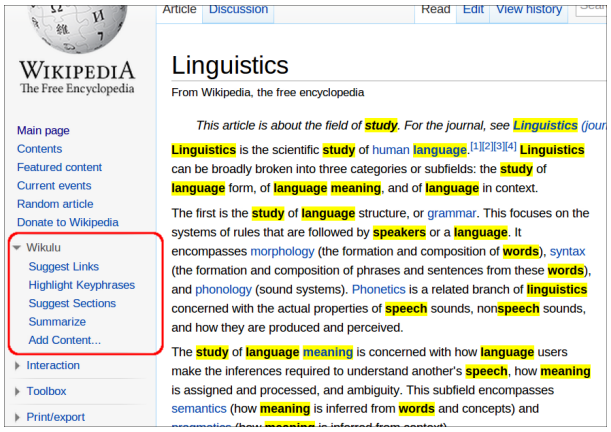


Figure 1: Integration of Wikulu with Wikipedia. The augmented toolbar (red box) and the results of a keyphrase extraction algorithm (yellow text spans) are highlighted.

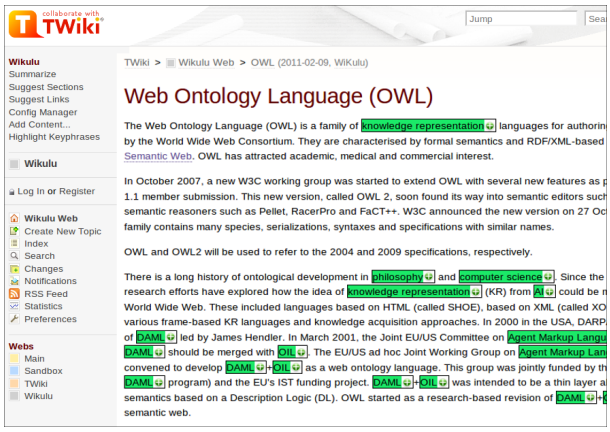


Figure 2: Automatic discovery of links to other wiki articles. Suitable text phrases to place a link on are highlighted in green.

corresponding NLP component, and highlights the returned keyphrases in the article. In the next section, we give a more detailed overview of the different types of support provided by Wikulu.

2 Supporting Wiki Users by Means of NLP

In this section, we present the different types of NLP-enabled support provided by Wikulu.

Detecting Duplicates Whenever users add new content to a wiki there is the danger of duplicating already contained information. In order to avoid duplication, users would need comprehensive knowledge of what content is already present in the wiki, which is almost impossible for large wikis like Wikipedia. Wikulu helps to detect potential duplicates by computing the text similarity between newly added content and each existing wiki page. If a potential duplicate is detected, the user is notified and may decide to augment the duplicate page instead of adding a new one. Wikulu integrates text similarity measures such as *Explicit Semantic Analysis* (Gabrilovich and Markovitch, 2007) and *Latent Semantic Analysis* (Landauer et al., 1998).

Suggesting Links While many wiki users readily add textual contents to wikis, they often restrain from also adding links to related pages. However, links in wikis are crucial as they allow users to quickly navigate from one page to another, or browse through the wiki. Therefore, it may be reasonable to augment a page about the topic *sentiment analysis* by a link to a page providing related information such as evaluation datasets. Wikulu supports users in this tedious task by automatically suggesting links. Link suggestion thereby is a two-step process: (a) first, suitable text phrases are extracted which might be worth to place a link on (see Figure 2), and (b) for each phrase, related pages are ranked by comparing their relevance to the current page, and then presented to the user. The user may thus decide whether she wants to use a detected phrase as a link or not, and if so, which other wiki page to link this phrase to. Wikulu currently integrates link suggestion algorithms by Geva (2007) and Itakura and Clarke (2007).

Semantic Searching The capabilities of a wiki's built-in search engine are typically rather limited as it traditionally performs e.g. keyword-based retrieval. If that keyword is not found in the wiki, the query returns an empty result set. However, a page might exist which is semantically related to the keyword, and should thus yield a match.

As the search engine is typically a core part of the wiki system, it is rather difficult to modify its behavior. However, by leveraging Wikulu's architecture, we can replace the default search mechanisms by algorithms which allow for *semantic search* to alleviate the vocabulary mismatch problem (Gurevych et al., 2007).

Segmenting Long Pages Due to the open editing policy of wikis, pages tend to grow rather fast.

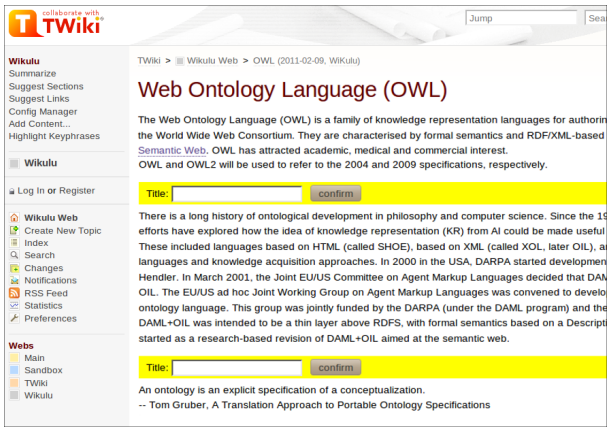


Figure 3: Analysis of a wiki article with respect to topical coherence. Suggested segment breaks are highlighted by yellow bars.

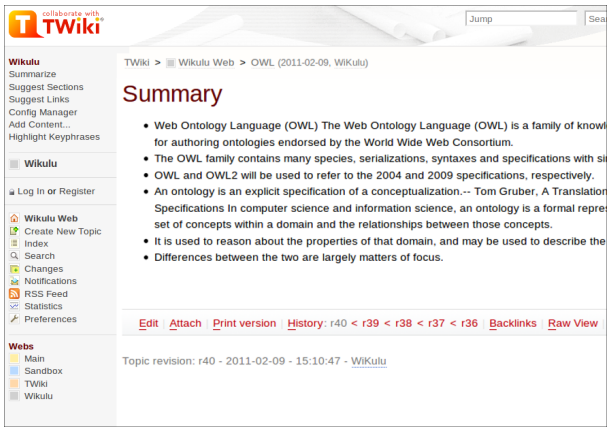


Figure 4: Extractive summary of the original wiki page shown in Figure 3

For users, it is thus a major challenge to keep an overview of what content is present on a certain page. Wikulu therefore supports users by analyzing long pages through employing text segmentation algorithms which detect topically coherent segments of text. It then suggests segment boundaries which the user may or may not accept for inserting a subheading which makes pages easier to read and better to navigate. As shown in Figure 3, users are also encouraged to set a title for each segment.[8] When accepting one or more of these suggested boundaries, Wikulu stores them persistently in the wiki. Wikulu currently integrates text segmentation methods such as *TextTiling* (Hearst, 1997) or *C99* (Choi, 2000).

Summarizing Pages Similarly to segmenting pages, Wikulu makes long wiki pages more accessible by generating an extractive summary. While generative summaries generate a summary in own words, extractive summaries analyze the original wiki text sentence-by-sentence, rank each sentence, and return a list of the most important ones (see Figure 4). Wikulu integrates extractive text summarization methods such as *LexRank* (Erkan and Radev, 2004).

Highlighting Keyphrases Another approach to assist users in better grasping the idea of a wiki page at a glance is to highlight important keyphrases (see Figure 1). As Tucker and Whittaker (2009) have shown, highlighting important phrases assists users with reading longer texts and yields faster understanding. Wikulu thus improves readability by employing automatic keyphrase extraction algorithms. Additionally, Wikulu allows to dynamically adjust the number of keyphrases shown by presenting a slider to the user. We integrated keyphrase extraction methods such as *TextRank* (Mihalcea and Tarau, 2004) and *KEA* (Witten et al., 1999).

[8] In future work, we plan to suggest suitable titles for each segment automatically.

3 Further Use Cases

Further use cases for supporting wiki users include (i) visually analyzing the results of NLP algorithms, (ii) educational purposes, and (iii) enabling semantic wikis.

Visually Analyzing the Results of NLP Algorithms Wikulu facilitates analyzing the results of NLP algorithms by using wiki pages as input documents and visualizing the results directly on that page. Consider an NLP algorithm which performs sentiment analysis. Typically, we were to put our analysis sentences in a text file, launch the NLP application, process the file, and would read the output from either a built-in console or a separate output file. This procedure suffers from two major drawbacks: (a) it is inconvenient to copy existing data into a custom input format which can be fed into the NLP system, and (b) the textual output does not allow presenting the results in a visually rich manner.

Wikulu tackles both challenges by using wiki pages as input/output documents. For instance,

by running the sentiment analysis component right from within the wiki, its output can be written back to the originating wiki page, resulting in visually rich, possibly interactive presentations.

Educational Purposes Wikulu is a handy tool for educational purposes as it allows to (a) rapidly create test data in a collaborative manner (see Section 2), and (b) visualize the results of NLP algorithms, as described above. Students can gather hands-on experience by experimenting with NLP components in an easy-to-use wiki system. They can both collaboratively edit input documents, and explore possible results of e.g. different configurations of NLP components. In our system prototype, we integrated highlighting parts-of-speech which have been determined by a POS tagger.

Enabling Semantic Wikis Semantic wikis such as the *Semantic MediaWiki* (Krötzsch et al., 2006) augment standard wikis with machine-readable semantic annotations of pages and links. As those annotations have to be entered manually, this step is often skipped by users which severely limits the usefulness of semantic wikis. Wikulu could support users e.g. by automatically suggesting the type of a link by means of relation detection or the type of a page by means of text categorization. Thus, Wikulu could constitute an important step towards the *semantification* of the content contained in wikis.

4 System Architecture

In this section, we detail our system architecture and describe what is necessary to make NLP algorithms available through our system. We also give a walkthrough of Wikulu's information flow.

4.1 Core Components

Wikulu builds upon a modular architecture, as depicted in Figure 5. It acts as an HTTP proxy server which intercepts the communication between the web browser and the target wiki engine, while it allows to run any *Apache UIMA*-compliant NLP component using an extensible plugin mechanism.

In the remainder of this section, we introduce each module: (a) the proxy server which allows to add Wikulu to any target wiki engine, (b) the JavaScript injection that bridges the gap between the client- and server-side code, (c) the plugin manager which gives access to any *Apache UIMA*-based NLP component, and (d) the wiki abstraction layer which offers a high-level interface to typical wiki operations such as reading and writing the wiki content.

Proxy Server Wikulu is designed to work with any underlying wiki engine such as *MediaWiki* or *TWiki*. Consequently, we implemented it as an HTTP proxy server which allows it to be enabled at any time by changing the proxy settings of a user's web browser.[9] The proxy server intercepts all requests between the user who interacts with her web browser, and the underlying wiki engine. For example, Wikulu passes certain requests to its language processing components, or augments the default wiki toolbar by additional commands. We elaborate on the latter in the following paragraph.

JavaScript Injection Wikulu modifies the requests between web browser and target wiki by injecting custom client-side JavaScript code. Wikulu is thus capable of altering the default behavior of the wiki engine, e.g. replacing a keyword-based retrieval by enhanced search methods (cf. Section 2), adding novel behavior such as additional toolbar buttons or advanced input fields, or augmenting the originating web page after a certain request has been processed, e.g. an NLP algorithm has been run.

Plugin Manager Wikulu does not perform language processing itself. It relies on *Apache UIMA*-compliant NLP components which use wiki pages (or parts thereof) as input texts. Wikulu offers a sophisticated plugin manager which takes care of dynamically loading those NLP components. The plugin loader is designed to run plugins either every time a wiki page loads, or manually by picking them from the augmented wiki toolbar.

The NLP components are available as server-side Java classes. Via direct web remoting[10], those components are made accessible through a JavaScript proxy object. Wikulu offers a generic language processing plugin which takes the current page contents

[9] The process of enabling a custom proxy server can be simplified by using web browser extensions such as *Multiproxy Switch* (`https://addons.mozilla.org/de/firefox/addon/multiproxy-switch`).

[10] `http://directwebremoting.org`

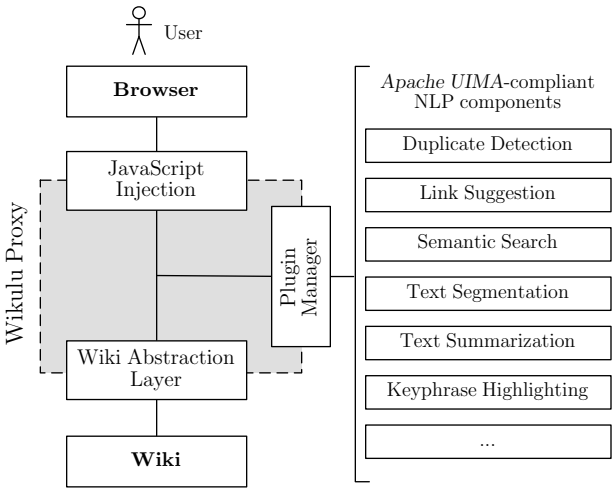


Figure 5: Wikulu acts as a proxy server which intercepts the communication between the web browser and the underlying wiki engine. Its plugin manager allows to integrate any *Apache UIMA*-compliant NLP component.

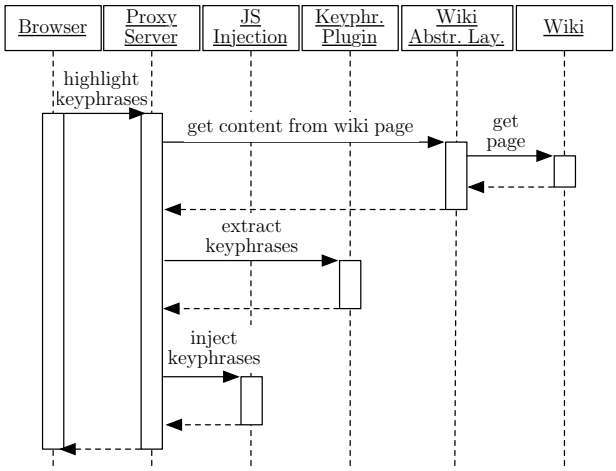


Figure 6: Illustration of Wikulu's information flow when a user has requested to *highlight keyphrases on the current page* as described in Section 4.2

as input text, runs an NLP component, and writes its output back to the wiki. To run a custom *Apache UIMA*-compliant NLP component with Wikulu, one just needs to plug that particular NLP component into the generic plugin. No further adaptations to the generic plugin are necessary. However, more advanced users may create fully customized plugins.

Wiki Abstraction Layer Wikulu communicates with the underlying wiki engine via an abstraction layer. That layer provides a generic interface for accessing and manipulating the underlying wiki engine. Thereby, Wikulu can both be tightly coupled to a certain wiki instance such as *MediaWiki* or *TWiki*, while being flexible at the same time to adapt to a changing environment. New adaptors for other target wiki engines such as *Confluence*[11] can be added with minimal effort.

4.2 Walk-Through Example

Let's assume that a user encounters a wiki page which is rather lengthy. She realizes that Wikulu's keyphrase extraction component might help her to better grasp the idea of this page at a glance, so she activates Wikulu by setting her web browser to pass all requests through the proxy server. After applying the settings, the JavaScript injection module adds additional links to the wiki's toolbar on the originating wiki page. Having decided to apply keyphrase extraction, she then invokes that NLP component by clicking the corresponding link (see Figure 6). Before the request is passed to that component, Wikulu extracts the wiki page contents using the high-level wiki abstraction layer. Thereafter, the request is passed via direct web remoting to the NLP component which has been loaded by Wikulu's plugin mechanism. After processing the request, the extracted keyphrases are returned to Wikulu's custom JavaScript handlers and finally highlighted in the originating wiki page.

5 Related Work

Supporting wiki users with NLP techniques has not attracted a lot of research attention yet. A notable exception is the work by Witte and Gitzinger (2007). They propose an architecture to connect wikis to services providing NLP functionality which are based on the *General Architecture for Text Engineering* (Cunningham et al., 2002). Contrary to Wikulu, though, their system does not integrate transparently with an underlying wiki engine, but rather uses a separate application to apply NLP techniques. Thereby, wiki users can leverage the power of NLP algorithms, but need to interrupt their current workflow to switch to a different application.

[11] `http://www.atlassian.com/software/confluence`

Moreover, their system is only loosely coupled with the underlying wiki engine. While it allows to read and write existing pages, it does not allow further modifications such as adding user interface controls.

A lot of work in the wiki community is done in the context of Wikipedia. For example, the *FastestFox*[12] plug-in for Wikipedia is able to suggest links to related articles. However, unlike Wikulu, FastestFox is tailored towards Wikipedia and cannot be used with any other wiki platform.

6 Summary

We presented Wikulu, an extensible system which integrates natural language processing techniques with wikis. Wikulu addresses the major challenge of supporting wiki users with their everyday tasks. Besides that, we demonstrated how Wikulu serves as a flexible environment for (a) visually analyzing the results of NLP algorithms, (b) educational purposes, and (c) enabling semantic wikis. By its modular and flexible architecture, we envision that Wikulu can support wiki users both in small focused environments as well as in large-scale communities such as Wikipedia.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Klaus Tschira Foundation under project No. 00.133.2008. We would like to thank Johannes Hoffart for designing and implementing the foundations of this work, as well as Artem Vovk and Carolin Deeg for their contributions.

References

Michel Buffa. 2006. Intranet Wikis. In *Proceedings of the IntraWebs Workshop at the 15th International Conference on World Wide Web*.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 26–33.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175.

Güneş Erkan and Dragomir Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

David Ferrucci and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, pages 1–26.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611.

Shlomo Geva. 2007. GPX: Ad-Hoc Queries and Automated Link Discovery in the Wikipedia. In *Preproceedings of the INEX Workshop*, pages 404–416.

Iryna Gurevych, Christof Müller, and Torsten Zesch. 2007. What to be?–Electronic Career Guidance Based on Semantic Relatedness. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 1032–1039.

Marti A. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Kelly Y. Itakura and Charles L. A. Clarke. 2007. University of Waterloo at INEX2007: Adhoc and Link-the-Wiki Tracks. In *INEX 2007 Workshop Preproceedings*, pages 417–425.

Markus Krötzsch, Denny Vrandečić, and Max Völkel. 2006. Semantic MediaWiki. In *Proc. of the 5th International Semantic Web Conference*, pages 935–942.

Thomas K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. An introduction to Latent Semantic Analysis. *Discourse Processes*, 25(2):259–284.

Bo Leuf and Ward Cunningham. 2001. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Simon Tucker and Steve Whittaker. 2009. Have A Say Over What You See: Evaluating Interactive Compression Techniques. In *Proceedings of the Intl. Conference on Intelligent User Interfaces*, pages 37–46.

René Witte and Thomas Gitzinger. 2007. Connecting wikis and natural language processing systems. In *Proc. of the Intl. Symposium on Wikis*, pages 165–176.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255.

[12] `http://smarterfox.com`

A Speech-based Just-in-Time Retrieval System using Semantic Search

Andrei Popescu-Belis, Majid Yazdani, Alexandre Nanchen, and Philip N. Garner
Idiap Research Institute
Rue Marconi 19, CP 592
1920 Martigny, Switzerland
{apbelis,myazdani,ananchen,pgarner}@idiap.ch

Abstract

The Automatic Content Linking Device is a just-in-time document retrieval system which monitors an ongoing conversation or a monologue and enriches it with potentially related documents, including multimedia ones, from local repositories or from the Internet. The documents are found using keyword-based search or using a semantic similarity measure between documents and the words obtained from automatic speech recognition. Results are displayed in real time to meeting participants, or to users watching a recorded lecture or conversation.

1 Introduction

Enriching a monologue or a conversation with related content, such as textual or audio-visual documents on the same topic, is a task with multiple applications in the field of computer-mediated human-human communication. In this paper, we describe the Automatic Content Linking Device (ACLD), a system that analyzes spoken input from one or more speakers using automatic speech recognition (ASR), in order to retrieve related content, in real-time, from a variety of repositories. These include local document databases or archives of multimedia recordings, as well as websites. Local repositories are queried using a keyword-based search engine, or using a semantic similarity measure, while websites are queried using commercial search engines.

We will first describe the scenarios of use of the ACLD in Section 2, and review previous systems for just-in-time retrieval in Section 3. The ACLD components will be outlined in Sections 4.1 to 4.5. Four types of evaluation results obtained with our system will finally be summarized in Sections 5.1 to 5.4.

2 Content Linking: Scenarios of Use

Just-in-time information retrieval, i.e. finding useful documents without the need for a user to initiate a direct search for them, is one of the ways in which the large quantity of knowledge that is available in networked environments can be efficiently put to use. To perform this task, a system must consider explicit and implicit input from users, mainly speech or typed input, and attempt to model their context, in order to provide recommendations, which users are free to consult if they feel the need for additional information.

One of the main scenarios of use for the ACLD involves people taking part in meetings, who often mention documents containing facts under discussion, but do not have the time to search for them without interrupting the discussion flow. The ACLD performs this search for them. Moreover, as the ACLD was developed on meetings from the AMI Corpus, it can also perform the same operations on a replayed meeting, as a complement to a meeting browser, for development or demonstration purposes.

In a second scenario, content linking is performed over live or recorded lectures, for instance in a computer-assisted learning environment for individual students. The ACLD enriches the lectures with related material drawn from various repositories, through a search process that can be guided in real

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 80–85,
Portland, Oregon, USA, 21 June 2011.

time by its user. The advantage of real-time content linking over a more static enrichment, such as the Feynman lectures at Microsoft Research,[1] is that users can tune search parameters at will while viewing the lecture.

3 Just-in-Time Retrieval Systems

The first precursors to the ACLD were the Fixit query-free search system (Hart and Graham, 1997), the Remembrance Agent for just-in-time retrieval (Rhodes and Maes, 2000), and the Implicit Queries (IQ) system (Dumais et al., 2004). Fixit monitored the state of a user's interaction with a diagnostic system, and excerpts from maintenance manuals depending on the interaction state. The Remembrance Agent was integrated to the Emacs text editor, and ran searches over emails or notes at regular time intervals (every few seconds) using the latest 20–500 words typed by the user. The IQ system generated context-sensitive searches based on a user's ongoing activities on their computer, such as writing email. A version of the Remembrance Agent called Jimminy was conceived as a wearable assistant for taking notes, but ASR was only simulated for evaluation (Rhodes, 1997).

The Watson system (Budzik and Hammond, 2000) monitored the user's operations in a text editor, but proposed a more complex mechanism than the Remembrance Agent for selecting terms for queries, which were directed to a web search engine. Another assistant for an authoring environment was developed in the A-Propos project (Puerta Melguizo et al., 2008). A query-free system was designed for enriching television news with articles from the Web (Henziker et al., 2005).

The FAME interactive space (Metze and al., 2006), which provides multi-modal access to recordings of lectures via a table top interface, bears many similarities to the ACLD. However, it requires the use of specific voice commands by one user only, and does not spontaneously follow a conversation.

More recently, several speech-based search engines have become available, including as smart phone applications. Conversely, many systems allow searching of spoken document archives.[2] Inspiration from these approaches, which are not query-free, can nevertheless be useful to just-in-time retrieval. Other related systems are the Speech Spotter (Goto et al., 2004) and a personal assistant using dual-purpose speech (Lyons et al., 2004), which enable users to search for information using commands that are identified in the speech flow.

The ACLD improves over numerous past ones by giving access to indexed multimedia recordings as well as websites, with fully operational ASR and semantic search, as we now explain.

4 Description of the ACLD

The architecture of the ACLD comprises the following functions: document preparation, text extraction and indexing; input sensing and query preparation; search and integration of results; user interface to display the results.

4.1 Document Preparation and Indexing

The preparation of the local database of documents for content linking involves mainly the extraction of text, and then the indexing of the documents, which is done using Apache Lucene software. Text can be extracted from a large variety of formats (including MS Office, PDF, and HTML) and hierarchies of directories are recursively scanned. The document repository is generally prepared before using the ACLD, but users can also add files at will. Because past discussions are relevant to subsequent ones, they are passed through offline ASR and then chunked into smaller units (e.g. of fixed length, or based on a homogeneous topic). The resulting texts are indexed along with the other documents.

The ACLD uses external search engines to search in external repositories, for instance the Google Web search API or the Google Desktop application to search the user's local drives.

4.2 Sensing the User's Information Needs

We believe that the most useful cues about the information needs of participants in a conversation, or of people viewing a lecture, are the words that are spoken during the conversation or the lecture. For the ACLD, we use the AMI real-time ASR system (Garner et al., 2009). One of its main features is the use of a pre-compiled grammar, which allows it to retain accuracy even when running in real-

[1] See http://research.microsoft.com/apps/tools/tuva/.

[2] See workshops at http://www.searchingspeech.org.

time on a low resource machine. Of course, when content linking is done over past meetings, or for text extraction from past recordings, the ASR system runs slower than real-time to maximize accuracy of recognition. However, the accuracy of real-time ASR is only about 1% lower than the unconstrained mode which takes several times real-time.

For the RT07 meeting data, when using signals from individual headset microphones, the AMI ASR system reaches about 38% word error rate. With a microphone array, this increases to about 41%. These values indicate that enough correct words are sensed by the real-time ASR to make it applicable to the ACLD, and that a robust search mechanism could help avoiding retrieval errors due to spurious words.

The words obtained from the ASR are filtered for stopwords, so that only content words are used for search; our list has about 80 words. Furthermore, we believe that existing knowledge about the important terminology of a domain or project can be used to increase the impact of specific words on search. A list of pre-specified keywords can be defined based on such knowledge and can be modified while running the ACLD. For instance, for remote control design as in the AMI Corpus scenario, this list includes about 30 words such as 'chip', 'button', or 'material'. If any of them is detected in the ASR output, then their importance is increased for searching, but otherwise all the other words from the ASR (minus the stopwords) are used for constructing the query.

4.3 Querying the Document Database

The Query Aggregator (QA) uses the ASR words to retrieve the most relevant documents from one or more databases. The current version of the ACLD makes use of semantic search (see next subsection), while previous versions used word-based search from Apache Lucene for local documents, or from the Google Web or Google Desktop APIs. ASR words from the latest time frame are put together (minus the stopwords) to form queries, and recognized keywords are boosted in the Lucene query. Queries are formulated at regular time intervals, typically every 15-30 seconds, or on demand. This duration is a compromise between the need to gather enough words for search, and the need to refresh the search results reasonably often.

The results are integrated with those from the previous time frame, using a persistence model to smooth variations over time. The model keeps track of the salience of each result, initialized from their ranking among the search results, then decreasing in time unless the document is again retrieved. The rate of decrease (or its inverse, persistence) can be tuned by the user, but in any case, all past results are saved by the user interface and can be consulted at any time.

4.4 Semantic Search over Wikipedia

The goal of our method for semantic search is to improve the relevance of the retrieved documents, and to make the mechanism more robust to noise from the ASR. We have applied to document retrieval the graph-based model of semantic relatedness that we recently developed (Yazdani and Popescu-Belis, 2010), which is also related to other proposals (Strube and Ponzetto, 2006; Gabrilovich and Markovitch, 2007; Yeh et al., 2009).

The model is grounded in a measure of semantic relatedness between text fragments, which is computed using random walk over the network of Wikipedia articles – about 1.2 million articles from the WEX data set (Metaweb Technologies, 2010). The articles are linked through hyperlinks, and also through lexical similarity links that are constructed upon initialization. The random walk model allows the computation of a *visiting probability (VP)* from one article to another, and then a VP between sets of articles, which has been shown to function as a measure of semantic relatedness, and has been applied to various NLP problems. To compute relatedness between two text fragments, these are first projected represented into the network by the ten closest articles in terms of lexical similarity.

For the ACLD, the use of semantic relatedness for document retrieval amounts to searching, in a very large collection, the documents that are the most closely related to the words from the ASR in a given timeframe. Here, the document collection is (again) the set of Wikipedia articles from WEX, and the goal is to return the eight most related articles. Such a search is hard to perform in real time; hence, the solution that was found makes use of several approximations to compute average VP between the ASR fragment and all articles in the Wikipedia network.

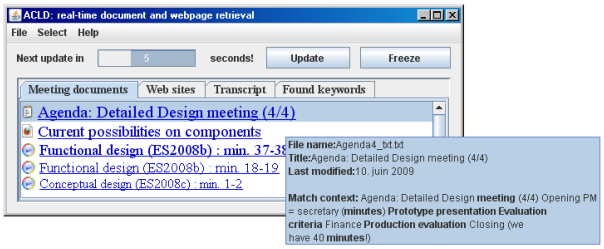


Figure 1: Unobtrusive UI displaying document results. Hovering the mouse over a result (here, the most relevant one) displays a pop-up window with more information about it.

4.5 The User Interface (UI)

The main goal of the UI is to make available all information produced by the system, in a configurable way, allowing users to see a larger or smaller amount of information according to their needs. A modular architecture with a flexible layout has been implemented, maximizing the accessibility but also the understandability of the results, and displaying also intermediary data such as ASR words and found keywords. The UI displays up to five widgets, which can be arranged at will:

1. ASR results with highlighted keywords.
2. Tag-cloud of keywords, coding for recency and frequency of keywords.
3. Names of documents and past meeting snippets found by the QA.
4. Names of web pages found via the Google API.
5. Names of local files found via the Google Desktop API.

Two main arrangements are intended, though many others are possible: an informative full-screen UI, shown in Figure 2 with widgets 1–4; and an unobtrusive widget UI, with superposed tabs, shown in Figure 1 with widget 3.

The document names displayed in widgets 3–5 function as hyperlinks to the documents, launching appropriate external viewers when the user clicks on them. Moreover, when hovering over a document name, a pop-up window displays metadata and document excerpts that match words from the query, as an explanation of why the document was retrieved.

5 Evaluation Experiments

Four types of evidence for the relevance and utility of the ACLD are summarized in this section.

5.1 Feedback from Potential Users

The ACLD was demonstrated to about 50 potential users (industrial partners, focus groups, etc.) in a series of sessions of about 30 minutes, starting with a presentation of the ACLD and continuing with a discussion and elicitation of feedback. The overall concept was generally found useful, with positive verbal evaluations. Feedback for smaller and larger improvements was collected: e.g. the importance of matching context, linking on demand, and the UI unobtrusive mode.

5.2 Pilot Task-based Experiments

A pilot experiment was conducted by a team at the University of Edinburgh with an earlier version of the unobtrusive UI. Four subjects had to complete a task that was started in previous meetings (ES2008a-b-c from the AMI Corpus). The goal was to compare two conditions, *with* vs. *without* the ACLD, in terms of satisfied constraints, overall efficiency, and satisfaction. Two pilot runs have shown that the ACLD was being consulted about five times per meeting. Therefore, many more runs are required to reach statistical significance of observations, and remain to be executed depending on future resources.

5.3 Usability Evaluation of the UI

The UI was submitted to a usability evaluation experiment with nine non-technical subjects. The subjects used the ACLD over a replayed meeting recording, and were asked to perform several tasks with it, such as adding a keyword to monitor, searching for a word, or changing the layout. The subjects then rated usability-related statements, leading to an assessment on the System Usability Scale (Brooke, 1996).

The overall usability score was 68% (SD: 10), which is considered as 'acceptable usability' for the SUS. The average task-completion time was 45–75 seconds. In free-form feedback, subjects found the system helpful to review meetings but also lectures, appreciated the availability of documents, but also noted that search results (with keyword-based

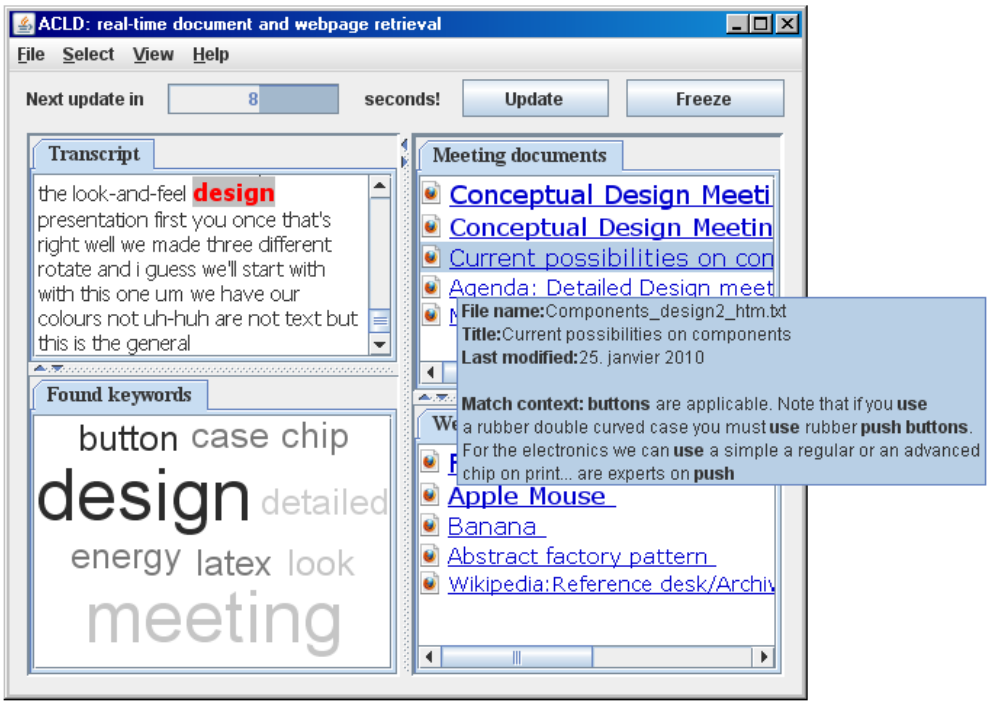


Figure 2: Full screen UI with four widgets: ASR, keywords, document and website results.

search) were often irrelevant. They also suggested simplifying the UI (menus, layout) and embedding a media player for use in the meeting or lecture replay scenario.

5.4 Comparing the Relevance of Keyword-based vs. Semantic Search

We compared the output of semantic search with that of keyword-based search. The ASR transcript of one AMI meeting (ES2008d) was passed to both search methods, and 'evaluation snippets' containing the manual transcript for one-minute excerpts, accompanied by the 8-best Wikipedia articles found by each method were produced. Overall, 36 snippets were generated. The manual transcript shown to subjects was enriched with punctuation and speakers' names, and the names of the Wikipedia pages were placed on each side of the transcript frame.

Subjects were then asked to read each snippet, and decide which of the two document sets was the most relevant to the discussion taking place, i.e. the most useful as a suggestion to the participants. They could also answer 'none', and could consult the result if necessary.

Results were obtained from 8 subjects, each seeing 9 snippets out of 36. Every snippet was thus seen by two subjects. The subjects agreed on 23 (64%) snippets and disagreed on 13 (36%). In fact, the number of true disagreements not including the answer 'none' was only 7 out of 36.

Over the 23 snippets on which subjects agreed, the result of semantic search was judged more relevant than that of keyword search for 19 snippets (53% of the total), and the reverse for 4 snippets only (11%). Alternatively, if one counts the votes cast by subjects in favor of each system, regardless of agreement, then semantic search received 72% of the votes and keyword-based only 28%. These numbers show that semantic search quite clearly improves relevance in comparison to keyword-based one, but there is still room for improvement.

6 Conclusion

The ACLD is, to the best of our knowledge, the first just-in-time retrieval system to use spontaneous speech and to support access to multimedia documents and web pages, using a robust semantic search method. Future work will aim at improving the relevance of semantic search, at modeling context to

improve timing of results, and at inferring relevance feedback from users. The ACLD should also be applied to specific use cases, and an experiment with group work in a learning environment is under way.

Acknowledgments

The authors gratefully acknowledge the support of the EU AMI and AMIDA Integrated Projects (http://www.amiproject.org) and of the Swiss IM2 NCCR on Interactive Multimodal Information Management (http://www.im2.ch).

References

John Brooke. 1996. SUS: A 'quick and dirty' usability scale. In Patrick W. Jordan, Bruce Thomas, Bernard A. Weerdmeester, and Ian L. McClelland, editors, *Usability evaluation in industry*, pages 189–194. Taylor and Francis, London, UK.

Jay Budzik and Kristian J. Hammond. 2000. User interactions with everyday applications as context for just-in-time information access. In *IUI 2000 (5th International Conference on Intelligent User Interfaces)*, New Orleans, LA.

Susan Dumais, Edward Cutrell, Raman Sarin, and Eric Horvitz. 2004. Implicit Queries (IQ) for contextualized search. In *SIGIR 2004 (27th ACM SIGIR Conference) Demonstrations*, page 534, Sheffield, UK.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI 2007 (20th International Joint Conference on Artificial Intelligence)*, pages 6–12, Hyderabad, India.

Philip N. Garner, John Dines, Thomas Hain, Asmaa El Hannani, Martin Karafiat, Danil Korchagin, Mike Lincoln, Vincent Wan, and Le Zhang. 2009. Real-time ASR from meetings. In *Interspeech 2009 (10th Annual Conference of the Intl. Speech Communication Association)*, pages 2119–2122, Brighton, UK.

Masataka Goto, Koji Kitayama, Katsunobu Itou, and Tetsunori Kobayashi. 2004. Speech Spotter: On-demand speech recognition in human-human conversation on the telephone or in face-to-face situations. In *ICSLP 2004 (8th International Conference on Spoken Language Processing)*, pages 1533–1536, Jeju Island.

Peter E. Hart and Jamey Graham. 1997. Query-free information retrieval. *IEEE Expert: Intelligent Systems and Their Applications*, 12(5):32–37.

Monika Henziker, Bay-Wei Chang, Brian Milch, and Sergey Brin. 2005. Query-free news search. *World Wide Web: Internet and Web Information Systems*, 8:101–126.

Kent Lyons, Christopher Skeels, Thad Starner, Cornelis M. Snoeck, Benjamin A. Wong, and Daniel Ashbrook. 2004. Augmenting conversations using dual-purpose speech. In *UIST 2004 (17th Annual ACM Symposium on User Interface Software and Technology)*, pages 237–246, Santa Fe, NM.

Metaweb Technologies. 2010. Freebase Wikipedia Extraction (WEX). http://download.freebase.com/wex/.

Florian Metze and al. 2006. The 'Fame' interactive space. In *Machine Learning for Multimodal Interaction II*, LNCS 3869, pages 126–137. Springer, Berlin.

Maria Carmen Puerta Melguizo, Olga Monoz Ramos, Lou Boves, Toine Bogers, and Antal van den Bosch. 2008. A personalized recommender system for writing in the Internet age. In *LREC 2008 Workshop on NLP Resources, Algorithms, and Tools for Authoring Aids*, pages 21–26, Marrakech, Morocco.

Bradley J. Rhodes and Pattie Maes. 2000. Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3-4):685–704.

Bradley J. Rhodes. 1997. The Wearable Remembrance Agent: A system for augmented memory. *Personal Technologies: Special Issue on Wearable Computing*, 1:218–224.

Michael Strube and Simone Paolo Ponzetto. 2006. Wikirelate! Computing semantic relatedness using Wikipedia. In *AAAI 2006 (21st National Conference on Artificial Intelligence)*, pages 1419–1424, Boston, MA.

Majid Yazdani and Andrei Popescu-Belis. 2010. A random walk framework to compute textual semantic similarity: A unified model for three benchmark tasks. In *ICSC 2010 (4th IEEE International Conference on Semantic Computing)*, pages 424–429, Pittsburgh, PA.

Eric Yeh, Daniel Ramage, Christopher D. Manning, Eneko Agirre, and Aitor Soroa. 2009. WikiWalk: random walks on Wikipedia for semantic relatedness. In *TextGraphs-4 (4th Workshop on Graph-based Methods for NLP)*, pages 41–49, Singapore.

MACAON
An NLP Tool Suite for Processing Word Lattices

Alexis Nasr Frédéric Béchet Jean-François Rey Benoît Favre Joseph Le Roux*
Laboratoire d'Informatique Fondamentale de Marseille- CNRS - UMR 6166
Université Aix-Marseille
(alexis.nasr,frederic.bechet,jean-francois.rey,benoit.favre,joseph.le.roux)
@lif.univ-mrs.fr

Abstract

MACAON is a tool suite for standard NLP tasks developed for French. MACAON has been designed to process both human-produced text and highly ambiguous word-lattices produced by NLP tools. MACAON is made of several native modules for common tasks such as a tokenization, a part-of-speech tagging or syntactic parsing, all communicating with each other through XML files . In addition, exchange protocols with external tools are easily definable. MACAON is a fast, modular and open tool, distributed under GNU Public License.

1 Introduction

The automatic processing of textual data generated by NLP software, resulting from Machine Translation, Automatic Speech Recognition or Automatic Text Summarization, raises new challenges for language processing tools. Unlike native texts (texts produced by humans), this new kind of texts is the result of imperfect processors and they are made of several hypotheses, usually weighted with confidence measures. Automatic text production systems can produce these weighted hypotheses as n-best lists, word lattices, or confusion networks. It is crucial for this space of ambiguous solutions to be kept for later processing since the ambiguities of the lower levels can sometimes be resolved during high-level processing stages. It is therefore important to be able to represent this ambiguity.

MACAON is a suite of tools developped to process ambiguous input and extend inference of input modules within a global scope. It consists in several modules that perform classical NLP tasks (tokenization, word recognition, part-of-speech tagging, lemmatization, morphological analysis, partial or full parsing) on either native text or word lattices. MACAON is distributed under GNU public licence and can be downloaded from `http://www.macaon.lif.univ-mrs.fr/`.

From a general point of view, a MACAON module can be seen as an annotation device[1] which adds a new level of annotation to its input that generally depends on annotations from preceding modules. The modules communicate through XML files that allow the representation different layers of annotation as well as ambiguities at each layer. Moreover, the initial XML structuring of the processed files (logical structuring of a document, information from the Automatic Speech Recognition module ...) remains untouched by the processing stages.

As already mentioned, one of the main characteristics of MACAON is the ability for each module to accept ambiguous inputs and produce ambiguous outputs, in such a way that ambiguities can be resolved at a later stage of processing. The compact representation of ambiguous structures is at the heart of the MACAON exchange format, described in section 2. Furthermore every module can weight the solutions it produces. such weights can be used to rank solutions or limit their number for later stages

*This work has been funded by the French Agence Nationale pour la Recherche, through the projects SEQUOIA (ANR-08-EMER-013) and DECODA (2009-CORD-005-01)

[1] Annotation must be taken here in a general sense which includes tagging, segmentation or the construction of more complex objets as syntagmatic or dependencies trees.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 86–91,
Portland, Oregon, USA, 21 June 2011.

of processing.

Several processing tools suites alread exist for French among which SXPIPE (Sagot and Boullier, 2008), OUTILEX (Blanc et al., 2006), NOOJ[2] or UNITEX[3]. A general comparison of MACAON with these tools is beyond the scope of this paper. Let us just mention that MACAON shares with most of them the use of finite state machines as core data representation. Some modules are implemented as standard operations on finite state machines.

MACAON can also be compared to the numerous development frameworks for developping processing tools, such as GATE[4], FREELING[5], ELLOGON[6] or LINGPIPE[7] that are usually limited to the processing of native texts.

The MACAON exchange format shares a certain number of features with linguistic annotation scheme standards such as the Text Encoding Initiative[8], XCES[9], or EAGLES[10]. They all aim at defining standards for various types of corpus annotations. The main difference between MACAON and these approaches is that MACAON defines an exchange format between NLP modules and not an annotation format. More precisely, this format is dedicated to the compact representation of ambiguity: some information represented in the exchange format are to be interpreted by MACAON modules and would not be part of an annotation format. Moreover, the MACAON exchange format was defined from the bottom up, originating from the authors' need to use several existing tools and adapt their input/output formats in order for them to be compatible. This is in contrast with a top down approach which is usually chosen when specifying a standard. Still, MACAON shares several characteristics with the LAF (Ide and Romary, 2004) which aims at defining high level standards for exchanging linguistic data.

[2] www.nooj4nlp.net/pages/nooj.html
[3] www-igm.univ-mlv.fr/~unitex
[4] gate.ac.uk
[5] garraf.epsevg.upc.es/freeling
[6] www.ellogon.org
[7] alias-i.com/lingpipe
[8] www.tei-c.org/P5
[9] www.xml-ces.org
[10] www.ilc.cnr.it/eagles/home.html

2 The MACAON exchange format

The MACAON exchange format is based on four concepts: *segment*, *attribute*, *annotation level* and *segmentation*.

A segment refers to a segment of the text or speech signal that is to be processed, as a sentence, a clause, a syntactic constituent, a lexical unit, a named entity ... A segment can be equipped with attributes that describe some of its aspects. A syntactic constituent, for example, will define the attribute *type* which specifies its syntactic type (Noun Phrase, Verb Phrase ...). A segment is made of one or more smaller segments.

A sequence of segments covering a whole sentence for written text, or a spoken utterance for oral data, is called a *segmentation*. Such a sequence can be weighted.

An *annotation level* groups together segments of a same type, as well as segmentations defined on these segments. Four levels are currently defined: pre-lexical, lexical, morpho-syntactic and syntactic.

Two relations are defined on segments: the *precedence* relation that organises linearly segments of a given level into segmentations and the *dominance* relation that describes how a segment is decomposed in smaller segments either of the same level or of a lower level.

We have represented in figure 2, a schematic representation of the analysis of the reconstructed output a speech recognizer would produce on the input *time flies like an arrow*[11]. Three annotation levels have been represented, lexical, morpho-syntactic and syntactic. Each level is represented by a finite-state automaton which models the precedence relation defined over the segments of this level. Segment *time*, for example, precedes segment *flies*. The segments are implicitly represented by the labels of the automaton's arcs. This label should be seen as a reference to a more complex objet, the actual segment. The dominance relations are represented with dashed lines that link segments of different levels. Segment *time*, for example, is dominated by segment *NN* of the morpho-syntactic level.

This example illustrates the different ambiguity cases and the way they are represented.

[11] For readability reasons, we have used an English example, MACAON, as mentioned above, currently exists for French.

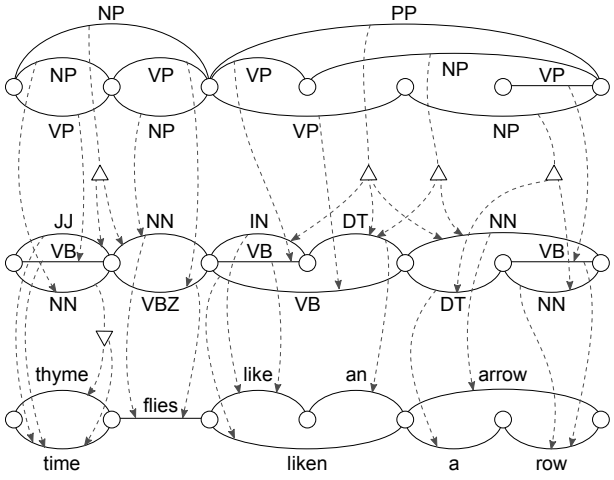


Figure 1: Three annotation levels for a sample sentence. Plain lines represent annotation hypotheses within a level while dashed lines represent links between levels. Triangles with the tip up are "and" nodes and triangles with the tip down are "or" nodes. For instance, in the part-of-speech layer, The first NN can either refer to "time" or "thyme". In the chunking layer, segments that span multiple part-of-speech tags are linked to them through "and" nodes.

The most immediate ambiguity phenomenon is the segmentation ambiguity: several segmentations are possible at every level. This ambiguity is represented in a compact way through the factoring of segments that participate in different segmentations, by way of a finite state automaton.

The second ambiguity phenomenon is the dominance ambiguity, where a segment can be decomposed in several ways into lower level segments. Such a case appears in the preceding example, where the *NN* segment appearing in one of the outgoing transition of the initial state of the morpho-syntactic level dominates both *thyme* and *time* segments of the lexical level. The triangle with the tip down is an "or" node, modeling the fact that *NN* corresponds to *time* or *thyme*.

Triangles with the tip up are "and" nodes. They model the fact that the *PP* segment of the syntactic level dominates segments *IN*, *DT* and *NN* of the morpho-syntactic level.

2.1 XML representation

The MACAON exchange format is implemented in XML. A segment is represented with the XML tag `<segment>` which has four mandatory attributes:

- `type` indicates the type of the segment, four different types are currently defined: `atome` (prelexical unit usually referred to as token in english), `ulex` (lexical unit), `cat` (part of speech) and `chunk` (a non recursive syntactic unit).

- `id` associates to a segment a unique identifier in the document, in order to be able to reference it.

- `start` and `end` define the span of the segment. These two attributes are numerical and represent either the index of the first and last character of the segment in the text string or the beginning and ending time of the segment in a speech signal.

A segment can define other attributes that can be useful for a given description level. We often find the `stype` attribute that defines subtypes of a given type.

The dominance relation is represented through the use of the `<sequence>` tag. The domination of the three segments *IN*, *DT* and *NN* by a *PP* segment, mentionned above is represented below, where `p1`, `p2` and `p3` are respectively the `id`s of segments *IN*, *DT* and *NN*.

```
<segment type="chunk" stype="PP" id="c1">
  <sequence>
    <elt segref="p1"/>
    <elt segref="p2"/>
    <elt segref="p3"/>
  </sequence>
</segment>
```

The ambiguous case, described above where segment *NN* dominates segments *time* or *thyme* is represented below as a disjunction of sequences inside a segment. The disjunction itself is not represented as an XML tag. `l1` and `l2` are respectively the `id`s of segments *time* and *thyme*.

```
<segment type="cat" stype="NN" id="c1">
  <sequence>
    <elt segref="l1" w="-3.37"/>
  </sequence>
  <sequence>
    <elt segref="l2" w="-4.53"/>
  </sequence>
</segment>
```

The dominance relation can be weighted, by way of the attribute `w`. Such a weight represents in the preceding example the conditional log-probability of a lexical unit given a part of speech, as in a hidden Markov model.

The precedence relation (i.e. the organization of segments in segmentations), is represented as a weighted finite state automaton. Automata are represented as a start state, accept states and a list of transitions between states, as in the following example that corresponds to the lexical level of our example.

```
<fsm n="9">
  <start n="0"/>
  <accept n="6"/>
  <ltrans>
    <trans o="0" d="1" i="l1" w="-7.23"/>
    <trans o="0" d="1" i="l2" w="-9.00"/>
    <trans o="1" d="2" i="l3" w="-3.78"/>
    <trans o="2" d="3" i="l4" w="-7.37"/>
    <trans o="3" d="4" i="l5" w="-3.73"/>
    <trans o="2" d="4" i="l6" w="-6.67"/>
    <trans o="4" d="5" i="l7" w="-4.56"/>
    <trans o="5" d="6" i="l8" w="-2.63"/>
    <trans o="4" d="6" i="l9" w="-7.63"/>
  </ltrans>
</fsm>
```

The `<trans/>` tag represents a transition, its `o,d,i` and `w` features are respectively the origin, and destination states, its label (the `id` of a segment) and a weight.

An annotation level is represented by the `<section>` tag which regroups two tags, the `<segments>` tag that contains the different `segment` tags defined at this annotation level and the `<fsm>` tag that represents all the segmentations of this level.

3 The MACAON architecture

Three aspects have guided the architecture of MACAON: openness, modularity, and speed. Openness has been achieved by the definition of an exchange format which has been made as general as possible, in such a way that mapping can be defined from and to third party modules as ASR, MT systems or parsers. Modularity has been achieved by the definition of independent modules that communicate with each other through XML files using standard UNIX pipes. A module can therefore be replaced easily. Speed has been obtained using efficient algorithms and a representation especially designed to load linguistic data and models in a fast way.

MACAON is composed of libraries and components. Libraries contain either linguistic data, models or API functions. Two kinds of components are presented, the MACAON core components and third party components for which mappings to and from the MACAON exchange format have been defined.

3.1 Libraries

The main MACAON library is `macaon_common`. It defines a simple interface to the MACAON exchange format and functions to load XML MACAON files into memory using efficient data structures. Other libraries `macaon_lex`, `macaon_code` and `macaon_tagger_lib` represent the lexicon, the morphological data base and the tagger models in memory.

MACAON only relies on two third-party libraries, which are `gfsm`[12], a finite state machine library and `libxml`, an XML library[13].

3.2 The MACAON core components

A brief description of several standard components developed in the MACAON framework is given below. They all comply with the exchange format described above and add a `<macaon_stamp>` to the XML file that indicates the name of the component, the date and the component version number, and recognizes a set of standard options.

maca_select is a pre-processing component: it adds a `macaon` tag under the target tags specified by the user to the input XML file. The following components will only process the document parts enclosed in `macaon` tags.

maca_segmenter segments a text into sentences by examining the context of punctuation with a regular grammar given as a finite state automaton. It is disabled for automatic speech transcriptions which do not typically include punctuation signs and come with their own segmentation.

[12] `ling.uni-potsdam.de/~moocow/projects/gfsm/`

[13] `xmlsoft.org`

maca_tokenizer tokenizes a sentence into pre-lexical units. It is also based on regular grammars that recognize simple tokens as well as a predefined set of special tokens, such as time expressions, numerical expressions, urls....

maca_lexer allows to regroup pre-lexical units into lexical units. It is based on the le*fff* French lexicon (Sagot et al., 2006) which contains around 500,000 forms. It implements a dynamic programming algorithm that builds all the possible grouping of pre-lexical units into lexical units.

maca_tagger associates to every lexical unit one or more part-of-speech labels. It is based on a trigram Hidden Markov Model trained on the French Treebank (Abeillé et al., 2003). The estimation of the HMM parameters has been realized by the SRILM toolkit (Stolcke, 2002).

maca_anamorph produces the morphological analysis of lexical units associated to a part of speech. The morphological information come from the le*fff* lexicon.

maca_chunker gathers sequences of part-of-speech tags in non recursive syntactic units. This component implements a cascade of finite state transducers, as proposed by Abney (1996). It adds some features to the initial Abney proposal, like the possibility to define the head of a chunk.

maca_conv is a set of converters from and to the MACAON exchange format. `htk2macaon` and `fsm2macaon` convert word lattices from the HTK format (Young, 1994) and ATT FSM format (Mohri et al., 2000) to the MACAON exchange format. `macaon2txt` and `txt2macaon` convert from and to plain text files. `macaon2lorg` and `lorg2macaon` convert to and from the format of the LORG parser (see section 3.3).

maca_view is a graphical interface that allows to inspect MACAON XML files and run the components.

3.3 Third party components

MACAON is an open architecture and provides a rich exchange format which makes possible the representation of many NLP tools input and output in the MACAON format. MACAON has been interfaced with the SPEERAL Automatic Speech Recognition System (Nocera et al., 2006). The word lattices produced by SPEERAL can be converted to pre-lexical MACAON automata.

MACAON does not provide any native module for parsing yet but it can be interfaced with any already existing parser. For the purpose of this demonstration we have chosen the LORG parser developed at NCLT, Dublin[14]. This parser is based on PCFGs with latent annotations (Petrov et al., 2006), a formalism that showed state-of-the-art parsing accuracy for a wide range of languages. In addition it offers a sophisticated handling of unknown words relying on automatically learned morphological clues, especially for French (Attia et al., 2010). Moreover, this parser accepts input that can be tokenized, postagged or pre-bracketed. This possibility allows for different settings when interfacing it with MACAON.

4 Applications

MACAON has been used in several projects, two of which are briefly described here, the DEFINIENS project and the LUNA project.

DEFINIENS (Barque et al., 2010) is a project that aims at structuring the definitions of a large coverage French lexicon, the *Trésor de la langue française*. The lexicographic definitions have been processed by MACAON in order to decompose the definitions into complex semantico-syntactic units. The data processed is therefore native text that possesses a rich XML structure that has to be preserved during processing.

LUNA[15] is a European project that aims at extracting information from oral data about hotel booking. The word lattices produced by an ASR system have been processed by MACAON up to a partial syntactic level from which frames are built. More details can be found in (Béchet and Nasr, 2009). The key aspect of the use of MACAON for the LUNA project is the ability to perform the linguistic analyses on the multiple hypotheses produced by the ASR system. It is therefore possible, for a given syntactic analysis, to

[14] `www.computing.dcu.ie/~lorg`. This software should be freely available for academic research by the time of the conference.

[15] `www.ist-luna.eu`

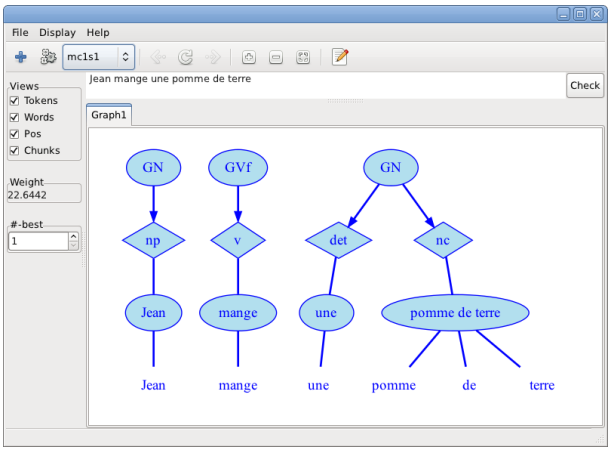


Figure 2: Screenshot of the MACAON visualization interface (for French models). It allows to input a text and see the n-best results of the annotation.

find all the word sequences that are compatible with this analysis.

Figure 2 shows the interface that can be used to see the output of the pipeline.

5 Conclusion

In this paper we have presented MACAON, an NLP tool suite which allows to process native text as well as several hypotheses automatically produced by an ASR or an MT system. Several evolutions are currently under development, such as a named entity recognizer component and an interface with a dependency parser.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Steven Abney. 1996. Partial parsing via finite-state cascades. In Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information, Prague, Czech Republic, pages 8–15.

M. Attia, J. Foster, D. Hogan, J. Le Roux, L. Tounsi, and J. van Genabith. 2010. Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of SPMRL*.

Lucie Barque, Alexis Nasr, and Alain Polguère. 2010. From the definitions of the trésor de la langue française to a semantic database of the french language. In *EURALEX 2010*, Leeuwarden, Pays Bas.

Frédéric Béchet and Alexis Nasr. 2009. Robust dependency parsing for spoken language understanding of spontaneous speech. In *Interspeech*, Brighton, United Kingdom.

Olivier Blanc, Matthieu Constant, and Eric Laporte. 2006. Outilex, plate-forme logicielle de traitement de textes écrits. In *TALN 2006*, Leuven.

Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Natural language engineering*, 10(3-4):211–225.

M. Mohri, F. Pereira, and M. Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

P. Nocera, G. Linares, D. Massonié, and L. Lefort. 2006. Phoneme lattice based A* search algorithm for speech recognition. In *Text, Speech and Dialogue*, pages 83–111. Springer.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *ACL*.

Benoît Sagot and Pierre Boullier. 2008. Sxpipe 2: architecture pour le traitement présyntaxique de corpus bruts. *Traitement Automatique des Langues*, 49(2):155–188.

Benoît Sagot, Lionel Clément, Eric Villemonte de la Clergerie, and Pierre Boullier. 2006. The le*fff* 2 Syntactic Lexicon for French: Architecture, Acquisition, Use. In *International Conference on Language Resources and Evaluation*, Genoa.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, Denver, Colorado.

S.J. Young. 1994. The HTK Hidden Markov Model Toolkit: Design and Philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44.

Multimodal Menu-based Dialogue with Speech Cursor in DICO II+

Staffan Larsson
University of Gothenburg
Sweden
sl@ling.gu.se

Alexander Berman
Talkamatic AB
Sweden
alex@talkamatic.se

Jessica Villing
University of Gothenburg
Sweden
jessica@ling.gu.se

Abstract

This paper describes Dico II+, an in-vehicle dialogue system demonstrating a novel combination of flexible multimodal menu-based dialogueand a "speech cursor" which enables menu navigation as well as browsing long list using haptic input and spoken output.

1 Introduction

Dico is a multimodal in-car dialogue system application, originally developed in the DICO (with capital letters) project by Volvo Technology AB and the University of Gothenburg. Dico is built on top of the GoDiS dialogue system platform (Larsson, 2002), which in turn is implemented using TrindiKit (Traum and Larsson, 2003).

The main goal of the original Dico application (Olsson and Villing, 2005), (Villing and Larsson, 2006) was to develop an interface that is less distracting for the driver, and thus both safer and easier to use than existing interfaces. (Larsson and Villing, 2009) described the Dico II system resulting from work in the DICO project. Since then, the Dico demonstrator has been further developed.

In this paper, we describe the Dico II+ demonstrator which introduces a novel combination of flexible Multimodal Menu-Based Dialogue and a Speech Cursor which together enable flexible multimodal interaction without the need for looking at the screen. In the following, we will first argue for the usefulness of in-vehicle dialogue systems. We will then briefly present the GoDiS platform which Dico II+ is based on, as well as some aspects of flexible dialogue enabled by the GoDiS dialogue manager.

2 In-vehicle dialogue systems

Voice interaction is a very natural means of communication for humans, and enabling spoken interaction with technologies may thus make it easier and less cognitively demanding for people to interact with machines. However, this requires that the spoken interaction is similar to ordinary spoken human-human dialogue. A problem with available in-vehicle voice control technologies is that they are not flexible enough in terms of the interaction strategies and modalities offered to the user.

3 GoDiS features in Dico

GoDiS (Larsson, 2002) is an experimental dialogue system implementing a theory of Issue-Based Dialogue Management based on Ginzburg's concept of Questions Under Discussion (QUD). GoDiS is implemented using the TrindiKit, a toolkit for implementing dialogue move engines and dialogue systems based on the Information State approach (Traum and Larsson, 2003). GoDiS has been adapted to several different dialogue types, domains, and languages, including menu-based multimodal dialogue when acting as an interface to an mp3 player (Hjelm et al., 2005).

The GoDiS dialogue manager allows the user to interact more flexibly and naturally with menu-based interfaces to devices. General dialogue management issues such as accommodation, task switching and grounding are handled by the application-independent dialogue manager. Re-using these technologies in new applications enables rapid prototyping of advanced dialogue applications.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 92–96,
Portland, Oregon, USA, 21 June 2011.

3.1 Accommodation

The applications in Dico II+ are based on existing menu interfaces, using a method for converting menus into dialogue plans (see below). While it is possible for the novice user to let the system take initiative and guide the user through the menus step-by-step, expert users might prefer to use *accommodation* strategies enabling the user to skip through the menus and getting right to the point. Two examples are shown below.

Ex. 1:

S: "What do you want to do? Go to the phonebook, manage settings, or..."

U: "Call Lisa's home number"

S: "OK. Calling Lisa's home number."

Ex. 2:

U: "Lisa's home number"

S: "OK. Do you want to change the number or make a call?"

U: "Make a call."

S: "OK. Calling Lisa's home number."

3.2 Multiple simultaneous tasks and task switching

GoDiS enables arbitrarily nested subdialogues. It is possible to start one dialogue to perform a task, and then start a subdialogue before the first task is completed. When the second task is completed the system automatically returns to the first task, and explicitly signals this. This gives the user freedom to switch task at any time:

Ex. 3:

U: "Change Lisa's home number."

S: "Okay. Let's see. What phone number do you want instead?"

U: "Check my messages."

S: "You have got this message. Hi! I have got a new home number, it is (031)234567. Best regards, Lisa."

S: "Returning to change an entry. What phone number do you want instead?"

U: "oh three one twentythree fourtyfive sixtyseven."

U: "Okay. Changing Lisa's home number to oh three one two three four five six seven."

3.3 Feedback and grounding

The GoDiS dialogue manager provides general feedback strategies to make sure that the dialogue partners have contact, that the system can can hear what the user says, understands the words that are spoken (semantic understanding), understands the meaning of the utterance (pragmatic understanding) and accepts the dialogue moves performed in utterances.

As an example, the single user utterance "Lisa" may result in positive feedback on the semantic level but negative on the pragmatic, resulting in a system utterance consisting of two feedback moves and a clarification question: "Lisa. I don't quite understand. Do you want to make a call, change an entry in the phonebook, or delete an entry from the phonebook?"

4 Multimodal menu-based dialogue

Dico II+ implemented a concept of Multimodal Menu-based Dialogue (MMD). Technologies for MMD in menu-based applications have already been developed for other GoDiS applications (Hjelm et al., 2005) and the ideas behind these solutions were re-implemented and significantly improved in Dico.

A common argument for using spoken interaction in a car context is that the driver should be able to use a system without looking at a screen. However, there are many situations where current technology requires the user to look at a screen at some point in the interaction. The idea behind MMD is that the user should be able to switch between and combine modalities freely across and within utterances. This makes it possible to use the system using speech only, using traditional GUI interaction only, or using a combination of the two.

MMD enables *integrated multimodality* for user input, meaning that a single contribution can use several input modalities, e.g. *"Call this contact [click]"* where the [click] symbolises haptic input (e.g. a mouse click) which in this case selects a specific contact. For output, MMD uses *parallel mul-*

timodality, i.e., output is generally rendered both as speech and as GUI output. To use speech only, the user can merely ignore the graphical output and not use the haptic input device. To enable interaction using GUI only, speech input and output can be turned on or off using a button which toggles between "speech on" and "speech off" mode.

The GUI used in Dico II+ is a generic graphical interface for the GoDiS system, developed by Talkamatic AB with graphical adaptations for Dico. It represents GoDiS dialogue moves graphically as menus using a refined version of the conversion schema presented in (Larsson et al., 2001) . For example, alternative questions are represented as multiple choice menus, and wh-questions are represented as scrollable lists. Conversely, haptic user input from the GUI is interpreted as dialogue moves. Selecting an action in a multiple-choice menu corresponds to making a *request* move, and selecting an item in a scrollable list corresponds to an *answer* move.

5 Speech Cursor

This section describes an important addition to the GoDiS dialogue manager and Dico demonstrator, which enables the user to use spoken interaction in combination with haptic input to access all functionality (including browsing long lists) without ever having to look at the screen. In combination with the flexible dialogue capabilities of the GoDiS dialogue manager, and the concept of MMD, we believe that a Speech Cursor provides a powerful and user-friendly way of interacting with menu-based interfaces in cognitively demanding environments such as the in-vehicle environment.

5.1 The problem

A common argument for using spoken interaction in a car context is that the driver should be able to use a system without looking at a screen. However, there are many situations where current technology requires the user to look at a screen at some point in the interaction. This was true also for Dico II in the case of browsing lists; for example, to find out which contacts were listed in the phonebook, the user would at some point have to look at the screen.

Imagine that the user wants to select a song from a song database, and that the user has made restrictions filtering out 30 songs from the database. The dialogue system asks the user which of the songs she wants to hear displaying them in a list on the screen.

The user must now either look at the screen and use a scroll-wheel or similar to select a song, or look at the screen to see which songs are available, and then speak the proper song title. This means that part of the point of using spoken interaction in the car is lost. The example discusses car use, but is applicable any time when the user cannot or does not want to look at a screen, for instance when using a cellphone walking in a city, or when using a web application on a portable device.

An existing interaction strategy for addressing the problems of browsing lists is to allow a kind of meta-dialogue, where the system verbally presents a number of items (for instance 5) from the list, then asking the user if she or he would like to hear the subsequent 5 items, until the list has been read in its entirety or until the users responds negatively. While this strategy in principle solves the problem, it is rather time-consuming compared to browsing the list using a screen and a haptic input device (such as a scroll-wheel); this may decrease the perceived usability of the voice interface in comparison with traditional GUI-based interaction.

Some existing voice interaction systems use a technology to establish understanding which consists of displaying the top N best recognition hypotheses to the user, each one associated with a number, together with a verbal request to the user to say the number corresponding to the desired result. This situation, however, requires the user to look at a screen, and is arguably quite unnatural.

5.2 The solution: Speech Cursor

Dico II+ requires a haptic menu navigation device, such as a mouse (trackball, touch pad, TrackPoint^{TM}) with buttons, pointers and drivers, keyboard with arrow keys, or jog dial/shuttle wheel. A typical in-vehicle menu navigation device consists of three or four buttons (UP, DOWN, OK and possible BACK).

Every time a new item gets focus, the system reads out a voice icon - a spoken representation of the item. This representation can be textual, intended to be realised using a TTS, or in the form

of audio data, to be played directly. Every time a new element gets focus, all any ongoing voice output is interrupted by the voice icon for the element in focus.

6 Combining flexible dialogue and Speech Cursor

Combining flexible dialogue processing with the Speech Cursor concept[1] enables interaction where the user may freely choose between using domain-level spoken utterances (requests, confirmations, questions, answers etc.) and using the Speech Cursor. This is an improvement over existing technology in that it offers a greater variety of interaction styles which can be freely chosen and combined by the user, as well as offering flexible dialogue interaction. Specifically, it provides flexible spoken dialogue interaction in combination with user-friendly browsing of long lists without the need for looking at the screen.

Here is a walk-through of a sample interaction:

Ex. 4:

U: "I want to listen to Madonna"*(This utterance uses accommodation to allow the user to supply unrequested information)*

S: "There are 3 songs by Madonna. What song do you want?" *(Showing list of all songs by Madonna: 'Like a Prayer', 'La Isla Bonita', 'Music', ...)*

U: [DOWN]

S: "Like a Prayer from the album Like a Prayer" *("Like a Prayer" is highlighted)*

U: [DOWN]

S: "La Isla Bonita from the album True Blue" *("La Isla Bonita" is highlighted)*

U: [DOWN]

S: "Music from the alb+" *("Music" is highlighted)*

U: [UP]

S: "Like a Prayer"

U: [OK]

S: "OK, playing Like a Prayer."

Here is a further example:

Ex. 5:

U: "I want to add an ABBA song"

S: "What album?" *(Shows "Waterloo" and "Arrival)*

U: [DOWN]

S: "Wat..."

U: [DOWN]

S: "Arrival"

U: [OK]

S: "what song?" *(Shows "Mamma Mia" and "Money Money Money")*

U: "Mamma Mia"

A promotional demo film showing interactions similar to the above is available[2], courtesy of Talkamatic AB.

Acknowledgments

The work reported here was funded DICO, Vinnova project P28536-1. Additional funding was provided by Talkamatic AB, and by CLT, the Centre of Language Technology at the University of Gothenburg. Dico II+ was implemented by the authors, Fredrik Kronlid, Peter Ljunglöf and Andreas Wiberg. The authors gratefully acknowledge the assistance of Volvo Technology AB and the DICO project group. The GoDiS system is the property of Talkamatic AB.

References

David Hjelm, Ann-Charlotte Forslund, Staffan Larsson, and Andreas Wallentin. 2005. DJ GoDiS: Multimodal menu-based dialogue in an asychronous isu system. In Claire Gardent and Bertrand Gaiffe, editors, *Proceedings of the ninth workshop on the semantics and pragmatics of dialogue*.

[1] The combination of a (simple or flexible) spoken dialogue system and the voice cursor technology is patent pending.

[2] www.youtube.com/watch?v=yvLcQOeBAJE

Staffan Larsson and Jessica Villing. 2009. Multimodal menu-based dialogue in dico ii. In Jens Edlund, Joakim Gustafson, Anna Hjalmarsson, and Gabriel Skantze, editors, *Proceedings of DiaHolmia, 2009 Workshop on the Semantics and Pragmatics of Dialogue*.

Staffan Larsson, Robin Cooper, and Stina Ericsson. 2001. menu2dialog. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 41–45.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, Göteborg University.

Anna Olsson and Jessica Villing. 2005. Dico - a dialogue system for a cell phone. Master's thesis, Department of Linguistics, Goteborg University.

David Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In Ronnie Smith and Jan Kuppevelt, editors, *Current and New Directions in Discourse & Dialogue*. Kluwer Academic Publishers.

Jessica Villing and Staffan Larsson. 2006. Dico - a multimodal in-vehicle dialogue system. In D. Schlangen and R. Fernandez, editors, *Proceedings of the 10th workshop on the semantics and pragmatics of dialogue*.

Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia's Edit History

Oliver Ferschke, Torsten Zesch, and Iryna Gurevych
Ubiquitous Knowledge Processing Lab
Computer Science Department, Technische Universität Darmstadt
Hochschulstrasse 10, D-64289 Darmstadt, Germany
http://www.ukp.tu-darmstadt.de

Abstract

We present an open-source toolkit which allows (i) to reconstruct past states of Wikipedia, and (ii) to efficiently access the edit history of Wikipedia articles. Reconstructing past states of Wikipedia is a prerequisite for reproducing previous experimental work based on Wikipedia. Beyond that, the edit history of Wikipedia articles has been shown to be a valuable knowledge source for NLP, but access is severely impeded by the lack of efficient tools for managing the huge amount of provided data. By using a dedicated storage format, our toolkit massively decreases the data volume to less than 2% of the original size, and at the same time provides an easy-to-use interface to access the revision data. The language-independent design allows to process any language represented in Wikipedia. We expect this work to consolidate NLP research using Wikipedia in general, and to foster research making use of the knowledge encoded in Wikipedia's edit history.

1 Introduction

In the last decade, the free encyclopedia Wikipedia has become one of the most valuable and comprehensive knowledge sources in Natural Language Processing. It has been used for numerous NLP tasks, e.g. word sense disambiguation, semantic relatedness measures, or text categorization. A detailed survey on usages of Wikipedia in NLP can be found in (Medelyan et al., 2009).

The majority of Wikipedia-based NLP algorithms works on single snapshots of Wikipedia, which are published by the Wikimedia Foundation as XML dumps at irregular intervals.[1] Such a snapshot only represents the state of Wikipedia at a certain fixed point in time, while Wikipedia actually is a dynamic resource that is constantly changed by its millions of editors. This rapid change is bound to have an influence on the performance of NLP algorithms using Wikipedia data. However, the exact consequences are largely unknown, as only very few papers have systematically analyzed this influence (Zesch and Gurevych, 2010). This is mainly due to older snapshots becoming unavailable, as there is no official backup server. As a consequence, older experimental results cannot be reproduced anymore.

In this paper, we present a toolkit that solves both issues by reconstructing a certain past state of Wikipedia from its edit history, which is offered by the Wikimedia Foundation in form of a database dump. Section 3 gives a more detailed overview of the reconstruction process.

Besides reconstructing past states of Wikipedia, the revision history data also constitutes a novel knowledge source for NLP algorithms. The sequence of article edits can be used as training data for data-driven NLP algorithms, such as vandalism detection (Chin et al., 2010), text summarization (Nelken and Yamangil, 2008), sentence compression (Yamangil and Nelken, 2008), unsupervised extraction of lexical simplifications (Yatskar et al., 2010), the expansion of textual entailment corpora (Zanzotto and Pennacchiotti, 2010), or assesing the trustworthiness of Wikipedia articles (Zeng et al., 2006).

[1] http://download.wikimedia.org/

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 97–102,
Portland, Oregon, USA, 21 June 2011.

However, efficient access to this new resource has been limited by the immense size of the data. The revisions for all articles in the current English Wikipedia sum up to over 5 terabytes of text. Consequently, most of the above mentioned previous work only regarded small samples of the available data. However, using more data usually leads to better results, or how Church and Mercer (1993) put it "more data are better data". Thus, in Section 4, we present a tool to efficiently access Wikipedia's edit history. It provides an easy-to-use API for programmatically accessing the revision data and reduces the required storage space to less than 2% of its original size. Both tools are publicly available on Google Code (`http://jwpl.googlecode.com`) as open source software under the *LGPL v3*.

2 Related Work

To our knowledge, there are currently only two alternatives to programmatically access Wikipedia's revision history.

One possibility is to manually parse the original XML revision dump. However, due to the huge size of these dumps, efficient, random access is infeasible with this approach.

Another possibility is using the *MediaWiki API*[2], a web service which directly accesses live data from the Wikipedia website. However, using a web service entails that the desired revision for every single article has to be requested from the service, transferred over the Internet and then stored locally in an appropriate format. Access to all revisions of all Wikipedia articles for a large-scale analysis is infeasible with this method because it is strongly constricted by the data transfer speed over the Internet. Even though it is possible to bypass this bottleneck by setting up a local Wikipedia mirror, the MediaWiki API can only provide full text revisions, which results in very large amounts of data to be transferred.

Better suited for tasks of this kind are APIs that utilize databases for storing and accessing the Wikipedia data. However, current database-driven Wikipedia APIs do not support access to article revisions. That is why we decided to extend an established API with the ability to efficiently access Wikipedia's edit history. Two established Wikipedia APIs have been considered for this purpose.

Wikipedia Miner[3] (Milne and Witten, 2009) is an open source toolkit which provides access to Wikipedia with the help of a preprocessed database. It represents articles, categories and redirects as Java classes and provides access to the article content either as MediaWiki markup or as plain text. The toolkit mainly focuses on Wikipedia's structure, the contained concepts, and semantic relations, but it makes little use of the textual content within the articles. Even though it was developed to work language independently, it focuses mainly on the English Wikipedia.

Another open source API for accessing Wikipedia data from a preprocessed database is *JWPL*[4] (Zesch et al., 2008). Like Wikipedia Miner, it also represents the content and structure of Wikipedia as Java objects. In addition to that, JWPL contains a MediaWiki markup parser to further analyze the article contents to make available fine-grained information like e.g. article sections, info-boxes, or first paragraphs. Furthermore, it was explicitly designed to work with all language versions of Wikipedia.

We have chosen to extend JWPL with our revision toolkit, as it has better support for accessing article contents, natively supports multiple languages, and seems to have a larger and more active developer community. In the following section, we present the parts of the toolkit which reconstruct past states of Wikipedia, while in section 4, we describe tools allowing to efficiently access Wikipedia's edit history.

3 Reconstructing Past States of Wikipedia

Access to arbitrary past states of Wikipedia is required to (i) evaluate the performance of Wikipedia-based NLP algorithms over time, and (ii) to reproduce Wikipedia-based research results. For this reason, we have developed a tool called *TimeMachine*, which addresses both of these issues by making use of the revision dump provided by the Wikimedia Foundation. By iterating over all articles in the revision dump and extracting the desired revision of each article, it is possible to recover the state of Wikipedia at an earlier point in time.

[2] `http://www.mediawiki.org/wiki/API`

[3] `http://wikipedia-miner.sourceforge.net`

[4] `http://jwpl.googlecode.com`

Property	Description	Example Value
language	The Wikipedia language version	`english`
mainCategory	Title of the main category of the Wikipedia language version used	`Categories`
disambiguationCategory	Title of the disambiguation category of the Wikipedia language version used	`Disambiguation`
fromTimestamp	Timestamp of the first snapshot to be extracted	`20090101130000`
toTimestamp	Timestamp of the last snapshot to be extracted	`20091231130000`
each	Interval between snapshots in days	`30`
removeInputFilesAfterProcessing	Remove source files [true/false]	`false`
metaHistoryFile	Path to the revision dump	`PATH/pages-meta-history.xml.bz2`
pageLinksFile	Path to the page-to-page link records	`PATH/pagelinks.sql.gz`
categoryLinksFile	Path to the category membership records	`PATH/categorylinks.sql.gz`
outputDirectory	Output directory	`PATH/outdir/`

Table 1: Configuration of the TimeMachine

The TimeMachine is controlled by a single configuration file, which allows (i) to restore individual Wikipedia snapshots or (ii) to generate whole snapshot series. *Table 1* gives an overview of the configuration parameters. The first three properties set the environment for the specific language version of Wikipedia. The two timestamps define the start and end time of the snapshot series, while the interval between the snapshots in the series is set by the parameter *each*. In the example, the TimeMachine recovers 13 snapshots between Jan 01, 2009 at 01.00 p.m and and Dec 31, 2009 at 01.00 p.m at an interval of 30 days. In order to recover a single snapshot, the two timestamps have simply to be set to the same value, while the parameter '*each*' has no effect. The option *removeInputFilesAfterProcessing* specifies whether to delete the source files after processing has finished. The final four properties define the paths to the source files and the output directory.

The output of the TimeMachine is a set of eleven text files for each snapshot, which can directly be imported into an empty JWPL database. It can be accessed with the JWPL API in the same way as snapshots created using JWPL itself.

Issue of Deleted Articles The past snapshot of Wikipedia created by our toolkit is identical to the state of Wikipedia at that time with the exception of articles that have been deleted meanwhile. Articles might be deleted only by Wikipedia administrators if they are subject to copyright violations, vandalism, spam or other conditions that violate Wikipedia policies. As a consequence, they are removed from the public view along with all their revision information, which makes it impossible to recover them from any future publicly available dump.[5] Even though about five thousand pages are deleted every day, only a small percentage of those pages actually corresponds to meaningful articles. Most of the affected pages are newly created duplicates of already existing articles or spam articles.

4 Efficient Access to Revisions

Even though article revisions are available from the official Wikipedia revision dumps, accessing this information on a large scale is still a difficult task. This is due to two main problems. First, the revision dump contains all revisions as full text. This results in a massive amount of data and makes structured access very hard. Second, there is no efficient API available so far for accessing article revisions on a large scale.

Thus, we have developed a tool called *RevisionMachine*, which solves these issues. First, we describe our solution to the storage problem. Second, we present several use cases of the RevisionMachine, and show how the API simplifies experimental setups.

[5] `http://en.wikipedia.org/wiki/Wikipedia:DEL`

4.1 Revision Storage

As each revision of a Wikipedia article stores the full article text, the revision history obviously contains a lot of redundant data. The RevisionMachine makes use of this fact and utilizes a dedicated storage format which stores a revision only by means of the changes that have been made to the previous revision. For this purpose, we have tested existing diff libraries, like Javaxdelta[6] or java-diff[7], which calculate the differences between two texts. However, both their runtime and the size of the resulting output was not feasible for the given size of the data. Therefore, we have developed our own diff algorithm, which is based on a *longest common substring search* and constitutes the foundation for our revision storage format.

The processing of two subsequent revisions can be divided into four steps:

- First, the RevisionMachine searches for all common substrings with a user-defined minimal length.
- Then, the revisions are divided into blocks of equal length. Corresponding blocks of both revisions are then compared. If a block is contained in one of the common substrings, it can be marked as *unchanged*. Otherwise, we have to categorize the kind of change that occurred in this block. We differentiate between five possible actions: `Insert`, `Delete`, `Replace`, `Cut` and `Paste`[8]. This information is stored in each block and is later on used to encode the revision.
- In the next step, the current revision is represented by means of a sequence of actions performed on the previous revision.
  For example, in the adjacent revision pair
  r_1 : This is the very first sentence!
  r_2 : This is the second sentence
  r_2 can be encoded as
  `REPLACE 12 10 'second'`
  `DELETE 31 1`
- Finally, the string representation of this action sequence is compressed and stored in the database.

With this approach, we achieve to reduce the demand for disk space for a recent English Wikipedia dump containing all article revisions from 5470 GB to only 96 GB, i.e. by 98%. The compressed data is stored in a MySQL database, which provides sophisticated indexing mechanisms for high-performance access to the data.

Obviously, storing only the changes instead of the full text of each revision trades in speed for space. Accessing a certain revision now requires reconstructing the text of the revision from a list of changes. As articles often have several thousand revisions, this might take too long. Thus, in order to speed up the recovery of the revision text, every n-th revision is stored as a full revision. A low value of n decreases the time needed to access a certain revision, but increases the demand for storage space. We have found $n = 1000$ to yield a good trade-off[9]. This parameter, among a few other possibilities to fine-tune the process, can be set in a graphical user interface provided with the RevisionMachine.

4.2 Revision Access

After the converted revisions have been stored in the revision database, it can either be used standalone or combined with the JWPL data and accessed via the standard JWPL API. The latter option makes it possible to combine the possibilities of the RevisionMachine with other components like the JWPL parser for the MediaWiki syntax.

In order to set up the RevisionMachine, it is only necessary to provide the configuration details for the database connection (see *Listing 1*). Upon first access, the database user has to have write permission on the database, as indexes have to be created. For later use, read permission is sufficient. Access to the RevisionMachine is achieved via two API objects. The *RevisionIterator* allows to iterate over all revisions in Wikipedia. The *RevisionAPI* grants access to the revisions of individual articles. In addition to

[6] `http://javaxdelta.sourceforge.net/`

[7] `http://www.incava.org/projects/java/java-diff`

[8] `Cut` and `Paste` operations always occur pairwise. In addition to the other operations, they can make use of an additional temporary storage register to save the text that is being moved.

[9] If hard disk space is no limiting factor, the parameter can be set to 1 to avoid the compression of the revisions and maximize the performance.

```
//Set up database connection
DatabaseConfiguration db = new DatabaseConfiguration();
db.setDatabase("dbname");
db.setHost("hostname");
db.setUser("username");
db.setPassword("pwd");
db.setLanguage(Language.english);
//Create API objects
Wikipedia wiki = WikiConnectionUtils.getWikipediaConnection(db);
RevisionIterator revIt = new RevisionIterator(db);
RevisionApi revApi = new RevisionApi(db);
```

Listing 1: Setting up the RevisionMachine

that, the *Wikipedia* object provides access to JWPL functionalities.

In the following, we describe three use cases of the RevisionMachine API, which demonstrate how it is easily integrated into experimental setups.

Processing all article revisions in Wikipedia The first use case focuses on the utilization of the complete set of article revisions in a Wikipedia snapshot. *Listing 2* shows how to iterate over all revisions. Thereby, the iterator ensures that successive revisions always correspond to adjacent revisions of a single article in chronological order. The start of a new article can easily be detected by checking the timestamp and the article id. This approach is especially useful for applications in statistical natural language processing, where large amounts of training data are a vital asset.

Processing revisions of individual articles The second use case shows how the RevisionMachine can be used to access the edit history of a specific article. The example in *Listing 3* illustrates how all revisions for the article *Automobile* can be retrieved by first performing a page query with the JWPL API and then retrieving all revision timestamps for this page, which can finally be used to access the revision objects.

Accessing the meta data of a revision The third use case illustrates the access to the meta data of individual revisions. The meta data includes the name or IP of the contributor, the additional user comment for the revision and a flag that identifies a revision as minor or major. *Listing 4* shows how the number of edits and unique contributors can be used to indicate the level of edit activity for an article.

5 Conclusions

In this paper, we presented an open-source toolkit which extends *JWPL*, an API for accessing Wikipedia, with the ability to reconstruct past states of Wikipedia, and to efficiently access the edit history of Wikipedia articles.

Reconstructing past states of Wikipedia is a prerequisite for reproducing previous experimental work based on Wikipedia, and is also a requirement for the creation of time-based series of Wikipedia snapshots and for assessing the influence of Wikipedia growth on NLP algorithms. Furthermore, Wikipedia's edit history has been shown to be a valuable knowledge source for NLP, which is hard to access because of the lack of efficient tools for managing the huge amount of revision data. By utilizing a dedicated storage format for the revisions, our toolkit massively decreases the amount of data to be stored. At the same time, it provides an easy-to-use interface to access the revision data.

We expect this work to consolidate NLP research using Wikipedia in general, and to foster research making use of the knowledge encoded in Wikipedia's edit history. The toolkit will be made available as part of *JWPL*, and can be obtained from the project's website at Google Code. (`http://jwpl.googlecode.com`)

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz" (*LOEWE*) as part of the research center "Digital Humanities". We would also like to thank Simon Kulessa for designing and implementing the foundations of the RevisionMachine.

```
//Iterate over all revisions of all articles
while (revIt.hasNext()) {
    Revision rev = revIt.next()
    rev.getTimestamp();
    rev.getArticleID();
    //process revision ...
}
```

Listing 2: Iteration over all revisions of all articles

```
//Get article with title "Automobile"
Page article = wiki.getPage("Automobile");
int id = article.getPageId();
//Get all revisions for the article
Collection<Timestamp> revisionTimeStamps = revApi.getRevisionTimestamps(id);
for (Timestamp t:revisionTimeStamps) {
    Revision rev = revApi.getRevision(id, t);
    //process revision ...
}
```

Listing 3: Accessing the revisions of a specific article

```
//Meta data provided by the RevisionAPI
StringBuffer s = new StringBuffer();
s.append("The article has "+revApi.getNumberOfRevisions(pageId)+" revisions.\n");
s.append("It has "+revApi.getNumberOfUniqueContributors(pageId)+" unique contributors.\n");
s.append(revApi.getNumberOfUniqueContributors(pageId,true)+ " are registered users.\n");
//Meta data provided by the Revision object
s.append((rev.isMinor()?"Minor":"Major")+" revision by: "+rev.getContributorID());
s.append("\nComment: "+rev.getComment());
```

Listing 4: Accessing the meta data of a revision

References

Si-Chi Chin, W. Nick Street, Padmini Srinivasan, and David Eichmann. 2010. Detecting wikipedia vandalism with active learning and statistical language models. In *Proceedings of the 4th workshop on Information credibility*, WICOW '10, pages 3–10.

Kenneth W. Church and Robert L. Mercer. 1993. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24.

Olena Medelyan, David Milne, Catherine Legg, and Ian H. Witten. 2009. Mining meaning from wikipedia. *Int. J. Hum.-Comput. Stud.*, 67:716–754, September.

D. Milne and I. H. Witten. 2009. An open-source toolkit for mining Wikipedia. In *Proc. New Zealand Computer Science Research Student Conf.*, volume 9.

Rani Nelken and Elif Yamangil. 2008. Mining wikipedia's article revision history for training computational linguistics algorithms. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy (WikiAI)*, WikiAI08.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*, pages 137–140, Columbus, Ohio, June. Association for Computational Linguistics.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 365–368.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from wikipedia using co-training. In *Proceedings of the COLING-Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*.

Honglei Zeng, Maher Alhossaini, Li Ding, Richard Fikes, and Deborah L. McGuinness. 2006. Computing trust from revision history. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust*.

Torsten Zesch and Iryna Gurevych. 2010. The more the better? Assessing the influence of wikipedia's growth on semantic relatedness measures. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Torsten Zesch, Christof Mueller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*.

An Efficient Indexer for Large N-Gram Corpora

Hakan Ceylan
Department of Computer Science
University of North Texas
Denton, TX 76203
hakan@unt.edu

Rada Mihalcea
Department of Computer Science
University of North Texas
Denton, TX 76203
rada@cs.unt.edu

Abstract

We introduce a new publicly available tool that implements efficient indexing and retrieval of large N-gram datasets, such as the Web1T 5-gram corpus. Our tool indexes the entire Web1T dataset with an index size of only 100 MB and performs a retrieval of any N-gram with a single disk access. With an increased index size of 420 MB and duplicate data, it also allows users to issue wild card queries provided that the wild cards in the query are contiguous. Furthermore, we also implement some of the smoothing algorithms that are designed specifically for large datasets and are shown to yield better language models than the traditional ones on the Web1T 5-gram corpus (Yuret, 2008). We demonstrate the effectiveness of our tool and the smoothing algorithms on the English Lexical Substitution task by a simple implementation that gives considerable improvement over a basic language model.

1 Introduction

The goal of statistical language modeling is to capture the properties of a language through a probability distribution so that the probabilities of word sequences can be estimated. Since the probability distribution is built from a corpus of the language by computing the frequencies of the N-grams found in the corpus, the data sparsity is always an issue with the language models. Hence, as it is the case with many statistical models used in Natural Language Processing (NLP), the models give a much better performance with larger data sets.

However the large data sets, such as the Web1T 5-Gram corpus of (Brants and Franz, 2006), present a major challenge. The language models built from these sets cannot fit in memory, hence efficient accessing of the N-gram frequencies becomes an issue. Trivial methods such as linear or binary search over the entire dataset in order to access a single N-gram prove inefficient, as even a binary search over a single file of 10,000,000 records, which is the case of the Web1T corpus, requires in the worst case $\lceil log_2(10,000,000) \rceil = 24$ accesses to the disk drive.

Since the access to N-grams is costly for these large data sets, the implementation of further improvements such as smoothing algorithms becomes impractical. In this paper, we overcome this problem by implementing a novel, publicly available tool[1] that employs an indexing strategy that reduces the access time to any N-gram in the Web1T corpus to a single disk access. We also make a second contribution by implementing some of the smoothing models that take into account the size of the dataset, and are shown to yield up to 31% perplexity reduction on the Brown corpus (Yuret, 2008). Our implementation is space efficient, and provides a fast access to both the N-gram frequencies, as well as their smoothed probabilities.

2 Related Work

Language modeling toolkits are used extensively for speech processing, machine translation, and many other NLP applications. The two of the most popular toolkits that are also freely available are the *CMU Statistical Language Modeling (SLM) Toolkit* (Clarkson and Rosenfeld, 1997), and the *SRI Language Modeling Toolkit* (Stolcke, 2002). However,

[1] Our tool can be freely downloaded from the download section under http://lit.csci.unt.edu

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 103–108,
Portland, Oregon, USA, 21 June 2011.

even though these tools represent a great resource for building language models and applying them to various problems, they are not designed for very large corpora, such as the Web1T 5-gram corpus (Brants and Franz, 2006), hence they do not provide efficient implementations to access these data sets.

Furthermore, (Yuret, 2008) has recently shown that the widely popular smoothing algorithms for language models such as *Kneser-Ney* (Kneser and Ney, 1995), *Witten-Bell* (Witten and Bell, 1991), or *Absolute Discounting* do not realize the full potentials of very large corpora, which often come with missing counts. The reason for the missing counts is due to the omission of low frequency N-grams in the corpus. (Yuret, 2008) shows that with a modified version of Kneser-Ney smoothing algorithm, named as the Dirichlet-Kneser-Ney, a 31% reduction in perplexity can be obtained on the Brown corpus.

A tool similar to ours that uses a hashing technique in order to provide a fast access to the Web1T corpus is presented in detail in (Hawker et al., 2007). The tool provides access to queries with wild card symbols, and the performance of the tool on 10^6 queries on a 2.66 GHz processor with 1.5 GBytes of memory is given approximately as one hour. Another tool, *Web1T5-Easy*, described in (Evert, 2010), provides indexing of the Web1T corpus via relational database tables implemented in an SQLite engine. It allows interactive searches on the corpus as well as collocation discovery. The indexing time of this tool is reported to be two weeks, while the non-cached retrieval time is given to be in order of a few seconds. Other tools that implement a binary search algorithm as a simpler, yet less efficient method are also given in (Giuliano et al., 2007; Yuret, 2007).

3 The Web1T 5-gram Corpus

The Web1T 5-gram corpus (Brants and Franz, 2006) consists of sequences of words (N-grams) and their associated counts extracted from a Web corpus of approximately one trillion words. The length of each sequence, N, ranges from 1 to 5, and the size of the entire corpus is approximately 88GB (25GB in compressed form). The unigrams form the vocabulary of the corpus and are stored in a single file which includes around 13 million tokens and their associated counts. The remaining N-grams are stored separately across multiple files in lexicographic order. For example, there are 977,069,902 distinct trigrams in the dataset, and they are stored consecutively in 98 files in lexicographic order. Furthermore, each N-gram file contains 10,000,000 N-grams except the last one, which contains less. It is also important to note that N-grams with counts less than 40 are excluded from the dataset for $N = 2, 3, 4, 5$, and the tokens with less than 200 are excluded from the unigrams.

4 The Indexer

4.1 B$^+$-trees

We used a B$^+$-tree structure for indexing. A B$^+$-tree is essentially a balanced search tree where each node has several children. Indexing large files using B$^+$ trees is a popular technique implemented by most database systems today as the underlying structure for efficient range queries. Although many variations of B$^+$-trees exist, we use the definition for primary indexing given in (Salzberg, 1988). Therefore we assume that the data, which is composed of records, is only stored in the leaves of the tree and the internal nodes store only the keys.

The data in the leaves of a B$^+$-tree is grouped into *buckets*, where the size of a bucket is determined by a bucket factor parameter, *bkfr*. Therefore at any given time, each bucket can hold a number of records in the range $[1, bkfr]$. Similarly, the number of keys that each internal node can hold is determined by the *order* parameter, *v*. By definition, each internal node except the root can have any number of keys in the range $[v, 2v]$, and the root must have at least one key. Finally, an internal node with k keys has $k + 1$ children.

4.2 Mapping Unigrams to Integer Keys

A key in a B$^+$-tree is a lookup value for a record, and a record in our case is an N-gram together with its count. Therefore each line of an N-gram file in the Web1T dataset makes up a record. Since each N-gram is distinct, it is possible to use the N-gram itself as a key. However in order to reduce the storage requirements and make the comparisons faster during a lookup, we map each unigram to an integer, and form the keys of the records using the integer values instead of the tokens themselves.[2]

To map unigrams to integers, we use the unigrams sorted in lexicographic order and assign an integer value to each unigram starting from 1. In other words, if we let the m-tuple $U = (t_1, t_2, ..., t_m)$ represent all the unigrams sorted in lexicographic order,

[2]This method does not give optimal storage, for which one should implement a compression Huffman coding scheme.

then for a unigram t_i, i gives its key value. The key of trigram "$t_i\ t_j\ t_k$" is simply given as "$i\ j\ k$." Thus, the comparison of two keys can be done in a similar fashion to the comparison of two N-grams; we first compare the first integer of each key, and in case of equality, we compare the second integers, and so on. We stop the comparison as soon as an inequality is found. If all the comparisons result in equality then the two keys (N-grams) are equal.

4.3 Searching for a Record

We construct a B$^+$-tree for each N-gram file in the dataset for $N = 2, 3, 4, 5$, and keep the key of the first N-gram for each file in memory. When a query q is issued, we first find the file that contains q by comparing the key of q to the keys in memory. Since this is an in-memory operation, it can be simply done by performing a binary search. Once the correct file is found, we then search the B$^+$-tree constructed for that file for the N-gram q by using its key.

As is the case with any binary search tree, a search in a B$^+$-tree starts at the root level and ends in the leaves. If we let r_i and p_j represent a key and a pointer to the child of an internal node respectively, for $i = 1, 2, ..., k$ and $j = 1, 2, ..., k + 1$, then to search an internal node, including the root, for a key q, we first find the key r_m that satisfies one of the following:

- $(q < r_m) \wedge (m = 1)$
- $(r_{m-1} \leq q) \wedge (r_m > q)$ for $1 < m \leq k$
- $(q > r_m) \wedge (m = k)$

If one of the first two cases is satisfied, the search continues on the child node found by following p_m, whereas if the last condition is satisfied, the pointer p_{m+1} is followed. Since the keys in an internal node are sorted, a binary search can be performed to find r_m. Finally, when a leaf node is reached, the entire bucket is read into memory first, then a record with a key value of q is searched.

4.4 Constructing a B$^+$-tree

The construction of a B$^+$-tree is performed through successive record insertions.[3] Given a record, we first compute its key, find the leaf node it is supposed to be in, and insert it if the bucket is not full. Otherwise, the leaf node is split into two nodes, each containing $\lceil bkfr/2 \rceil$, and $\lfloor bkfr/2 \rfloor + 1$ records, and the first key of the node containing the larger key values is placed into the parent internal node together with the node's pointer. The insertion of a key to an internal node is similar, only this time both split nodes contain v values, and the middle key value is sent up to the parent node.

Note that not all the internal nodes of a B$^+$-tree have to be kept on the disk, and read from there each time we do a search. In practice, all but the last two levels of a B$^+$-tree are placed in memory. The reason for this is the high branching factor of the B$^+$-trees together with their effective storage utilization. It has been shown in (Yao, 1978) that the nodes of a high-order B$^+$-tree are $ln2 \approx 69\%$ full on average.

However, note that the tree will be fixed in our case, i.e., once it is constructed we will not be inserting any other N-gram records. Therefore we do not need to worry about the 69% space utilization, but instead try to make each bucket, and each internal node full. Thus, with a $bkfr = 1250$, and $v = 100$, an N-gram file with 10,000,000 records would have 8,000 leaf nodes on level 3, 40 internal nodes on level 2, and the root node on level 1. Furthermore, let us assume that integers, disk and memory pointers all hold 8 bytes of space. Therefore a 5-gram key would require 40 bytes, and a full internal node in level 2 would require $(200x40) + (201x8) = 9,608$ bytes. Thus the level 2 would require $9,608x40 \approx 384$ Kbytes, and level 1 would require $(40 * 40) + (41 * 8) = 1,928$ bytes. Hence, a Web1T 5-gram file, which has an average size of 286 MB can be indexed with approximately 386 Kbytes. There are 118 5-gram files in the Web1T dataset, so we would need $386x118 \approx 46$ MBytes of memory space in order to index all of them. A similar calculation for 4-grams, trigrams, and bigrams for which the bucket factor values are selected as 1600, 2000, and 2500 respectively, shows that the entire Web1T corpus, except unigrams, can be indexed with approximately 100 MBytes, all of which can be kept in memory, thereby reducing the disk access to only one. As a final note, in order to compute a key for a given N-gram quickly, we keep the unigrams in memory, and use a hashing scheme for mapping tokens to integers, which additionally require 178 Mbytes of memory space.

The choice of the bucket factor and the inter-

[3] Note that this may cause efficiency issues for very large files as memory might become full during the construction process, hence in practice, the file is usually sorted prior to indexing.

nal node order parameters depend on the hard-disk speed, and the available memory.[4]. Recall that even to fetch a single N-gram record from the disk, the entire bucket needs to be read. Therefore as the bucket factor parameter is reduced, the size of the index will grow, but the access time would be faster as long as the index could be entirely fit in memory. On the other hand, with a too large bucket factor, although the index can be made smaller, thereby reducing the memory requirements, the access time may be unacceptable for the application. Note that a random reading of a bucket of records from the hard-disk requires the disk head to first go to the location of the first record, and then do a sequential read.[5] Assuming a hard-disk having an average transfer rate of 100 MBytes, once the disk head finds the correct location, a 40 bytes N-gram record can be read in $4x10^{-7}$ seconds. Thus, assuming a seek time around 8-10 ms, even with a bucket factor of 1,000, it can be seen that the seek time is still the dominating factor. Therefore, as the bucket size gets smaller than 1,000, even though the index size will grow, there would be almost no speed up in the access time, which justifies our parameter choices.

4.5 Handling Wild Card Queries

Having described the indexing scheme, and how to search for a single N-gram record, we now turn our attention to queries including one or more wild card symbols, which in our case is the underscore character "_", as it does not exist among the unigram tokens of the Web1T dataset. We manually add the wild card symbol to our mapping of tokens to integers, and map it to the integer *0*, so that a search for a query with a wild card symbol would be unsuccessful but would point to the first record in the file that replaces the wild card symbol with a real token as the key for the wild card symbol is guaranteed to be the smallest. Having found the first record we perform a sequential read until the last read record does not match the query. The reason this strategy works is because the N-grams are sorted in lexicographic order in the data set, and also when we map unigram tokens to integers, we preserve their order, i.e., the first token in the lexicographically sorted unigram list is assigned the value 1, the second is assigned 2, and so forth. For example, for a given query *Our Honorable _*, the record that would be pointed at the end of search in the trigram file *3gm-0041* is the N-gram *Our Honorable Court 186*, which is the first N-gram in the data set that starts with the bigram *Our Honorable*.

Note however that the methodology that is described to handle the queries with wild card symbols will only work if the wild card symbols are the last tokens of the query and they are contiguous. For example a query such as *Our _ Court* will not work as N-grams satisfying this query are not stored contiguously in the data set. Therefore in order to handle such queries, we need to store additional copies of the N-grams sorted in different orders. When the last occurrence of the contiguous wild card symbols is in position p of a query N-gram for $p = 0, 1, ..., N-1$, then the N-grams sorted lexicographically starting from position $(p+1) mod N$ needs to be searched. A lexicographical sort for a position p, for $0 \leq p \leq (N-1)$ is performed by moving all the tokens in positions $0...(p-1)$ to the end for each N-gram in the data set. Thus, for all the bigrams in the data set, we need one extra copy sorted in position 1, for all the trigrams, we need two extra copies; one sorted in position 1, and another sorted in position 2, and so forth. Hence, in order to handle the contiguous wild card queries in any position, in addition to the 88 GBytes of original Web1T data, we need an extra disk space of 265 GBytes. Furthermore, the indexing cost of the duplicate data is an additional 320 MBytes. Thus, the total disk cost of the system will be approximately 353 GBytes plus the index size of 420 MBytes, and since we keep the entire index in memory, the final memory cost of the system will be 420 MBytes + 178 MBytes = 598 MBytes.

4.6 Performance

Given that today's commodity hardware comes with at least 4 GBytes of memory and 1 TBytes of hard-disk space, the requirements of our tool are reasonable. Furthermore, our tool is implemented in a client-server architecture, and it allows multiple clients to submit multiple queries to the server over a network. The server can be queried with an N-gram query either for its count in the corpus, or its smoothed probability with a given smoothing method. The queries with wild cards can ask for the retrieval of all the N-grams satisfying a query, or only for the total count so the network overhead can

[4]We used a 7200 RPM disk-drive with an average read seek time of 8.5 ms, write seek time of 10.0 ms, and a data transfer time up to 3 GBytes per second.

[5]A rotational latency should also be taken into account before the sequential reading can be done.

be avoided depending on the application needs.

Our program requires about one day of offline processing due to resorting the entire data a few times. Note that some of the files in the corpus need to be sorted as many as four times. For the sorting process, the files are first individually sorted, and then a k-way merge is performed. In our implementation, we used a min heap structure for this purpose, and k is always chosen as the number of files for a given N. The index construction however is relatively fast. It takes about an hour to construct the index for the 5-grams. Once the offline processing is done, it only takes a few minutes to start the server, and from that point the online performance of our tool is very fast. It takes about 1-2 seconds to process 1000 randomly picked 5-gram queries (with no wild card symbols), which may or may not exist in the corpus. For the queries asking for the frequencies only, our tool implements a small caching mechanism that takes the temporal locality into account. The mechanism is very useful for wild card queries involving stop words, such as *"the _"*, and *"of the _"* which occur frequently, and take a long time to process due to the sequential read of a large number of records from the data set.

5 Lexical Substitution

In this section we demonstrate the effectiveness of our tool by using it on the the English Lexical Substitution task, which was first introduced in SemEval 2007 (McCarthy and Navigli, 2007). The task requires both the human annotators and the participating systems to replace a target word in a given sentence with the most appropriate alternatives. The description of the tasks, the data sets, the performance of the participating systems as well as a post analysis of the results is given in (McCarthy and Navigli, 2009).

Although the task includes three subtasks, in this evaluation we are only concerned with one of them, namely the *best* subtask. The best subtask asks the systems and the annotators to provide only one substitute for the target words – the most appropriate one. Two separate datasets were provided with this task: a trial dataset was first provided in order for the participants to get familiar with the task and train their systems. The trial data used a lexical sample of 30 words with 10 instances each. The systems were then tested on a larger test data, which used a lexical sample of 171 words each again having 10 instances.

Our methodology for this task is very simple; we replace the target word with an alternative from a list of candidates, and find the probability of the context with the new word using a language model. The candidate that gives the highest probability is provided as the system's best guess. The list of candidates is obtained from two different lexical sources, WordNet (Fellbaum, 1998) and Roget's Thesaurus (Thesaurus.com, 2007). We retrieve all the synonyms for all the different senses of the word from both resources and combine them. We did not consider any lexical relations other than synonymy, and similarly we did not consider any words at a further semantic distance.

Model	Precision	Mod Precision
No Smoothing	10.13	14.78
Absolute Discounting	11.05	16.75
KN with Missing Counts	11.19	16.75
Dirichlet KN	10.98	15.76

Table 1: Results on the trial data

Model	Precision	Mod Precision
No Smoothing	9.01	14.15
Absolute Discounting	11.64	18.62
KN with Missing Counts	11.61	18.54
Dirichlet KN	11.03	17.48
Best Baseline	9.95	15.28
Best SEMEVAL System	12.90	20.65

Table 2: Results on the test data

We start with a simple language model that calculates the probability of the context of a word, and then continue with three smoothing algorithms discussed in (Yuret, 2008), namely *Absolute Discounting*, *Kneser-Ney with Missing Counts*, and the *Dirichlet-Kneser-Ney Discounting*. Note that all three are interpolated models, i.e., they do not just back-off to a lower order probability when an N-gram is not found, but rather use the higher and lower order probabilities all the time in a weighted fashion.

The results on the trial dataset are shown in Table 1, and the results on the test dataset are shown in Table 2. In all the experiments we use the trigram models, i.e., we keep N fixed to 3. Since our system makes a guess for all the target words in the set, our precision and recall scores, as well as the mod precision and the mod recall scores are the same, so only one from each is shown in the table. Note that the highest achievable score for this task is not 100%, but is restricted by the frequency of the best substitute, and it is given as 46.15%. The highest scoring participating system achieved 12.9%, which

gave a 2.95% improvement over the baseline (Yuret, 2008; McCarthy and Navigli, 2009); the scores obtained by the best SEMEVAL system as well as the best baseline calculated using the synonyms for the first synset in WordNet are also shown in Table 2.

On both the trial and the test data, we see that the interpolated smoothing algorithms consistently improve over the naive language modeling, which is an encouraging result. Perhaps a surprising result for us was the performance of the Dirichlet-Kneser-Ney Smoothing Algorithm, which is shown to give minimum perplexity on the Brown corpus out of the given models. This might suggest that the parameters of the smoothing algorithms need adjustments for each task.

It is important to note that this evaluation is meant as a simple proof of concept to demonstrate the usefulness of our indexing tool. We thus used a very simple approach for lexical substitution, and did not attempt to integrate several lexical resources and more sophisticated algorithms, as some of the best scoring systems did. Despite this, the performance of our system exceeds the best baseline, and is better than five out of the eight participating systems (see (McCarthy and Navigli, 2007)).

6 Conclusions

In this paper we described a new publicly available tool that provides fast access to large N-gram datasets with modest hardware requirements. In addition to providing access to individual N-gram records, our tool also handles queries with wild card symbols, provided that the wild cards in the query are contiguous. Furthermore, the tool also implements smoothing algorithms that try to overcome the missing counts that are typical to N-gram corpora due to the omission of low frequencies. We tested our tool on the English Lexical Substitution task, and showed that the smoothing algorithms give an improvement over simple language modeling.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation CAREER award #0747340 and IIS awards #0917170 and #1018613. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

T. Brants and A. Franz. 2006. Web 1T 5-gram corpus version 1. *Linguistic Data Consortium*.

P. Clarkson and R. Rosenfeld. 1997. Statistical language modeling using the cmu-cambridge toolkit. In *Proceedings of ESCA Eurospeech*, pages 2707–2710.

S. Evert. 2010. Google web 1t 5-grams made easy (but not for the computer). In *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop*, WAC-6 '10, pages 32–40.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

C. Giuliano, A. Gliozzo, and C. Strapparava. 2007. Fbk-irst: lexical substitution task exploiting domain and syntagmatic coherence. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 145–148.

T. Hawker, M. Gardiner, and A. Bennetts. 2007. Practical queries of a massive n-gram database. In *Proceedings of the Australasian Language Technology Workshop 2007*, pages 40–48, Melbourne, Australia.

R. Kneser and H. Ney. 1995. Improved backing-off for n-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184 vol.1.

D. McCarthy and R. Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53.

D. McCarthy and R. Navigli. 2009. The english lexical substitution task. *Language Resources and Evaluation*, 43:139–159.

B. Salzberg. 1988. *File structures: an analytic approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 2, pages 901–904, Denver, USA.

Thesaurus.com. 2007. Rogets new millennium thesaurus, first edition (v1.3.1).

I. H. Witten and T. C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094.

A. Chi-Chih Yao. 1978. On random 2-3 trees. *Acta Inf.*, 9:159–170.

D. Yuret. 2007. Ku: word sense disambiguation by substitution. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 207–213.

D. Yuret. 2008. Smoothing a tera-word language model. In *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pages 141–144.

SystemT: A Declarative Information Extraction System

Yunyao Li
IBM Research - Almaden
650 Harry Road
San Jose, CA 95120
yunyaoli@us.ibm.com

Frederick R. Reiss
IBM Research - Almaden
650 Harry Road
San Jose, CA 95120
frreiss@us.ibm.com

Laura Chiticariu
IBM Research - Almaden
650 Harry Road
San Jose, CA 95120
chiti@us.ibm.com

Abstract

Emerging text-intensive enterprise applications such as social analytics and semantic search pose new challenges of scalability and usability to Information Extraction (IE) systems. This paper presents SystemT, a declarative IE system that addresses these challenges and has been deployed in a wide range of enterprise applications. SystemT facilitates the development of high quality complex annotators by providing a highly expressive language and an advanced development environment. It also includes a cost-based optimizer and a high-performance, flexible runtime with minimum memory footprint. We present SystemT as a useful resource that is freely available, and as an opportunity to promote research in building scalable and usable IE systems.

1 Introduction

Information extraction (IE) refers to the extraction of structured information from text documents. In recent years, text analytics have become the driving force for many emerging enterprise applications such as compliance and data redaction. In addition, the inclusion of text has also been increasingly important for many traditional enterprise applications such as business intelligence. Not surprisingly, the use of information extraction has dramatically increased within the enterprise over the years. While the traditional requirement of extraction quality remains critical, enterprise applications pose several two challenges to IE systems:

1. *Scalability*: Enterprise applications operate over large volumes of data, often orders of magnitude larger than classical IE corpora. An IE system should be able to operate at those scales without compromising its execution efficiency or memory consumption.
2. *Usability*: Building an accurate IE system is an inherently labor intensive process. Therefore, the usability of an enterprise IE system in terms of ease of development and maintenance is crucial for ensuring healthy product cycle and timely handling of customer complains.

Traditionally, IE systems have been built from individual extraction components consisting of rules or machine learning models. These individual components are then connected procedurally in a programming language such as C++, Perl or Java. Such procedural logic towards IE cannot meet the increasing scalability and usability requirements in the enterprise (Doan et al., 2006; Chiticariu et al., 2010a).

Three decades ago, the database community faced similar scalability and expressivity challenges in accessing structured information. The community addressed these problems by introducing a relational algebra formalism and an associated declarative query language SQL. Borrowing ideas from the database community, several systems (Doan and others, 2008; Bohannon and others, 2008; Jain et al., 2009; Krishnamurthy et al., 2008; Wang et al., 2010) have been built in recent years taking an alternative declarative approach to information extraction. Instead of using procedural logic to implement the extraction task, declarative IE systems separate the description of *what* to extract from *how* to extract it, allowing the IE developer to build complex extrac-

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 109–114,
Portland, Oregon, USA, 21 June 2011.

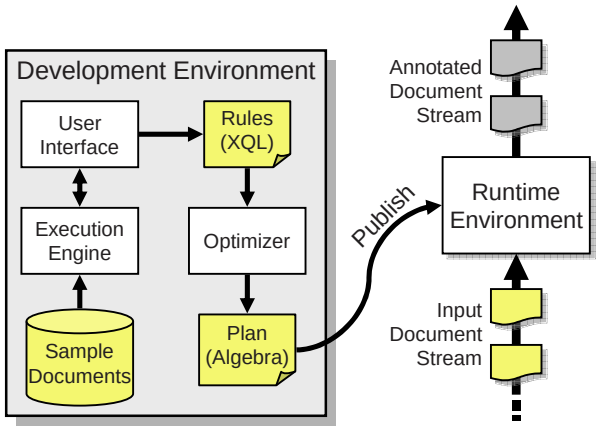


Figure 1: Overview of SystemT

```
create view Phone as
extract regex /\d{3}-\d{4}/ on D.text as number
from Document D;

create view Person as
extract dictionary 'firstNames.dict' on D.text as name
from Document D;

create view PersonPhoneAll as
select CombineSpans(P.name, Ph.number) as match
from Person P, Phone Ph
where FollowsTok(P.name, Ph.number, 0, 5);

create view PersonPhone as
select R.name as name
from PersonPhoneAll R
consolidate on R.name;

output view PersonPhone;
```

Figure 2: An AQL program for a *PersonPhone* task.

tion programs without worrying about performance considerations.

In this demonstration, we showcase one such declarative IE system called SystemT, designed to address the scalability and usability challenges. We illustrate how SystemT, currently deployed in a multitude of real-world applications and commercial products, can be used to develop and maintain IE annotators for enterprise applications. A free version of SystemT is available at *http://www.alphaworks.ibm.com/tech/systemt*.

2 Overview of SystemT

Figure 1 depicts the architecture of SystemT. The system consists of two major components: the Development Environment and the Runtime Environment. The SystemT Development Environment supports the iterative process of constructing and refining rules for information extraction. The rules are specified in a declarative language called AQL (F.Reiss et al., 2008). The Development Environment provides facilities for executing rules over a given corpus of representative documents and visualizing the results of the execution. Once a developer is satisfied with the results that her rules produce on these documents, she can publish her annotator.

Publishing an annotator is a two-step process. First, given an AQL annotator, there can be many possible graphs of *operators*, or execution plans, each of which faithfully implements the semantics of the annotator. Some of the execution plans are much more efficient than others. The SystemT Optimizer explores the space of the possible execution plans to choose the most efficient one. This execution plan is then given to the SystemT Runtime to instantiate the corresponding physical operators. Once the physical operators are instantiated, the SystemT Runtime feeds one document at a time through the graph of physical operators and outputs a stream of annotated documents.

The decoupling of the Development and Runtime environments is essential for the flexibility of the system. It facilitates the incorporating of various sophisticated tools to enable annotator development without sacrificing runtime performance. Furthermore, the separation permits the SystemT Runtime to be embedded into larger applications with minimum memory footprint. Next, we discuss individual components of SystemT in more details (Sections 3 – 6), and summarize our experience with the system in a variety of enterprise applications (Section 7).

3 The Extraction Language

In SystemT, developers express an information extraction program using a language called AQL. AQL is a declarative relational language similar in syntax to the database language SQL, which was chosen as a basis for our language due to its expressivity and familiarity. An AQL program (or an AQL annotator) consists of a set of AQL rules.

In this section, we describe the AQL language and its underlying algebraic operators. In Section 4, we explain how the SystemT optimizer explores a large space of possible execution plans for an AQL annotator and chooses one that is most efficient.

3.1 AQL

Figure 2 illustrates a (very) simplistic annotator of relationships between persons and their phone number. At a high-level, the annotator identifies person names using a simple dictionary of first names, and phone numbers using a regular expression. It then identifies pairs of *Person* and *Phone* annotations, where the latter follows the

former within 0 to 5 tokens, and marks the corresponding region of text as a *PersonPhoneAll* annotation. The final output *PersonPhone* is constructed by removing overlapping *PersonPhoneAll* annotations.

AQL operates over a simple relational data model with three data types: span, tuple, and view. In this data model, a *span* is a region of text within a document identified by its "begin" and "end" positions, while a *tuple* is a list of spans of fixed size. A *view* is a set of tuples. As can be seen from Figure 2, each AQL rule defines a view. As such, a view is the basic building block in AQL: it consists of a logical description of a set of tuples in terms of the document text, or the content of other views. The input to the annotator is a special view called `Document` containing a single tuple with the document text. The AQL annotator tags some views as *output views*, which specify the annotation types that are the final results of the annotator.

The example in Figure 2 illustrates two of the basic constructs of AQL. The `extract` statement specifies basic character-level extraction primitives, such as regular expressions or dictionaries (i.e., gazetteers), that are applied directly to the document, or a region thereof. The `select` statement is similar to the corresponding SQL statement, but contains an additional `consolidate on` clause for resolving overlapping annotations, along with an extensive collection of text-specific predicates.

To keep rules compact, AQL also allows a shorthand *pattern* notation similar to the syntax of the CPSL grammar standard (Appelt and Onyshkevych, 1998). For example, the `PersonPhoneAll` view in Figure 2 can also be expressed as shown below. Internally, SystemT translates each of these *extract pattern* statements into one or more *select* and *extract* statements.

```
create view PersonPhoneAll as
extract pattern
<P.name> <Token>{0,5} <Ph.number>
from Person P, Phone Ph;
```

SystemT has built-in multilingual support including tokenization, part of speech and gazetteer matching for over 20 languages using IBM LanguageWare. Annotator developers can utilize the multilingual support via AQL without having to configure or manage any additional resources. In addition, AQL allows user-defined functions in a restricted context in order to support operations such as validation or normalization. More details on AQL can be found in the AQL manual (Chiticariu et al., 2010b).

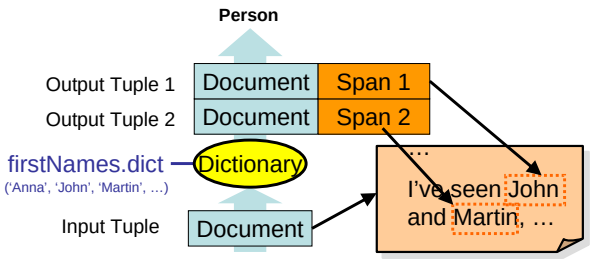


Figure 3: Dictionary Extraction Operator

3.2 Algebraic Operators in SystemT

SystemT executes AQL rules using graphs of operators. These operators are based on an algebraic formalism that is similar to the relational algebra formalism, but with extensions for text processing. Each *operator* in the algebra implements a single basic atomic IE operation, producing and consuming sets of tuples (i.e., views).

Fig. 3 illustrates the dictionary extraction operator in the algebra, which performs character-level dictionary matching. A full description of the 12 different operators of the algebra can be found in (F.Reiss et al., 2008). Three of the operators are listed below.

- The **Extract** operator ($\mathcal{E}$) performs character-level operations such as regular expression and dictionary matching over text, producing one tuple for each match.
- The **Select** operator (σ) takes as input a set of tuples and a predicate to apply to the tuples, and outputs all tuples that satisfy the predicate.
- The **Join** operator ($\bowtie$) takes as input two sets of tuples and a predicate to apply to pairs of tuples. It outputs all pairs satisfying the predicate.

Other operators include **PartOfSpeech** for part-of-speech detection, **Consolidate** for removing overlapping annotations, **Block** and **Group** for grouping together similar annotations occurring within close proximity to each other, as well as expressing more general types of aggregation, **Sort** for sorting, and **Union** and **Minus** for expressing set union and set difference, respectively.

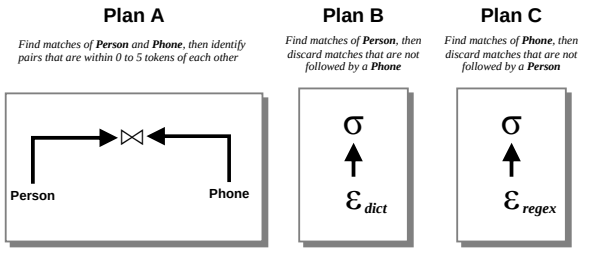


Figure 4: Execution strategies for *PersonPhoneAll* in Fig. 2

4 The Optimizer

Grammar-based IE engines such as (Boguraev, 2003; Cunningham et al., 2000) place rigid restrictions on the order in which rules can be executed. Such systems that implement the CPSL standard or extensions of it must use a finite state transducer to evaluate each level of the cascade with one or more left to right passes over the entire input token stream. In contrast, SystemT uses a declarative approach based on rules that specify *what* patterns to extract, as opposed to *how* to extract them. In a declarative IE system such as SystemT the specification of an annotator is completely separate from its implementation. In particular, the system does not place explicit constraints on the order of rule evaluation, nor does it require that intermediate results of an annotator collapse to a fixed-size sequence.

As shown in Fig. 1, the SystemT engine does not execute AQL directly; instead, the SystemT Optimizer compiles AQL into a graph of operators. Given a collection of AQL views, the optimizer generates a large number of different operator graphs, all of which faithfully implement the semantics of the original views. Even though these graphs always produce the same results, the execution strategies that they represent can have very different performance characteristics. The optimizer incorporates a *cost model* which, given an operator graph, estimates the CPU time required to execute the graph over an average document in the corpus. This cost model allows the optimizer to estimate the cost of each potential execution strategy and to choose the one with the fastest predicted running time.

Fig. 4 presents three possible execution strategies for the *PersonPhoneAll* rule in Fig. 2. If the optimizer estimates that the evaluation cost of *Person* is much lower than that of *Phone*, then it can determine that Plan B has the lowest evaluation cost among the three, because Plan B only evaluates *Phone* in the "right" neighborhood for each instance of *Person*. More details of our algorithms for enumerating plans can be found in (F.Reiss et al., 2008).

The optimizer in SystemT chooses the best execution plan from a large number of different algebra graphs available. Depending on the execution plan generated by the optimizer, SystemT may evaluate views out of order, or it may skip evaluating some views entirely. It may share work among views or combine multiple equivalent views together. Even within the context of a single view, the system can choose among several different execution strategies without affecting the semantics of the annotator. This decoupling is possible because of the declarative approach in SystemT, where the AQL rules specify only what patterns to extract and not how to extract them. Notice that many of these strategies cannot be implemented using a transducer. In fact, we have formally proven that within this large search space, there generally exists an execution strategy that implements the rule semantics far more efficiently than the fastest transducer could (Chiticariu et al., 2010b). This approach also allows for greater rule expressivity, because the rule language is not constrained by the need to compile to a finite state transducer, as in traditional CPSL-based systems.

5 The Runtime

The SystemT Runtime is a compact, small memory footprint, high-performance Java-based runtime engine designed to be embedded in a larger system. The runtime engine works in two steps. First, it instantiates the physical operators in the compiled operator graph generated by the optimizer. Second, once the first step has been completed, the runtime feeds documents through the operator graph one at a time, producing annotations.

SystemT exposes a generic Java API for the integration of its runtime environment with other applications. Furthermore, SystemT provides two specific instantiations of the Java API: a *UIMA API* and a *Jaql function* that allow the SystemT runtime to be seamlessly embedded in applications using the UIMA analytics framework (UIMA, 2010), or deployed in a Hadoop-based environment. The latter

allows SystemT to be embedded as a Map job in a map-reduce framework, thus enabling the system to scale up and process large volumes of documents in parallel.

5.1 Memory Consumption

Managing memory consumption is very important in information extraction systems. Extracting structured information from unstructured text requires generating and traversing large in-memory data structures, and the size of these structures determines how large a document the system can process with a given amount of memory.

Conventional rule-based IE systems cannot garbage-collect their main-memory data structures because the custom code embedded inside rules can change these structures in arbitrary ways. As a result, the memory footprint of the rule engine grows continuously throughout processing a given document.

In SystemT, the AQL view definitions clearly specify the data dependencies between rules. When generating an execution plan for an AQL annotator, the optimizer generates information about when it is safe to discard a given set of intermediate results. The SystemT Runtime uses this information to implement garbage collection based on reference-counting. This garbage collection significantly reduces the system's peak memory consumption, allowing SystemT to handle much larger documents than conventional IE systems.

6 The Development Environment

The SystemT Development Environment assists a developer in the iterative process of developing, testing, debugging and refining AQL rules. Besides standard editor features present in any well-respected IDE for programming languages such as syntax highlighting, the Development Environment also provides facilities for visualizing the results of executing the rules over a sample document collection as well as explaining in detail the *provenance* of any output annotation as the sequence of rules that have been applied in generating that output.

7 Evaluation

As discussed in Section 1, our goal in building SystemT was to address the scalability and usability challenges posed by enterprise applications. As such, our evaluation focuses on these two dimensions.

Application Type	Type of Platform
brand management	server-side
business insights	server-side
client-side mashups	client-side
compliance	server-side
search (email, web, patent)	server-side
security	server-side
server-side mashups	server-side

Table 1: Types of applications using SystemT

7.1 Scalability

Table 1 presents a diverse set of enterprise applications currently using SystemT. SystemT has been deployed in both client-side applications with strict memory constraints, as well as on applications on the cloud, where it can process petabytes of data in parallel. The focus on scalability in the design of SystemT is essential for its flexible execution model. First of all, efficient execution plans are generated automatically by the SystemT Optimizer based on sample document collections. This ensures that the same annotator can be executed efficiently for different types of document collections. In fact, our previous experimental study shows that the execution plan generated by the SystemT optimizer can be 20 times or more faster than a manually constructed plan (F.Reiss et al., 2008). Furthermore, the Runtime Environment of SystemT results in compact memory footprint and allows SystemT to be embedded in applications with strict memory requirements as small as 10MB.

In our recent study over several document collections of different sizes, we found that for the same set of extraction tasks, the SystemT throughput is at least an order of magnitude higher than that of a state-of-the-art grammar-based IE system, with much lower memory footprint (Chiticariu et al., 2010b). The high throughput and low memory footprint of SystemT allows it to satisfy the scalability requirement of enterprise applications.

7.2 Usability

Table 2 lists different types of annotators built using SystemT for a wide range of domains. Most,

Domain	Sample Annotators Built
blog	`Sentiment`, `InformalReview`
email	`ConferenceCall`, `Signature`, `Agenda`, `DrivingDirection`, `PersonPhone`, `PersonAddress`, `PersonEmailAddress`
financial	`Merger`, `Acquisition`, `JointVenture`, `EarningsAnnouncement`, `AnalystEarningsEstimate`, `DirectorsOfficers`, `CorporateActions`
generic	`Person`, `Location`, `Organization`, `PhoneNumber`, `EmailAddress`, `URL`, `Time`, `Date`
healthcare	`Disease`, `Drug`, `ChemicalCompound`
web	`Homepage`, `Geography`, `Title`, `Heading`

Table 2: List of Sample Annotators Built Using SystemT for Different Domains

if not all, of these annotators are already deployed in commercial products. The emphasis on usability in the design of SystemT has been critical for its successful deployment in various domains. First of all, the declarative approach taken by SystemT allows developers to build complex annotators without worrying about performance. Secondly, the expressiveness of the AQL language has greatly eased the burden of annotator developers when building complex annotators, as complex semantics such as duplicate elimination and aggregation can be expressed in a concise fashion (Chiticariu et al., 2010b). Finally, the Development Environment further facilitates annotator development, where the clean semantics of AQL can be exploited to automatically construct explanations of incorrect results to help a developer in identifying specific parts of the annotator responsible for a given mistake. SystemT has been successfully used by enterprise application developers in building high quality complex annotators, without requiring extensive training or background in natural language processing.

8 Demonstration

This demonstration will present the core functionalities of SystemT. In particular, we shall demonstrate the iterative process of building and debugging an annotator in the Development Environment. We will then showcase the execution plan automatically generated by the Optimizer based on a sample document collection, and present the output of the Runtime Environment using the execution plan. In our demonstration we will first make use of a simple annotator, as the one shown in Fig. 2, to illustrate the main constructs of AQL. We will then showcase the generic state-of-the-art SystemT Named Entities Annotator Library (Chiticariu et al., 2010c) to illustrate the quality of annotators that can be built in our system.

References

D. E. Appelt and B. Onyshkevych. 1998. The common pattern specification language. In *TIPSTER workshop*.

B. Boguraev. 2003. Annotation-based finite state processing in a large-scale nlp arhitecture. In *RANLP*.

P. Bohannon et al. 2008. Purple SOX Extraction Management System. *SIGMOD Record*, 37(4):21–27.

L. Chiticariu, Y. Li, S. Raghavan, and F. Reiss. 2010a. Enterprise information extraction: Recent developments and open challenges. In *SIGMOD*.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R. Reiss, and Shivakumar Vaithyanathan. 2010b. Systemt: an algebraic approach to declarative information extraction. ACL.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. 2010c. Domain adaptation of rule-based annotators for named-entity recognition tasks. EMNLP.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield.

A. Doan et al. 2008. Information extraction challenges in managing unstructured data. *SIGMOD Record*, 37(4):14–20.

A. Doan, R. Ramakrishnan, and S. Vaithyanathan. 2006. Managing Information Extraction: State of the Art and Research Directions. In *SIGMOD*.

F.Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. 2008. An algebraic approach to rule-based information extraction. In *ICDE*.

A. Jain, P. Ipeirotis, and L. Gravano. 2009. Building query optimizers for information extraction: the sqout project. *SIGMOD Rec.*, 37:28–34.

R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. 2008. SystemT: a system for declarative information extraction. *SIGMOD Record*, 37(4):7–13.

D. Z. Wang, E. Michelakis, M. J. Franklin, M. Garofalakis, and J. M. Hellerstein. 2010. Probabilistic declarative information extraction. In *ICDE*.

SciSumm: A Multi-Document Summarization System for Scientific Articles

Nitin Agarwal
Language Technologies Institute
Carnegie Mellon University
nitina@cs.cmu.edu

Kiran Gvr
Language Technologies Resource Center
IIIT-Hyderabad, India
kiran_gvr@students.iiit.ac.in

Ravi Shankar Reddy
Language Technologies Resource Center
IIIT-Hyderabad, India
krs_reddy@students.iiit.ac.in

Carolyn Penstein Rosé
Language Technologies Institute
Carnegie Mellon University
cprose@cs.cmu.edu

Abstract

In this demo, we present SciSumm, an interactive multi-document summarization system for scientific articles. The document collection to be summarized is a list of papers cited together within the same source article, otherwise known as a co-citation. At the heart of the approach is a topic based clustering of fragments extracted from each article based on queries generated from the context surrounding the co-cited list of papers. This analysis enables the generation of an overview of common themes from the co-cited papers that relate to the context in which the co-citation was found. SciSumm is currently built over the 2008 ACL Anthology, however the generalizable nature of the summarization techniques and the extensible architecture makes it possible to use the system with other corpora where a citation network is available. Evaluation results on the same corpus demonstrate that our system performs better than an existing widely used multi-document summarization system (MEAD).

1 Introduction

We present an interactive multi-document summarization system called SciSumm that summarizes document collections that are composed of lists of papers cited together within the same source article, otherwise known as a co-citation. The interactive nature of the summarization approach makes this demo session ideal for its presentation.

When users interact with SciSumm, they request summaries in context as they read, and that context determines the focus of the summary generated for a set of related scientific articles. This behaviour is different from some other non-interactive summarization systems that might appear as a black box and might not tailor the result to the specific information needs of the users in context. SciSumm captures a user's contextual needs when a user clicks on a co-citation. Using the context of the co-citation in the source article, we generate a query that allows us to create a summary in a query-oriented fashion. The extracted portions of the co-cited articles are then assembled into clusters that represent the main themes of the articles that relate to the context in which they were cited. Our evaluation demonstrates that SciSumm achieves higher quality summaries than a state-of-the-art multidocument summarization system (Radev, 2004).

The rest of the paper is organized as follows. We first describe the design goals for SciSumm in 2 to motivate the need for the system and its usefulness. The end-to-end summarization pipeline has been described in Section 3. Section 4 presents an evaluation of summaries generated from the system. We present an overview of relevant literature in Section 5. We end the paper with conclusions and some interesting further research directions in Section 6.

2 Design Goals

Consider that as a researcher reads a scientific article, she/he encounters numerous citations, most of them citing the foundational and seminal work that is important in that scientific domain. The text surrounding these citations is a valuable resource as it allows the author to make a statement about her

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 115–120,
Portland, Oregon, USA, 21 June 2011.

viewpoint towards the cited articles. However, to researchers who are new to the field, or sometimes just as a side-effect of not being completely up-to-date with related work in a domain, these citations may pose a challenge to readers. A system that could generate a small summary of the collection of cited articles that is constructed specifically to relate to the claims made by the author citing them would be incredibly useful. It would also help the researcher determine if the cited work is relevant for her own research.

As an example of such a co-citation consider the following citation sentence:

Various machine learning approaches have been proposed for chunking (Ramshaw and Marcus, 1995; Tjong Kim Sang, 2000a; Tjong Kim Sang et al. , 2000; Tjong Kim Sang, 2000b; Sassano and Utsuro, 2000; van Halteren, 2000).

Now imagine the reader trying to determine about widely used *machine learning* approaches for *noun phrase chunking*. He would probably be required to go through these cited papers to understand what is similar and different in the variety of chunking approaches. Instead of going through these individual papers, it would be quicker if the user could get the summary of the topics in all those papers that talk about the usage of *machine learning* methods in *chunking*. SciSumm aims to automatically discover these points of comparison between the co-cited papers by taking into consideration the contextual needs of a user. When the user clicks on a co-citation in context, the system uses the text surrounding that co-citation as evidence of the information need.

3 System Overview

A high level overview of our system's architecture is presented in Figure 1. The system provides a web based interface for viewing and summarizing research articles in the ACL Anthology corpus, 2008. The summarization proceeds in three main stages as follows:

- A user may retrieve a collection of articles of interest by entering a query. SciSumm responds by returning a list of relevant articles, including the title and a snippet based summary. For this SciSumm uses standard retrieval from a Lucene index.

- A user can use the title, snippet summary and author information to find an article of interest. The actual article is rendered in HTML after the user clicks on one of the search results. The co-citations in the article are highlighted in bold and italics to mark them as points of interest for the user.

- If a user clicks on one, SciSumm responds by generating a query from the local context of the co-citation. That query is then used to select relevant portions of the co-cited articles, which are then used to generate the summary.

An example of a summary for a particular topic is displayed in Figure 2. This figure shows one of the clusters generated for the citation sentence "Various machine learning approaches have been proposed for chunking (Ramshaw and Marcus, 1995; Tjong Kim Sang, 2000a; Tjong Kim Sang et al. , 2000; Tjong Kim Sang, 2000b; Sassano and Utsuro, 2000; van Halteren, 2000)". The cluster has a label *Chunk, Tag, Word* and contains fragments from two of the papers discussing this topic. A ranked list of such clusters is generated, which allows for swift navigation between topics of interest for a user (Figure 3). This summary is tremendously useful as it informs the user of the different perspectives of co-cited authors towards a shared problem (in this case "Chunking"). More specifically, it informs the user as to how different or similar approaches are that were used for this research problem (which is "Chunking").

3.1 System Description

SciSumm has four primary modules that are central to the functionality of the system, as displayed in Figure 1. First, the Text Tiling module takes care of obtaining tiles of text relevant to the citation context. Next, the clustering module is used to generate labelled clusters using the text tiles extracted from the co-cited papers. The clusters are ordered according to relevance with respect to the generated query. This is accomplished by the Ranking Module.

In the following sections, we discuss each of the main modules in detail.

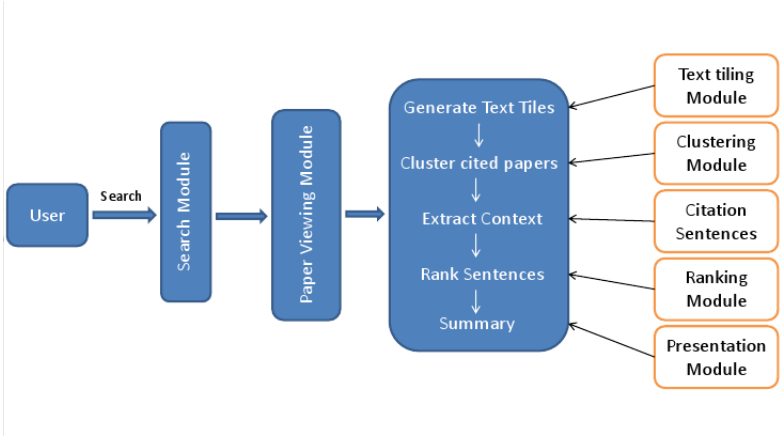


Figure 1: SciSumm summarization pipeline

3.2 Texttiling

The Text Tiling module uses the TextTiling algorithm (Hearst, 1997) for segmenting the text of each article. We have used text tiles as the basic unit for our summary since individual sentences are too short to stand on their own. This happens as a side-effect of the length of scientific articles. Sentences picked from different parts of several articles assembled together would make an incoherent summary. Once computed, text tiles are used to expand on the content viewed within the context associated with a co-citation. The intuition is that an embedded co-citation in a text tile is connected with the topic distribution of its context. Thus, we can use a computation of similarity between tiles and the context of the co-citation to rank clusters generated using Frequent Term based text clustering.

3.3 Frequent Term Based Clustering

The clustering module employs Frequent Term Based Clustering (Beil et al., 2002). For each co-citation, we use this clustering technique to cluster all the of the extracted text tiles generated by segmenting each of the co-cited papers. We settled on this clustering approach for the following reasons:

- Text tile contents coming from different papers constitute a sparse vector space, and thus the centroid based approaches would not work very well for integrating content across papers.
- Frequent Term based clustering is extremely fast in execution time as well as and relatively efficient in terms of space requirements.
- A frequent term set is generated for each cluster, which gives a comprehensible description that can be used to label the cluster.

Frequent Term Based text clustering uses a group of frequently co-occurring terms called a frequent term set. We use a measure of entropy to rank these frequent term sets. Frequent term sets provide a clean clustering that is determined by specifying the number of overlapping documents containing more than one frequent term set. The algorithm uses the first k term sets if all the documents in the document collection are clustered. To discover all the possible candidates for clustering, i.e., term sets, we used the *Apriori* algorithm (Agrawal et al., 1994), which identifies the sets of terms that are both relatively frequent and highly correlated with one another.

3.4 Cluster Ranking

The ranking module uses cosine similarity between the query and the centroid of each cluster to rank all the clusters generated by the clustering module. The context of a co-citation is restricted to the text of the segment in which the co-citation is found. In this way we attempt to leverage the expert knowledge of the author as it is encoded in the local context of the co-citation.

4 Evaluation

We have taken great care in the design of the evaluation for the SciSumm summarization system. In a

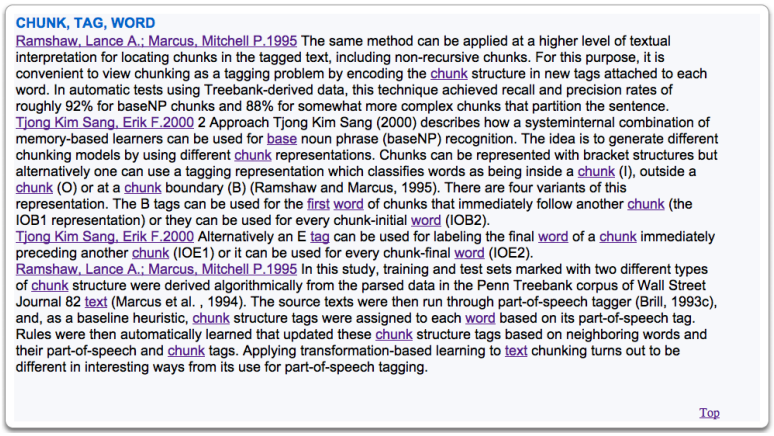


Figure 2: Example of a summary generated by our system. We can see that the clusters are cross cutting across different papers, thus giving the user a multi-document summary.

typical evaluation of a multi-document summarization system, gold standard summaries are created by hand and then compared against fixed length generated summaries. It was necessary to prepare our own evaluation corpus, consisting of gold standard summaries created for a randomly selected set of co-citations because such an evaluation corpus does not exist for this task.

4.1 Experimental Setup

An important target user population for multi-document summarization of scientific articles is graduate students. Hence to get a measure of how well the summarization system is performing, we asked 2 graduate students who have been working in the computational linguistics community to create gold standard summaries of a fixed length (8 sentences $\sim$ 200 words) for 10 randomly selected co-citations. We obtained two different gold standard summaries for each co-citation (i.e., 20 gold standard summaries total). Our evaluation is designed to measure the quality of the content selection. In future work, we will evaluate the usability of the SciSumm system using a task based evaluation.

In the absence of any other multi-document summarization system in the domain of scientific article summarization, we used a widely used and freely available multi-document summarization system called MEAD (Radev, 2004) as our baseline. MEAD uses centroid based summarization to create informative clusters of topics. We use the default configuration of MEAD in which MEAD uses length, position and centroid for ranking each sentence. We did not use query focussed summarization with MEAD. We evaluate its performance with the same gold standard summaries we use to evaluate SciSumm. For generating a summary from our system we used sentences from the tiles that are clustered in the top ranked cluster. Once all of the extracts included in that entire cluster are exhausted, we move on to the next highly ranked cluster. In this way we prepare a summary comprising of 8 highly relevant sentences.

4.2 Results

For measuring performance of the two summarization systems (SciSumm and MEAD), we compute the ROUGE metric based on the 2 * 10 gold standard summaries that were manually created. ROUGE has been traditionally used to compute the performance based on the N-gram overlap (ROUGE-N) between the summaries generated by the system and the target gold standard summaries. For our evaluation we used two different versions of the ROUGE metric, namely ROUGE-1 and ROUGE-2, which correspond to measures of the unigram and bigram overlap respectively. We computed four metrics in order to get a complete picture of how SciSumm performs in relation to the baseline, namely ROUGE-1 F-measure, ROUGE-1 Recall, ROUGE-2 F-measure, and ROUGE-2 Recall.

From the results presented in Figure 4 and 5, we can see that our system performs well on average in comparison to the baseline. Especially important is

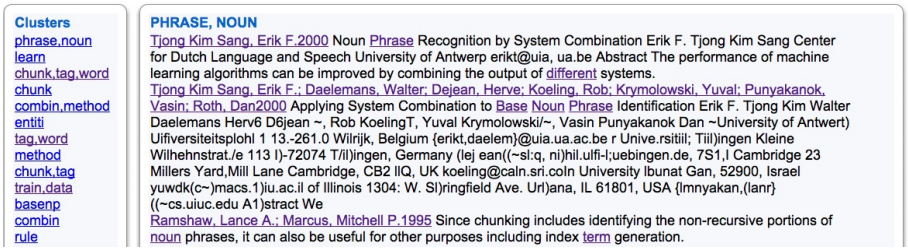


Figure 3: Clusters generated in response to a user click on the co-citation. The list of clusters in the left pane gives a bird-eye view of the topics which are present in the co-cited papers

Table 1: Average ROUGE results. * represents improvement significant at $p < .05$, † at $p < .01$.

Metric	MEAD	SciSumm
ROUGE-1 F-measure	0.3680	0.5123 †
ROUGE-1 Recall	0.4168	0.5018
ROUGE-1 Precision	0.3424	0.5349 †
ROUGE-2 F-measure	0.1598	0.3303 *
ROUGE-2 Recall	0.1786	0.3227 *
ROUGE-2 Precision	0.1481	0.3450 †

the performance of the system on recall measures, which shows the most dramatic advantage over the baseline. To measure the statistical significance of this result, we carried out a Student T-Test, the results of which are presented in the results section in Table 1. It is apparent from the p-values generated by T-Test that our system performs significantly better than MEAD on three of the metrics on which both the systems were evaluated using $(p < 0.05)$ as the criterion for statistical significance. This supports the view that summaries perceived as higher in value are generated using a query focused technique, where the query is generated automatically from the context of the co-citation.

5 Previous Work

Surprisingly, not many approaches to the problem of summarization of scientific articles have been proposed in the past. Qazvinian et al. (2008) present a summarization approach that can be seen as the converse of what we are working to achieve. Rather than summarizing multiple papers cited in the same source article, they summarize different viewpoints expressed towards the same paper from different papers that cite it. Nanba et al. (1999) argue in their work that a co-citation frequently implies a consistent viewpoint towards the cited articles. Another approach that uses bibliographic coupling has been used for gathering different viewpoints from which to summarize a document (Kaplan et al., 2008). In our work we make use of this insight by generating a query to focus our multi-document summary from the text closest to the citation.

6 Conclusion And Future Work

In this demo, we present SciSumm, which is an interactive multi-document summarization system for scientific articles. Our evaluation shows that the SciSumm approach to content selection outperforms another widely used multi-document summarization system for this summarization task.

Our long term goal is to expand the capabilities of SciSumm to generate literature surveys of larger document collections from less focused queries. This more challenging task would require more control over filtering and ranking in order to avoid generating summaries that lack focus. To this end, a future improvement that we plan to use is a variant on MMR (Maximum Marginal Relevance) (Carbonell et al., 1998), which can be used to optimize the diversity of selected text tiles as well as the relevance based ordering of clusters, i.e., so that more diverse sets of extracts from the co-cited articles will be placed at the ready fingertips of users.

Another important direction is to refine the interaction design through task-based user studies. As we collect more feedback from students and researchers through this process, we will used the insights gained to achieve a more robust and effective implementation.

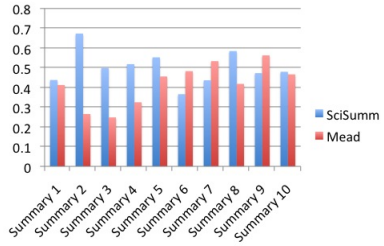


Figure 4: ROUGE-1 Recall

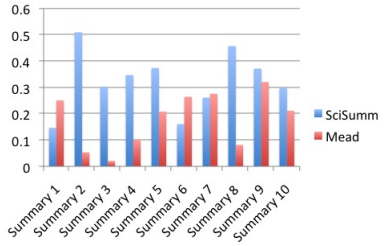


Figure 5: ROUGE-2 Recall

7 Acknowledgements

This research was supported in part by NSF grant EEC-064848 and ONR grant N00014-10-1-0277.

References

Agrawal R. and Srikant R. 1994. Fast Algorithm for Mining Association Rules In *Proceedings of the 20th VLDB Conference* Santiago, Chile, 1994

Baxendale, P. 1958. Machine-made index for technical literature - an experiment. *IBM Journal of Research and Development*

Beil F., Ester M. and Xu X 2002. Frequent-Term based Text Clustering In *Proceedings of SIGKDD '02* Edmonton, Alberta, Canada

Carbonell J. and Goldstein J. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries In *Research and Development in Information Retrieval*, pages 335–336

Councill I. G. , Giles C. L. and Kan M. 2008. ParsCit: An open-source CRF reference string parsing package *INTERNATIONAL LANGUAGE RESOURCES AND EVALUATION European Language Resources Association*

Edmundson, H.P. 1969. New methods in automatic extracting. *Journal of ACM.*

Hearst M.A. 1997 TextTiling: Segmenting text into multi-paragraph subtopic passages *In proceedings of LREC 2004, Lisbon, Portugal, May 2004*

Joseph M. T. and Radev D. R. 2007. Citation analysis, centrality, and the ACL Anthology

Kupiec J. , Pedersen J. , Chen F. 1995. A training document summarizer. In *Proceedings SIGIR '95*, pages 68-73, New York, NY, USA. 28(1):114–133.

Luhn, H. P. 1958. *IBM Journal of Research Development.*

Mani I. , Bloedorn E. 1997. Multi-Document Summarization by graph search and matching In *AAAI/IAAI*, pages 622-628. [15, 16].

Nanba H. , Okumura M. 1999. Towards Multi-paper Summarization Using Reference Information In *Proceedings of IJCAI-99*, pages 926–931 .

Paice CD. 1990. Constructing Literature Abstracts by Computer: Techniques and Prospects *Information Processing and Management* Vol. 26, No.1, pp, 171-186, 1990

Qazvinian V. , Radev D.R 2008. Scientific Paper summarization using Citation Summary Networks In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 689–696 Manchester, August 2008

Radev D. R . , Jing H. and Budzikowska M. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility based evaluation, and user studies In *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 21-30, Morristown, NJ, USA. [12, 16, 17].

Radev, Dragomir. 2004. *MEAD - a platform for multidocument multilingual text summarization.* In proceedings of LREC 2004, Lisbon, Portugal, May 2004.

Teufel S. , Moens M. 2002. Summarizing Scientific Articles - Experiments with Relevance and Rhetorical Status In *Journal of Computational Linguistics*, MIT Press.

Hal Daume III , Marcu D. 2006. Bayesian query-focussed summarization. In *Proceedings of the Conference of the Association for Computational Linguistics*, ACL.

Eisenstein J , Barzilay R. 2008. Bayesian unsupervised topic segmentation In *EMNLP-SIGDAT.*

Barzilay R , Lee L. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization In *Proceedings of 3rd Asian Semantic Web Conference (ASWC 2008), pp.182-188,.*

Kaplan D , Tokunaga T. 2008. Sighting citation sights: A collective-intelligence approach for automatic summarization of research papers using C-sites In *HLT-NAACL.*

Clairlib: A Toolkit for Natural Language Processing, Information Retrieval, and Network Analysis

Amjad Abu-Jbara
EECS Department
University of Michigan
Ann Arbor, MI, USA
amjbara@umich.edu

Dragomir Radev
EECS Department and
School of Information
University of Michigan
Ann Arbor, MI, USA
radev@umich.edu

Abstract

In this paper we present Clairlib, an open-source toolkit for Natural Language Processing, Information Retrieval, and Network Analysis. Clairlib provides an integrated framework intended to simplify a number of generic tasks within and across those three areas. It has a command-line interface, a graphical interface, and a documented API. Clairlib is compatible with all the common platforms and operating systems. In addition to its own functionality, it provides interfaces to external software and corpora. Clairlib comes with a comprehensive documentation and a rich set of tutorials and visual demos.

1 Introduction

The development of software packages and code libraries that implement algorithms and perform tasks in scientific areas is of great advantage for both researchers and educators. The availability of these tools saves the researchers a lot of the time and the effort needed to implement the new approaches they propose and conduct experiments to verify their hypotheses. Educators also find these tools useful in class demonstrations and for setting up practical programming assignments and projects for their students.

A large number of systems have been developed over the years to solve problems and perform tasks in Natural Language Processing, Information Retrieval, or Network Analysis. Many of these systems perform specific tasks such as parsing, Graph Partitioning, co-reference resolution, web crawling etc. Some other systems are frameworks for performing generic tasks in one area of focus such as NLTK (Bird and Loper, 2004) and GATE (Cunningham et al., 2002) for Natural Language Processing; Pajek (Batagelj and Mrvar, 2003) and GUESS (Adar, 2006) for Network Analysis and Visualization; and Lemur[1] for Language Modeling and Information Retrieval.

This paper presents Clairlib, an open-source toolkit that contains a suit of modules for generic tasks in Natural Language Processing (NLP), Information Retrieval (IR), and Network Analysis (NA). While many systems have been developed to address tasks or subtasks in one of these areas as we have just mentioned, Clairlib provides one integrated environment that addresses tasks in the three areas. This makes it useful for a wide range of applications within and across the three domains.

Clairlib is designed to meet the needs of researchers and educators with varying purposes and backgrounds. For this purpose, Clairlib provides three different interfaces to its functionality: a graphical interface, a command-line interface, and an application programming interface (API).

Clairlib is developed and maintained by the Computational Linguistics and Information Retrieval (CLAIR) group at the University of Michigan. The first version of Clairlib was released in the year 2007. It has been heavily developed since then until it witnessed a qualitative leap by adding the Graphical Interface and many new features to the latest version that we are presenting here.

Clairlib core modules are written in Perl. The GUI was written in Java. The Perl back-end and the Java front-end are efficiently tied together through a communication module. Clairlib is compatible with

[1] http://www.lemurproject.org/

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 121–126,
Portland, Oregon, USA, 21 June 2011.

all the common platforms and operating systems. The only requirements are a Perl interpreter and Java Runtime Environment (JRE).

Clairlib has been used in several research projects to implement systems and conduct experiments. It also has been used in several academic courses.

The rest of this paper is organized as follows. In Section 2, we describe the structure of Clairlib. In Section 3, we present its functionality. Section 4 presents some usage examples. We conclude in Section 5.

2 System Overview

Clairlib consists of three main components: the core library, the command-line interface, and the graphical user interface. The three components were designed and connected together in a manner that aims to achieve simplicity, integration, and ease of use. In the following subsections, we briefly describe each of the three components.

2.1 Modules

The core of Clairlib is a collection of more than 100 modules organized in a shallow hierarchy, each of which performs a specific task or implements a certain algorithm. A set of core modules define the data structures and perform the basic processing tasks. For example, Clair::Document defines a data structure for holding textual data in various formats, and performs the basic text processing tasks such as tokenization, stemming, tag stripping, etc.

Another set of modules perform more specific tasks in the three areas of focus (NLP, IR, and NA). For example, Clair::Bio::GIN::Interaction is devoted to protein-protein interaction extraction from biomedical text.

A third set contains modules that interface Clairlib to external tools. For example, Clair::Utils::Parse provides an interface to Charniak parser (Charniak, 2000), Stanford parser (Klein and Manning, 2003), and Chunklink[2].

Each module has a well-defined API. The API is oriented to developers to help them write applications and build systems on top of Clairlib modules; and to researchers to help them write applications and setup custom experiments for their research.

[2]http://ilk.uvt.nl/team/sabine/chunklink/README.html

2.2 Command-line Interface

The command-line interface provides an easy access to many of the tasks that Clairlib modules implement. It provides more than 50 different commands. Each command is documented and demonstrated in one or more tutorials. The function of each command can be customized by passing arguments with the command. For example, the command

partition.pl -graph graph.net -method GirvanNewman -n 4

uses the GrivanNewman algorithm to divide a given graph into 4 partitions.

2.3 Graphical User Interface

The graphical user interface (GUI) is an important feature that has been recently added to Clairlib and constituted a quantum leap in its development. The main purpose of the GUI is to make the rich set of Clairlib functionalities easier to access by a larger number of users from various levels and backgrounds especially students and users with limited or no programming experience.

It is also intended to help students do their assignments, projects, and research experiments in an interactive environment. We believe that visual tools facilitate understanding and make learning a more enjoyable experience for many students. Focusing on this purpose, the GUI is tuned for simplicity and ease of use more than high computational efficiency. Therefore, while it is suitable for small and medium scale projects, it is not guaranteed to work efficiently for large projects that involve large datasets and require heavy processing. The command-line interface is a better choice for large projects.

The GUI consists of three components: the Network Editor/Visualizer/Analyzer, the Text Processor, and the Corpus Processor. The Network component allows the user to 1) build a new network using a set of drawing and editing tools, 2) open existing networks stored in files in several different formats, 3) visualize a network and interact with it, 4) compute different statistics for a network such as diameter, clustering coefficient, degree distribution, etc., and 5) perform several operations on a network such as random walk, label propagation, partitioning, etc. This component uses the open source library, JUNG[3] to visualize networks. Figure 1 shows

[3]http://jung.sourceforge.net/

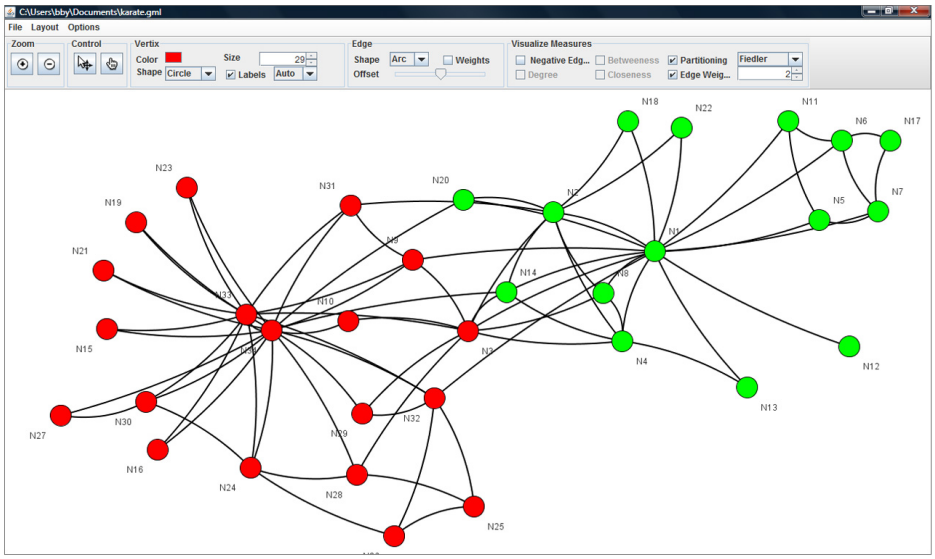


Figure 1: A screenshot for the network visualization component of Clairlib

a screenshot for the Network Visualizer.

The Text Processing component allows users to process textual data published on the internet or imported from a file stored on the disk. It can process data in plain, html, or PDF format. Most of the text processing capabilities implemented in Clairlib core library are available through this component. Figure 2 shows a screenshot of the text processing component.

The Corpus Processing component allows users to build a corpus of textual data out of a collection of files in plain, HTML, or PDF format; or by crawling a website. Several tasks could be performed on a corpus such as indexing, querying, summarization, information extraction, hyperlink network construction, etc.

Although these components can be run independently, they are very integrated and designed to easily interact with each other. For example, a user can crawl a website using the Corpus component, then switch to the Text Processing component to extract the text from the web documents and stem all the words, then switch back to the Corpus component to build a document similarity graph. The graph can then be taken to the Network component to be visualized and analyzed.

2.4 Documentation

Clairlib comes with an extensive documentation. The documentation contains the installation information for different platforms, a description of all Clairlib components and modules, and a lot of usage examples. In addition to this documentation, Clairlib provides three other resources:

API Reference

The API Reference provides a complete description of each module in the library. It describes each subroutine, the task it performs, the arguments it takes, the value it returns, etc. This reference is useful for developers who want to use Clairlib modules in their own applications and systems. The API Reference is published on the internet.

Tutorials

Tutorials teach users how to use Clairlib by examples. Each tutorial addresses a specific task and provides a set of instructions to complete the task using Clairlib command-line tools or its API.

Visual Demos

Visual demos target the users of the graphical interface. The demos visually show how to start the GUI and how to use its components to perform several tasks.

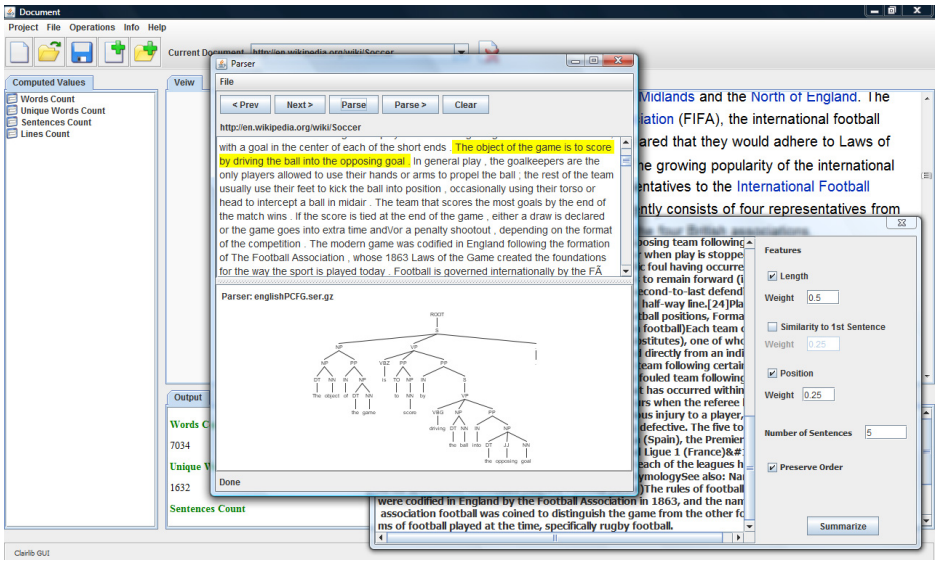


Figure 2: A screenshot for the text processing component of Clairlib

3 Functionality

Clairlib provides modules and tools for a broad spectrum of tasks. Most of the functionalities are native to Clairlib. Some functionalities, however, are imported from other open-source packages or external software. This section lists the main functionalities categorized by their areas.

3.1 Natural Language Processing

NLP functionalities include Tokenization, Sentence Segmentation, Stemming, HTML Tags Stripping, Syntactic Parsing, Dependency Parsing, Part-of-Speech Tagging, Document Classification, LexRank, Summarization, Synthetic Corpus Generation, N-grams Extraction, XML Parsing, XML Tree Building, Text Similarity, Political Text Analysis, and Protein Name Tagging.

3.2 Information Retrieval

IR functionalities include Web Crawling, Indexing, TF-IDF, PageRank, Phrase Based Retrieval, Fuzzy OR Queries, Latent Semantic Indexing, Web Search, Automatic Link Extraction, and Protein-Protein Interaction Extraction.

3.3 Network Analysis

Network Analysis functionalities include Network Statistics, Random Network Generation, Network Visualization, Network Partitioning, Community Finding, Random Walks, Flow Networks, Signed Networks, and Semi-supervised Graph-based Classification. Network Statistics include Centralities, Clustering Coefficient, Shortest Paths, Diameter, Triangles, Triplets, etc.

Some of these functionalities are implemented using several approaches. For example, Clairlib have implementations for 5 graph partitioning algorithms. This makes Clairlib a useful tool for conducting experiments for comparative studies.

4 Uses of Clairlib

The diverse set of domains that Clairlib covers and the different types of interfaces it provides make it suitable for use in many contexts. In this section, we highlight some of its uses.

Education

Clairlib contains visual tools that instructors can use to do class demonstrations to help their students understand the basic concepts and the algorithms they face during their study. For example, the random walk simulator can be used to teach the students how random walk works by showing a sample network and then walk randomly step-by-step through it and show the students how the probabilities change after each step.

It can also be used to create assignments of varying levels of difficulty and different scopes. Instruc-

tors may ask their students to do experiments with a dataset using Clairlib, write applications that use the API, extend an existing module, or contribute new modules to Clairlib. One example could be to ask the students to a build a simple information retrieval system that indexes a collection of documents and executes search queries on it.

Clairlib has been used to create assignments and projects in NLP and IR classes at the University of Michigan and Columbia University. The experience was positive for both the instructors and the students. The instructors were able to design assignments that cover several aspects of the course and can be done in a reasonable amount of time. The students used the API to accomplish their assignments and projects. This helped them focus on the important concepts rather than diving into fine programming details.

Research

Clairlib contains implementations for many algorithms and approaches that solve common problems. It also comes with a number of corpora and annotated datasets. This makes it a good resource for researchers to build systems and conduct experiments.

Clairlib was successfully used in several research projects. Examples include Political Text Analysis (Hassan et al., 2008), Scientific Paper Summarization (Qazvinian and Radev, 2009), Blog Networks Analysis (Hassan et al., 2009), Protein Interaction Extraction (Ozgur and Radev, 2009), and Citation-Based Summarization (Abu-Jbara and Radev, 2011).

4.1 Examples

In this subsection, we present some examples where Clairlib has been used.

Example: Protein-Protein Interaction Extraction

This is an example of a project that builds an information extraction system and uses Clairlib as its main processing component (Ozgur and Radev, 2009). This system is now part of a larger bioinformatics project, NCIBI.

The system uses Clairlib to process a biomedical article: 1) splits it into sentences using the segmentation module, 2) parses each sentence using the interface to the Stanford Dependency Parser, 3) tags the protein names, 4) extracts protein-protein interactions using a specific Clairlib module devoted to this task, and then 5) it builds a protein interaction network in which nodes are proteins and edges represent interaction relations. Figure 3 shows an example protein interaction network extracted from the abstracts of a collection of biomedical articles from PubMed. This network is then analyzed to compute node centralities and the basic network statistics.

Example: Scientific Paper Summarization Using Citation Networks

This is an example of a research work that used Clairlib to implement an approach and conduct experiments to support the research hypothesis. Qazvinian and Radev (2009) used Clairlib to implement their method for citation-based summarization. Given a set of sentences that cite a paper, they use Clairlib to 1) construct a cosine similarity network out of these sentences, 2) find communities of similar sentences using Clairlib community finding module, 3) run Clairlib LexRank module to rank the sentences, 4) extract the sentence with the highest rank from each community, and finally 5) return the set of extracted sentences as a summary paragraph.

Example: Text Classification

This is an example of a teaching assignment that was used in an introductory course on information retrieval at the University of Michigan. Students were given the 20-newsgroups corpus (a large set of news articles labeled by their topic and split into training and testing sets) and were asked to use Clairlib API to: 1) stem the text of the documents, 2) convert each document into a feature vector based on word frequencies, 2) train a multi-class Perceptron or Naive Bayes classifier on the documents in the training set, and finally 3) classify the documents in the testing set using the trained classifier.

5 Conclusions

Clairlib is a broad-coverage toolkit for Natural Language Processing, Information Retrieval, and Network Analysis. It provides a simple, integrated, interactive, and extensible framework for education and research uses. It provides an API, a command-

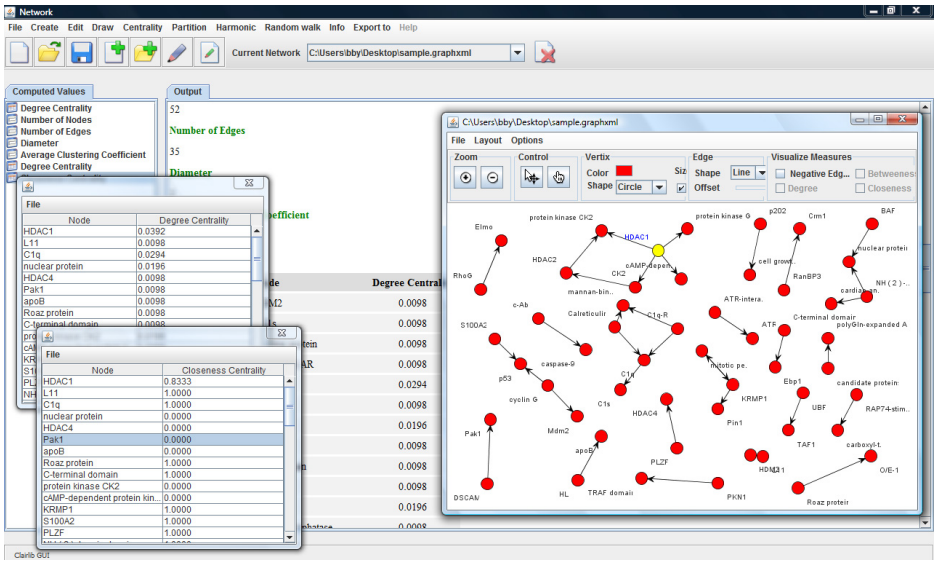


Figure 3: Clairlib used to construct and analyze a protein network extracted from biomedical articles

line interface, and graphical user interface for the convenience of users with varying purposes and backgrounds. Clairlib is well-documented, easy to learn, and simple to use. It has been tested for various types of tasks in various environments.

Clairlib is an open source project and we welcome all the contributions. Readers who are interested in contributing to Clairlib are encouraged to contact the authors.

Acknowledgements

We would like to thank Mark Hodges, Anthony Fader, Mark Joseph, Joshua Gerrish, Mark Schaller, Jonathan dePeri, Bryan Gibson, Chen Huang, Arzucan Ozgur, and Prem Ganeshkumar who contributed to the development of Clairlib.

This work was supported in part by grants R01-LM008106 and U54-DA021519 from the US National Institutes of Health, U54 DA021519, IDM 0329043, DHB 0527513, 0534323, and 0527513 from the National Science Foundation, and W911NF-09-C-0141 from IARPA.

References

R. Gaizauskas, P. J. Rodgers and K. Humphreys 2001. Visual Tools for Natural Language Processing. *Journal of Visual Languages and Computing, Volume 12, Issue 4, Pages 375-412.*

Arzucan Ozgor and Dragomir Radev 2009. Supervised classification for extracting biomedical events. *Proceedings of the BioNLP'09 Workshop Shared Task on Event Extraction at NAACL-HLT, Boulder, Colorado, USA, pages 111-114*

Ahmed Hassan, Dragomir R. Radev, Junghoo Cho, Amruta Joshi. 2009. Content Based Recommendation and Summarization in the Blogosphere. *ICWSM-2009.*

Vahed Qazvinian, Dragomir Radev. 2008. Scientific Paper Summarization Using Citation Summary Networks. *COLING 2008.*

Ahmed Hassan, Anthony Fader, Michael Crespin, Kevin Quinn, Burt Monroe, Michael Colaresi and Dragomir Radev. 2008. Tracking the Dynamic Evolution of Participants Salience in a Discussion. *COLING 2008.*

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. *Proceedings of NAACL-2000.*

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of ACL-2003.*

Amjad Abu-Jbara and Dragomir Radev 2011. Coherent Citation-based Summarization of Scientific Papers *Proceedings of ACL-2011.*

H. Cunningham and D. Maynard and K. Bontcheva and V. Tablan 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications *Proceedings of ACL-2002, Philadelphia.*

Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit *Proceedings of ACL-2004.*

V. Batagelj and A. Mrvar 2003. Pajek - Analysis and Visualization of Large Networks *Springer, Berlin.*

Eytan Adar. 2006. GUESS: A Language and Interface for Graph Exploration *CHI 2006.*

C-Feel-It: A Sentiment Analyzer for Micro-blogs

Aditya Joshi[1] **Balamurali A R**[2] **Pushpak Bhattacharyya**[1] **Rajat Mohanty** [3]
[1]Dept. of Computer Science and Engineering, IIT Bombay, Mumbai
[2] IITB-Monash Research Academy, IIT Bombay, Mumbai
[3] AOL India (R&D), Bangalore
India
{adityaj,balamurali,pb}@cse.iitb.ac.in r.mohanty@teamaol.com

Abstract

Social networking and micro-blogging sites are stores of opinion-bearing content created by human users. We describe *C-Feel-It*, a system which can tap opinion content in posts (called tweets) from the micro-blogging website, Twitter. This web-based system categorizes tweets pertaining to a search string as positive, negative or objective and gives an aggregate sentiment score that represents a sentiment snapshot for a search string. We present a qualitative evaluation of this system based on a human-annotated tweet corpus.

1 Introduction

A major contribution of Web 2.0 is the explosive rise of user-generated content. The content has been a by-product of a class of Internet-based applications that allow users to interact with each other on the web. These applications which are highly accessible and scalable represent a class of media called *social media*. Some of the currently popular social media sites are Facebook (www.facebook.com), Myspace (www.myspace.com), Twitter (www.Twitter.com) etc. User-generated content on the social media represents the views of the users and hence, may be opinion-bearing. Sales and marketing arms of business organizations can leverage on this information to know more about their customer base. In addition, prospective customers of a product/service can get to know what other users have to say about the product/service and make an informed decision.

C-Feel-It is a web-based system which predicts sentiment in micro-blogs on Twitter (called tweets). (Screencast at: http://www.youtube.com/user/cfeelit/) C-Feel-It uses a rule-based system to classify tweets as positive, negative or objective using inputs from four sentiment-based knowledge repositories. A weighted-majority voting principle is used to predict sentiment of a tweet. An overall sentiment score for the search string is assigned based on the results of predictions for the tweets fetched. This score which is represented as a percentage value gives a live snapshot of the sentiment of users about the topic.

The rest of the paper is organized as follows: Section 2 gives background study of Twitter and related work in the context of sentiment analysis for Twitter. The system architecture is explained in section 3. A qualitative evaluation of our system based on annotated data is described in section 4. Section 5 summarizes the paper and points to future work.

2 Background study

Twitter is a micro-blogging website and ranks second among the present social media websites (Prelovac, 2010). A micro-blog *allows users to exchange small elements of content such as short sentences, individual pages, or video links* (Kaplan and Haenlein, 2010). More about Twitter can be found here [1].

In Twitter, a micro-blogging post is called a *tweet* which can be upto 140 characters in length. Since the length is constrained, the language used in tweets is highly unstructured. Misspellings, slangs, contractions and abbreviations are commonly used in tweets. The following example highlights these problems in a typical tweet:

'*Big brother doing sian massey no favours. Let her ref. She's good at it you know#lifesapitch*'

We choose Twitter as the data source because of the sheer quantity of data generated and its fast reachability across masses. Additionally, Twitter allows information to flow freely and instantaneously unlike FaceBook or MySpace. These aspects of

[1]http://support.twitter.com/groups/31-twitter-basics

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 127–132,
Portland, Oregon, USA, 21 June 2011.

Twitter makes it a source for getting a live snapshot of the things happenings on the web.

In the context of sentiment classification of tweets Alec et al. (2009a) describes a distant supervision-based approach for sentiment classification. The training data for this purpose is created following a semi-supervised approach that exploits emoticons in tweets. In their successive work, Alec et al. (2009b) additionally use hashtags in tweets to create training data. Topic-dependent clustering is performed on this data and classifiers corresponding to each are modeled. This approach is found to perform better than a single classifier alone.

We believe that the models trained on data created using semi-supervised approaches cannot classify all variants of tweets. Hence, we follow a rule-based approach for predicting sentiment of a tweet. An approach like ours provides a generic way of solving sentiment classification problems in micro-blogs.

3 Architecture

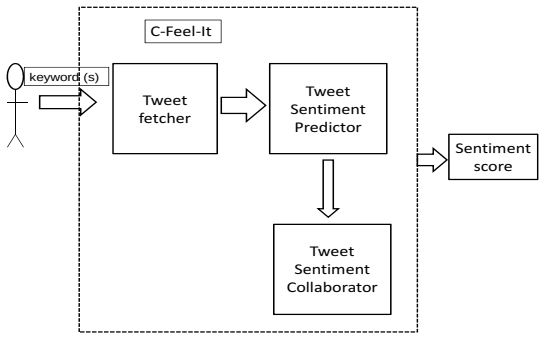


Figure 1: Overall Architecture

The overall architecture of C-Feel-It is shown in Figure 1. C-Feel-It is divided into three parts: **Tweet Fetcher**, **Tweet Sentiment Predictor** and **Tweet Sentiment Collaborator**. All predictions are positive, negative or objective/neutral. C-Feel-It offers two implementations of a rule-based sentiment prediction system. We refer to them as version 1 and 2. The two versions differ in the Tweet Sentiment Predictor module. This section describes different modules of C-Feel-It and is organized as follows. In subsections 3.1, 3.2 & 3.3, we describe the three functional blocks of C-FeeL-It. In subsection 3.4, we explain how four lexical resources are mapped to the desired output labels. Finally, subsection 3.5 gives implementation details of C-Feel-It.

Input to C-Feel-It is a *search string* and a *version number*. The versions are described in detail in subsection 3.2.

Output given by C-Feel-It is two-level: tweet-wise prediction and overall prediction. For tweet-wise prediction, sentiment prediction by each of the resources is returned. On the other hand, overall prediction combines sentiment from all tweets to return the percentage of positive, negative and objective content retrieved for the search string.

3.1 Tweet Fetcher

Tweet fetcher obtains tweets pertaining to a search string entered by a user. To do so, we use live feeds from Twitter using an API [2]. The parameters passed to the API ensure that system receives the latest 50 tweets about the keyword in English. This API returns results in XML format which we parse using a Java SAX parser.

3.2 Tweet Sentiment Predictor

Tweet sentiment predictor predicts sentiment for a single tweet. The architecture of Tweet Sentiment Predictor is shown in Figure 2 and can be divided into three fundamental blocks: *Preprocessor, Emoticon-based Sentiment Predictor, Lexicon-based Sentiment Predictor* (refer Figure 3 & 4). The first two blocks are same for both the versions of C-Feel-It. The two versions differ in the working of the Lexicon-based Sentiment Predictor.

Preprocessor

The noisy nature of tweets is a classical challenge that any system working on tweets needs to encounter. Preprocessor deals with obtaining *clean tweets*. We do not deploy any spelling correction module. However, the preprocessor handles extensions and contractions found in tweets as follows.

Handling extensions: Extensions like '*bessssst*' are common in tweets. However, to look up resources, it is essential that these words are normalized to their dictionary equivalent. We replace consecutive occurrences of the same letter (if more than

[2] http://search.Twitter.com/search.atom

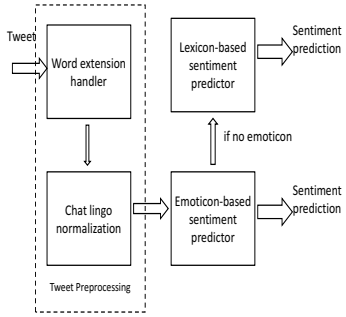


Figure 2: Tweet Sentiment Predictor: Version 1 and 2

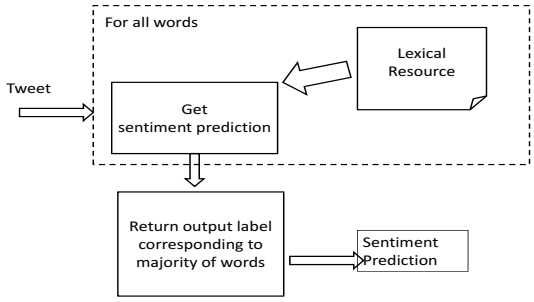


Figure 3: Lexicon-based Sentiment Predictor: C-Feel-It Version 1

three occurrences of the same letter) with a single letter and replace the word.

An important issue here is that extensions are in fact strong indicators of sentiment. Hence, we replace an extended word by two occurences of the contracted word. This gives a higher weight to the extended word and retains its contribution to the sentiment of the tweet.

Chat lingo normalization: Words used in chat/Internet language that are common in tweets are not present in the lexical resources. We use a dictionary downloaded from http://chat.reichards.net/ . A chat word is replaced by its dictionary equivalent.

Emoticon-based Sentiment Predictor

Emoticons are visual representations of emotions frequently used in the user-generated content on the Internet. We observe that in most cases, emoticons pinpoint the sentiment of a tweet. We use an emoticon mapping from http://chat.reichards.net/smiley.shtml. An emoticon is mapped to an output label: positive or negative. A tweet containing one of these emoticons that can be mapped to the desired output labels directly. While we understand that this heuristic does not work in case of sarcastic tweets, it does provide a benefit in most cases.

Lexicon-based Sentiment Predictor

For a tweet, the Lexicon-based Sentiment Predictor gives a prediction each for four resources. In addition, it returns one prediction which combines the four predictions by weighting them on the basis of their accuracies. We remove stop words [3] from the tweet and stem the words using Lovins stemmer (Lovins, 1968). Negation in tweets is handled by inverting sentiment of words after a negating word. The words *'no', 'never', 'not'* are considered negating words and a context window of three words after a negative words is considered for inversion. The two versions of C-Feel-It vary in their Lexicon-based Sentiment Predictor. Figure 3 shows the Lexicon-based Sentiment Predictor for version 1. For each word in the tweet, it gets the prediction from a lexical resource. We use the intuition that a positive tweet has positive words outnumbering other words, a negative tweet has negative words outnumbering other words and an objective tweet has objective words outnumbering other words.

Figure 4 shows the Lexicon-based Sentiment Predictor for version 2. As opposed to the earlier version, version 2 gets prediction from the lexical resource for some words in the tweet. This is because certain parts-of-speech have been found to be better indicators of sentiment (Pang and Lee, 2004). A tweet is annotated with parts-of-speech tags and the POS bi-tags (i.e. *a pattern of two consecutive POS*) are marked. The words corresponding to a set of optimal POS bi-tags are retained and only these words used for lookup. The prediction for a tweet uses majority vote-based approach as for version 1. The optimal POS bi-tags have been derived experimentally by using top 10% features on information gain-based-pruning classifier on polarity dataset by (Pang and Lee, 2005). We used Stanford POS tagger(Tou,

[3] http://www.ranks.nl/resources/stopwords.html

2000) for tagging the tweets.

Note: The dataset we use to find optimal POS bi-tags consists of movie reviews. We understand that POS bi-tags hence derived may not be universal across domains.

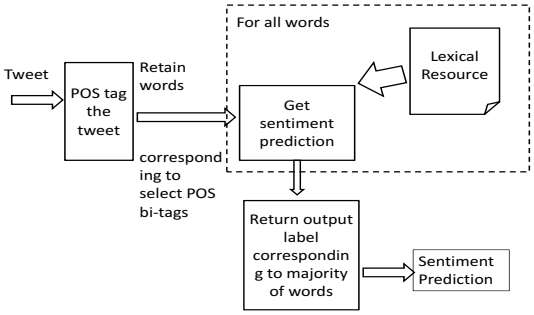


Figure 4: Lexicon-based Sentiment Predictor: C-Feel-It Version 2

3.3 Tweet Sentiment Collaborator

Based on predictions of individual tweets, the Tweet Sentiment Collaborator gives overall prediction with respect to a keyword in form of percentage of positive, negative and objective content. This is on the basis of predictions by each resource by weighting them according to their accuracies. These weights have been assigned to each resource based on experimental results. For each resource, the following scores are determined.

$$posscore[r] = \sum_{i=1}^{m} p_i w_{pi}$$

$$negscore[r] = \sum_{i=1}^{m} n_i w_{ni}$$

$$objscore[r] = \sum_{i=1}^{m} o_i w_{oi}$$

$where$

$posscore[r]$ = Positive score for search string r

$negscore[r]$ = Negative score for search string r

$objscore[r]$ = Objective score for search string r

m = Number of resources used for prediction

p_i, n_i, o_i = Positive,negative & objective count of tweet predicted respectively using resource i

w_{pi}, w_{ni}, o_{oi} = Weights for respective classes derived for each resource i

We normalize these scores to get the final positive, negative and objective pertaining to search string r. These scores are represented in form of percentage.

3.4 Resources

Sentiment-based lexical resources annotate words/concepts with polarity. The completeness of these resources individually remains a question. To achieve greater coverage, we use four different sentiment-based lexical resources for C-Feel-It. They are described as follows.

1. SentiWordNet (Esuli and Sebastiani, 2006) assigns three scores to synsets of WordNet: positive score, negative score and objective score. When a word is looked up, the label corresponding to maximum of the three scores is returned. For multiple synsets of a word, the output label returned by majority of the synsets becomes the prediction of the resource.
2. Subjectivity lexicon (Wiebe et al., 2004) is a resource that annotates words with tags like parts-of-speech, prior polarity, magnitude of prior polarity (weak/strong), etc. The prior polarity can be positive, negative or neutral. For prediction using this resource, we use this prior polarity.
3. Inquirer (Stone et al., 1966) is a list of words marked as positive, negative and neutral. We use these labels to use Inquirer resource for our prediction.
4. Taboada (Taboada and Grieve, 2004) is a word-list that gives a count of collocations with positive and negative seed words. A word closer to a positive seed word is predicted to be positive and vice versa.

3.5 Implementation Details

The system is implemented in JSP (JDK 1.6) using NetBeans IDE 6.9.1. For the purpose of tweet annotation, an internal interface was written in PHP 5 with MySQL 5.0.51a-3ubuntu5.7 for storage.

4 System Analysis

4.1 Evaluation Data

For the purpose of evaluation, a total of 7000 tweets were downloaded by using popular trending topics of 20 domains (like books, movies, electronic gadget, etc.) as keywords for searching tweets. In order to download the tweets, we used the API provided by Twitter [4] that crawls latest tweets pertaining to keywords.

Human annotators assigned to a tweet one out of 4 classes: positive, negative, objective and objective-spam.

[4] http://search.twitter.com/search.atom?

A tweet is assigned to objective-spam category if it contains promotional links or incoherent text which was possibly not created by a human user. Apart from these nominal class labels, we also assigned the positive/negative tweets scores ranging from +2 to -2 with +2 being the most positive and -2 being the most negative score respectively. If the tweet belongs to the objective category, a value of zero is assigned as the score.

The spam category has been included in the annotation as a future goal of modeling a spam detection layer prior to the sentiment detection. However, the current version of C-Feel-It does not have a spam detection module and hence for evaluation purpose, we use only the data belonging to classes other than objective-spam.

4.2 Qualitative Analysis

In this section, we perform a qualitative evaluation of actual results returned by C-Feel-It. The errors described in this section are in addition to the errors due to misspellings and informal language. These erroneous results have been obtained from both version 1 and 2. They have been classified into eleven categories and explained henceforth.

4.2.1 Sarcastic Tweets

Tweet: *Hoge, Jaws, and Palantonio are brilliant together talking X's and O's on ESPN right now.*
Label by C-Feel-It: Positive
Label by human annotator: Negative

The sarcasm in the above tweet lies in the use of a positive word '*brilliant*' followed by a rather trivial action of '*talking Xs and Os*'. The positive word leads to the prediction by C-Feel-It where in fact, it is a negative tweet for the human annotator.

4.2.2 Lack of Sense Understanding

Tweet: *If your tooth hurts drink some pain killers and place a warm/hot tea bag like chamomile on your tooth and hold it. it will relieve the pain*
Label by C-Feel-It: Negative

This tweet is objective in nature. The words 'pain', 'killers', etc. in the tweet give an indication to C-Feel-It that the tweet is negative. This misguided implication is because of multiple senses of these words (for example, '*pain*' can also be used in the sentence '*symptoms of the disease are body pain and irritation in the throat*' where it is non-sentiment-bearing). The lack of understanding of word senses and being unable to distinguish between them leads to this error.

4.2.3 Lack of Entity Specificity

Tweet: *Casablanca and a lunch comprising of rice and fish: a good sunday*
Keyword: Casablanca
Label by C-Feel-It: Positive
Label by human annotator: Objective

In the above tweet, the human annotator understood that though the tweet contains the keyword '*Casablanca*', it is not Casablanca about which sentiment is expressed. The system finds a positive word '*good*' and marks the tweet as positive. This error arises because the system cannot find out which sentence/parts of sentence is expressing opinion about the target entity.

4.2.4 Coverage of Resources

Tweet: *I'm done with this bullshit. You're the psycho not me.*
Label by SentiWordNet: Negative
Label by Taboada/Inquirer: Objective
Label by human annotator: Negative

On manual verification, it was observed that an entry for the emotion-bearing word '*bullshit*' is present in SentiWordNet while Inquirer and Taboada resource do not have them. This shows that the coverage of the lexical resource affects the performance of a system and may introduce errors.

4.2.5 Absence of Named Entity Recognition

Tweet: *@user I don't think I need to guess, but ok, close encounters of the third kind? Lol*
Entity: Close encounters of the third kind
Label by C-Feel-It: Positive

The words comprising the name of the film '*Close encounters of the third kind*' are also looked up. Inability to identify the named entity leads the system into this trap.

4.2.6 Requirement of World Knowledge

Tweet: *The soccer world cup boasts an audience twice that of the Summer Olympics.*
Label by C-Feel-It: Negative

To judge the opinion of this tweet, one requires an understanding of the fact that larger the audience, more favorable it is for a sports tournament. This world knowledge is important for a system that aims to handle tweets like these.

4.2.7 Mixed Emotion Tweets

Tweet: *oh but that last kiss tells me it's goodbye, just like nothing happened last night. but if i had one chance, i'd do it all over again*
Label by C-Feel-It: Positive

The tweet contains emotions of positive as well as negative variety and it would in fact be difficult for a human as well to identify the polarity. The mixed nature of the tweet leads to this error by the system.

4.2.8 Lack of Context

Tweet: *I'll have to say it's a tie between Little Women or To kill a Mockingbird*

Label by C-Feel-It: Negative
Label by human user: Positive

The tweet has a sentiment which will possibly be clear in the context of the conversation. Going by the tweet alone, while one understands that an comparative opinion is being expressed, it is not possible to tag it as positive or negative.

4.2.9 Concatenated Words

Tweet: *To Kill a Mockingbird is a #goodbook.*
Label by C-Feel-It: Negative

The tweet has a hashtag containing concatenated words '*goodbook*' which get overlooked as out-of-dictionary words and hence, are not used for sentiment prediction. The sentiment of '*good*' is not detected.

4.2.10 Interjections

Tweet: *Oooh. Apocalypse Now is on bluray now.*
Label by C-Feel-It: Objective
Label by human user: Positive

The extended interjection '*Oooh*' is an indicator of sentiment. Since it does not have a direct prior polarity, it is not present in any of the resources. However, this interjection is an important carrier of sentiment.

4.2.11 Comparatives

Tweet: *The more years I spend at Colbert Heights..the more disgusted I get by the people there. I'm soooo ready to graduate.*
Label by C-Feel-It: Positive
Label by human user: Negative

The comparatives in the sentence expressed by '*..more disgusted I get..*' have to be handled as a special case because '*more*' is an intensification of the negative sentiment expressed by the word '*disgusted*'.

5 Summary & Future Work

In this paper, we described a system which categorizes live tweets related to a keyword as positive, negative and objective based on the predictions of four sentiment-based resources. We also presented a qualitative evaluation of our system pointing out the areas of improvement for the current system.

A sentiment analyzer of this kind can be tuned to take inputs from different sources on the internet (for example, wall posts on facebook). In order to improve the quality of sentiment prediction, we propose two additions. Firstly, while we use simple heuristics to handle extensions of words in tweets, a deeper study is required to decipher the pragmatics involved. Secondly, a spam detection module that eliminates promotional tweets before performing sentiment detection may be added to the current system. Our goal with respect to this system is to deploy it for predicting share market values of firms based on sentiment on social networks with respect to related entitites.

Acknowledgement

We thank Akshat Malu and Subhabrata Mukherjee, IIT Bombay for their assistance during generation of evaluation data.

References

Go Alec, Huang Lei, and Bhayani Richa. 2009a. Twitter sentiment classification using distant supervision. Technical report, Standford University.

Go Alec, Bhayani Richa, Raghunathan Karthik, and Huang Lei. 2009b. May.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC-06*, Genova, Italy.

Andreas M. Kaplan and Michael Haenlein. 2010. The early bird catches the news: Nine things you should know about micro-blogging. *Business Horizons*, 54(2):05 – 113.

Julie B. Lovins. 1968. Development of a Stemming Algorithm. June.

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*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL-05*.

Vladimir Prelovac. 2010. Top social media sites. Web, May.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press.

Maite Taboada and Jack Grieve. 2004. Analyzing Appraisal Automatically. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, pages 158–161, Stanford, US.

2000. *Enriching the knowledge sources used in a maximum entropy part-of-speech tagger*, Stroudsburg, PA, USA. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computional Linguistics*, 30:277–308, September.

IMASS: An Intelligent Microblog Analysis and Summarization System

Jui-Yu Weng Cheng-Lun Yang Bo-Nian Chen Yen-Kai Wang Shou-De Lin
Department of Computer Science and Information Engineering
National Taiwan University
{r98922060,r99944042,f92025,b97081,sdlin}@csie.ntu.edu.tw

Abstract

This paper presents a system to summarize a Microblog post and its responses with the goal to provide readers a more constructive and concise set of information for efficient digestion. We introduce a novel two-phase summarization scheme. In the first phase, the post plus its responses are classified into four categories based on the intention, interrogation, sharing, discussion and chat. For each type of post, in the second phase, we exploit different strategies, including opinion analysis, response pair identification, and response relevancy detection, to summarize and highlight critical information to display. This system provides an alternative thinking about machine-summarization: by utilizing AI approaches, computers are capable of constructing deeper and more user-friendly abstraction.

1 Introduction

As Microblog services such as Twitter have become increasingly popular, it is critical to reconsider the applicability of the existing NLP technologies on this new media sources. Take summarization for example, a Microblog user usually has to browse through tens or even hundreds of posts together with their responses daily, therefore it can be beneficial if there is an intelligent tool assisting summarizing those information.

Automatic text summarization (ATS) has been investigated for over fifty years, but the majority of the existing techniques might not be appropriate for Microblog write-ups. For instance, a popular kind of approaches for summarization tries to identify a subset of information, usually in sentence form, from longer pieces of writings as summary (Das and Martins, 2007). Such extraction-based methods can hardly be applied to Microblog texts because many posts/responses contain only one sentence.

Below we first describe some special characteristics that deviates the Microblog summarization task from general text summarization.

a. The number of sentences is limited, and sentences are usually too short and casual to contain sufficient structural information or cue phrases. Unlike normal blogs, there is a strict limitation on the number of characters for each post (e.g. 140 characters for Twitter and Plurk maximum). Microblog messages cannot be treated as complete documents so that we cannot take advantage of the structural information. Furthermore, users tend to regard Microblog as a chatting board. They write casually with slangs, jargons, and incorrect grammar.
b. Microblog posts can serve several different purposes. At least three different types of posts are observed in Microblogs, expressing feeling, sharing information, and asking questions. Structured language is not the only means to achieve those goals. For example, people sometimes use attachment, as links or files, for sharing, and utilize emoticons and pre-defined qualifiers to express their feelings. The diversity of content differ Microblogs from general news articles. Consequently, using one mold to fit all types of Microblog posts is not sufficient. Different summarization schemes for posts with different purposes are preferred.
c. Posts and responses in Microblogs are more similar to a multi-persons dialogue corpus. One of the main purposes of a Microblog is to serve as the fast but not instant communication channel among multiple users. Due to the free-chatting, multi-user characteristics, the topic of a post/response thread can drift quickly. Sometimes, the topic of discussion at the end of the thread is totally unrelated to that of the post.

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 133–138,
Portland, Oregon, USA, 21 June 2011.

This paper introduces a framework that summarizes a post with its responses. Motivated by the abovementioned characteristics of Microblogs, we plan to use a two-phase summarization scheme to develop different summarization strategies for different type of posts (see Figure 1). In the first phase, a post will be automatically classified into several categories including interrogation, discussion, sharing and chat based on the intention of the users. In the second phase, the system chooses different summarization components for different types of posts.

The novelties of this system are listed below.

1. Strategically, we propose an underlying 2-phase framework for summarizing Microblog posts. The system can be accessed online at http://mslab.csie.ntu.edu.tw/~fishyz/plurk/.
2. Tactically, we argue that it is possible to integrate post-intention classification, opinion analysis, response relevancy and response-pair mining to create an intelligent summarization framework for Microblog posts and responses. We also found that the content features are not as useful as the temporal or positional features for text mining in Microblog.
3. Our work provides an alternative thinking about ATS. It is possible to go beyond the literal meaning of summarization to exploit advanced text mining methods to improve the quality and usability of a summarization system.

2 Summarization Framework and Experiments

Below we discuss our two-phase summarization framework and the experiment results on each individual component. Note that our experiments were tested on the Plurk dataset, which is one of the most popular micro-blogging platforms in Asia.

Our observation is that Microblog posts can have different purposes. We divide them into four categories, *Interrogation*, *Sharing*, *Discussion*, and *Chat*.

The *Interrogation* posts are questions asked in public with the hope to obtain some useful answers from friends or other users. However, it is very common that some repliers do not provide meaningful answers. The responses might serve the purpose for clarification or, even worse, have nothing to do with the question. Hence we believe the most appropriate summarization process for this kind of posts is to find out which replies really respond to the question. We created a *response relevance detection* component to serve as its summarization mechanism.

The *Sharing* posts are very frequently observed in Microblog as Microbloggers like to share interesting websites, pictures, and videos with their friends. Other people usually write down their comments or feelings on the shared subjects in the responses. To summarize such posts, we obtain the statistics on how many people have positive, neutral, and negative attitude toward the shared subjects. We introduce the *opinion analysis* component that provides the analysis on whether the information shared is recommended by the respondents.

We also observe that some posts contain characteristics of both *Interrogation* and *Sharing*. The users may share a hyperlink and ask for others' opinions at the same time. We create a category named *Discussion* for these posts, and apply both response ranking and opinion analysis engines on this type of posts.

Finally, there are posts which simply act as the solicitation for further chat. For example, one user writes "so sad…" and another replies "what happened". We name this type of posts/responses as *Chat*. This kind of posts can sometimes involve multiple persons and the topic may gradually drift to a different one. We believe the plausible summarization strategy is to group different messages based on their topics. Therefore for *Chat* posts, we designed a *response pair identification* system to accomplish such goal. We group the related responses together for display, and the number of groups represents the number of different topics in this thread.

Figure 1 shows the flow of our summarization

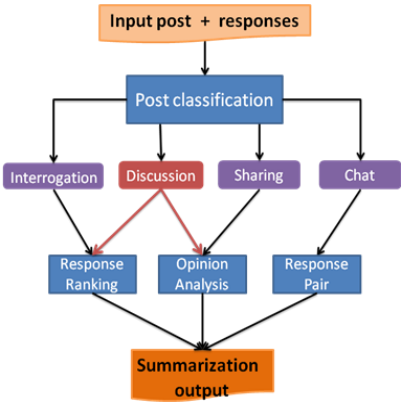


Figure 1. System architecture

framework. When an input post with responses comes in, the system first determines its intention, based on which the system adopts proper strategies for summarization. Below we discuss the technical parts of each sub-system with experiment results.

2.1 Post Intention Classification

This stage aims to classify each post into four categories, *Interrogation*, *Sharing*, *Discussion*, and *Chat*. One tricky issue is that the *Discussion* label is essentially a combination of interrogation and sharing labels. Therefore, simply treating it as an independent label and use a typical multi-label learning method can hurt the performance. We obtain 76.7% (10-fold cross validation) in accuracy by training a four-class classifier using the 6-gram character language model. To improve the performance, we design a decision-tree based framework that utilizes both manually-designed rules and discriminant classification engine (see Figure 2). The system first checks whether the posts contains URLs or pointers to files, then uses a binary classifier to determine whether the post is interrogative.

For the experiment, we manually annotate 6000 posts consisting of 1840 *interrogation*, 2002 *sharing*, 1905 *chat*, and 254 *discussion* posts. We train a 6-gram language model as the binary interrogation classifier. Then we integrate the classifier into our system and test on 6000 posts to obtain a testing accuracy of 82.8%, which is significantly better than 76.7% with multi-class classification.

2.2 Opinion Analysis

Opinion analysis is used to evaluate public preference on the shared subject. The system classifies responses into 3 categories, positive, negative, and neutral.

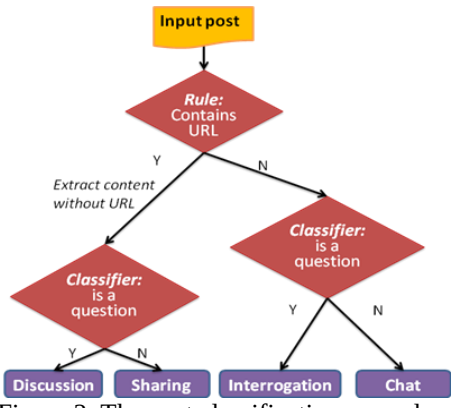


Figure 2. The post classification procedure

Here we design a two-level classification framework using Naïve-Bayes classifiers which takes advantage of the learned 6-gram language model probabilities as features. First of all, we train a binary classifier to determine if a post or a reply is opinionative. This step is called the subjectivity test. If the answer is yes, we then use another binary classifier to decide if the opinion is positive or negative. The second step is called the polarity test.

For subjectivity test, we manually annotate 3244 posts, in which half is subjective and half is objective. The 10-fold cross validation shows average accuracy of 70.5%.

For polarity test, we exploit the built-in emoticons in Plurk to automatically extract posts with positive and negative opinions. We collect 10,000 positive and 10,000 negative posts as training data to train a language model of Naïve Bayes classifier, and evaluate on manually annotated data of 3121 posts, with 1624 positive and 1497 negative to obtain accuracy of 0.722.

2.3 Response Pair Identification

Conversation in micro-blogs tends to diverge into multiple topics as the number of responses grows. Sometimes such divergence may result in responses that are irrelevant to the original post, thus creating problems for summarization. Furthermore, because the messages are usually short, it is difficult to identify the main topics of these dialogue-like responses using only keywords in the content for summarization. Alternatively, we introduce a subcomponent to identify Response Pairs in micro-blogs. A Response Pair is a pair of responses that the latter specifically responds to the former. Based on those pairs we can then form clusters of messages to indicate different group of topics and mes-

Feature	*Description*	*Weight*
Backward Referencing	Latter response content contains former responder's display name	0.055
Forward Referencing of user name	Former response contains latter response's author's user name	0.018
Response position difference	Number of responses in between responses	**0.13**
Content similarity	Contents' cosine similarity using n-gram models.	0.025
Response time difference	Time difference between responses in seconds	0.012

Table 1. Feature set with their description and weights

sages.

Looking at the content of micro-blogs, we observe that related responses are usually adjacent to each other as users tend to closely follow whether their messages are responded and reply to the responses from others quickly. Therefore besides content features, we decide to add the temporal and ordering features (See Table 1) to train a classifier that takes a pair of messages as inputs and return whether they are related. By identifying the response pairs, our summarization system is able to group the responses into different topic clusters and display the clusters separately. We believe such functionality can assist users to digest long Microblog discussions.

For experiment, the model is trained using LIBSVM (Chang and Lin, 2001) (RBF kernel) with 6000 response pairs, half of the training set positive and half negative. The positive data can be obtained automatically based on Plurk's built in annotation feature. Responses with @user_name string in the content are matched with earlier responses by the author, user_name. Based on the learned weights of the features, we observe that content feature is not very useful in determining the response pairs. In a Microblog dialogue, respondents usually do not repeat the question nor duplicate the keywords. We also have noticed that there is high correlation between the responses relatedness and the number of other responses between them. For example, users are less likely to respond to a response if there have been many replies about this response already. Statistical analysis on positive training data shows that the average number of responses between related responses is 2.3.

We train the classifier using 6000 automatically-extracted pairs of both positive and negative instances. We manually annotated 1600 pairs of data for testing. The experiment result reaches 80.52% accuracy in identifying response pairs. The baseline model which uses only content similarity feature reaches only 45% in accuracy.

2.4 Response Relevance Detection

For interrogative posts, we think the best summary is to find out the relevent responses as potential answers.

We introduce a *response relevancy detection* component for the problem. Similar to previous components, we exploit a supervised learning approach and the features' weights, learned by LIBSVM with RBF kernel, are shown in Table 2.

Feature	*Weight*
Response position	**0.170**
Response time difference	0.008
Response length	0.003
Occurrence of interrogative words	0.023
Content similarity	0.023

Table 2. Feature set and their weights

Temporal and Positional Features

A common assertion is that the earlier responses have higher probability to be the answers of the question. Based on the learned weights, it is not surprising that most important feature is the position of the response in the response hierarchy. Another interesting finding by our system is that meaningful replies do not come right away. Responses posted within ten seconds are usually for chatting/clarification or ads from robots.

Content Features

We use the length of the message, the cosine similarity of the post and the responses, and the occurrence of the interrogative words in response sentences as content features.

Because the interrogation posts in Plurk are relatively few, we manually find a total of 382 positive and 403 negative pairs for training and use 10-fold cross validation for evaluation.

We implement the component using LIBSVM (RBF Kernel) classifier. The baseline is to always select the first response as the only relevant answer. The results show that the accuracy of baseline reaches 67.4%, far beyond that of our system 73.5%.

3 System Demonstration

In this section, we show some snapshots of our summarization system with real examples using Plurk dataset. Our demo system is designed as a

Figure 3. The IMASS interface

Original post and responses:

iantearle	**asks**	**if anyone can send him the Helvetica font for mac???!!! PLEASSSSSEEEEE**
trinitydechou	has	it, but not at mac.... so if you don't get it later, let me know I can fire it over to you.
iantearle	will	let you know!
trinitydechou	:	bounces on you
iantearle	:	naughty!
iantearle	:	i do
iantearle	:	Ian check your email, i sent you the fonts

Summarization:

iantearle	**asks**	**if anyone can send him the Helvetica font for mac???!!! PLEASSSSSEEEEE**
trinitydechou	has	it, but not at mac.... so if you don't get it later, let me know I can fire it over to you.
iantearle	:	Ian check your email, i sent you the fonts

Figure 4. An example of interrogative post.

search engine (see Figure 3). Given a query term, our system first returns several posts containing the query string under the search bar. When one of the posts is selected, it will generate a summary according to the detected intention and show it in a pop-up frame. We have recorded a video demonstrating our system. The video can be viewed at http://imss-acl11-demo.co.cc/.

For interrogative posts, we perform the response relevancy detection. The summary contains the question and relevant answers. Figure 4 is an example of summary of an interrogative post. We can see that responses other than the first and the last responses are filtered because they are less relevant to the question.

For sharing posts, the summary consists of two parts. A pie chart that states the percentage of each opinion group is displayed. Then the system picks three responses from the majority group or one response from each group if there is no significant difference. Figure 5 is an example that most friends of the user *dfrag* give positive feedback to the shared video link.

For discussion posts, we combine the response relevancy detection subsystem and the opinion analysis sub-system for summarization. The former first eliminates the responses that are not likely to be the answer of the post. The latter then generates a summary for the post and relevant responses. The result is similar to sharing posts.

For chat posts, we apply the response pair identification component to generate the summary. In the example, Figure 6, the original Plurk post is about one topic while the responses diverge to one

Original post and responses:

dfrag	**:**	**http://www.youtube.com/watch?v=KgISPI7g14Q Yo, fo' real. Boo yaakasha!**
CatC	says	mornin'
dfrag	:	That yawning smiley made me yawn! WTF!?
CatC	says	lol
CatC	says	my work here is done.
dfrag	:	LoL
TVisio	:	Who do you feel more sorry for? One is an anal retentive curmudgeon, the other is an anal Yo!
Snailrind	loves	Sacha Baron Cohen.
Snailrind	thinks	he's at his best when his interviewee has no idea who he is.
1bzymama	:	Meh. It was alright
Snailrind	says	"is it because I is black?"

Summarization:

Recommend!!

dfrag	**:**	**http://www.youtube.com /watch?v=KgISPI7g14Q Yo, fo' real. Boo yaakasha!**	***negative***
CatC	says	lol	*positive*
CatC	says	my work here is done.	*positive*
dfrag	:	LoL	*positive*

Figure 5. An example of sharing post.

Original post and responses:

TwilaMarie	**:**	**time to catch up on Dollhouse. lovin tv-dome.net!**
chaotixfusion	:	
TwilaMarie	:	hiya Rio! Git your message this evening
TwilaMarie	:	got8 LOL
rthefish	:	Having a hard time getting up in that show. I have the past 2 I need to watch though.
TwilaMarie	:	i really like it! love all the free pc-tv places to catch up on things. spent the last two days doing just that
chaotixfusion	:	ahh okay , at least you got it

Summarization:

TwilaMarie(*TwilaMarie*): time to catch up on Dollhouse. lovin tv-dome.net!
Rio(*chaotixfusion*):
TwilaMarie(*TwilaMarie*): hiya Rio! Git your message this evening
TwilaMarie(*TwilaMarie*): got8 LOL
Rio(*chaotixfusion*): ahh okay , at least you got it

TwilaMarie(*TwilaMarie*): time to catch up on Dollhouse. lovin tv-dome.net!
rthefish(*rthefish*): Having a hard time getting up in that show. I have the past 2 I need to watch though.
TwilaMarie(*TwilaMarie*): i really like it! love all the free pc-tv places to catch up on things. spent the last two days doing just that

Figure 6. An Example of chat post

or more unrelated topics. Our system clearly separates the responses into multiple groups. This representation helps the users to quickly catch up with the discussion flow. The users no longer have to read interleaving responses from different topics and guess which topic group a response is referring to.

4 Related Work

We have not seen many researches focusing on the issues of Microblog summarization. We found only one work that discusses about the issues of summarization for Microblogs (Sharifi et al., 2010). Their goal, however, is very different from ours as they try to summarize multiple posts and do not consider the responses. They propose the Phrase Reinforcement Algorithm to find the most commonly used phrase that encompasses the topic phrase, and use these phrases to compose the summary. They are essentially trying to solve a multi-document summarization problem while our problem is more similar to short dialog summarization because the dialogue nature of Microblogs is one of the most challenging part that we tried to overcome.

In dialogue summarization, many researchers have pointed out the importance of detecting response pairs in a conversation. Zechner (2001) performs an in depth analysis and evaluation in the area of open domain spoken dialogue summarization. He uses decision tree classifier with lexical features like POS tags to identify questions and applies heuristic rules like maximum distance between speakers to extract answers. Shrestha and McKeown (2004) propose a supervised learning method to detect question-answer pairs in Email conversations. Zhou and Hovy (2005) concentrates on summarizing dialogue-style technical internet relay chats using supervised learning methods. Zhou further clusters chat logs into several topics and then extract some essential response pairs to form summaries. Liu et al. (2006) propose to identify question paragraph via analyzing each participant's status, and then use cosine measure to select answer paragraphs for online news dataset.

The major differences between our components and the systems proposed by others lie in the selection of features. Due to the intrinsic difference between the writing styles of Microblog and other online sources, our experiments show that the content feature is not as useful as the position and temporal features.

5 Conclusion

In terms of length and writing styles, Microblogs possess very different characteristics than other online information sources such as web blogs and news articles. It is therefore not surprising that different strategies are needed to process Microblog messages. Our system uses an effective strategy to summarize the post/response by first determine the intention and then perform different analysis depending on the post types. Conceptually, this work intends to convey an alternative thinking about machine-summarization. By utilizing text mining and analysis techniques, computers are capable of providing more intelligent summarization than information condensation.

Acknowledgements

This work was supported by National Science Council, National Taiwan University and Intel Corporation under Grants NSC99-2911-I-002-001, 99R70600, and 10R80800.

References

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM : a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Dipanjan Das and André F.T. Martins. 2007. A Survey on Automatic Text Summarization. Literature Survey for the Language and Statistics II Course. CMU.

Chuanhan Liu, Yongcheng Wang, and Fei Zheng. 2006. Automatic Text Summarization for Dialogue Style. In Proceedings of the IEEE International Conference on Information Acquisition. 274-278

Beaux Sharifi, Mark A. Hutton, and Jugal Kalita. 2010. Summarizing Microblogs Automatically. In Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT). 685-688

Lokesh Shrestha and Kathleen McKeown. 2004. Detection of Question-Answer Pairs in Email Conversations. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010).

Klaus Zechner. 2001. Automatic Generation of Concise Summaries of Spoken Dialogues in Unrestricted Domains. In Proceedings of the 24th ACM-SIGIR International Conference on Research and Development in Information Retrieval. 199-207.

Liang Zhou and Eduard Hovy. 2005. Digesting virtual geek culture: The summarization of technical internet relay chats, in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005). 298-305.

An Interface for Rapid Natural Language Processing Development in UIMA

Balaji R. Soundrarajan, Thomas Ginter, Scott L. DuVall
VA Salt Lake City Health Care System and University of Utah
balaji@cs.utah.edu, {thomas.ginter, scott.duvall}@utah.edu

Abstract

This demonstration presents the Annotation Librarian, an application programming interface that supports rapid development of natural language processing (NLP) projects built in Apache Unstructured Information Management Architecture (UIMA). The flexibility of UIMA to support all types of unstructured data – images, audio, and text – increases the complexity of some of the most common NLP development tasks. The Annotation Librarian interface handles these common functions and allows the creation and management of annotations by mirroring Java methods used to manipulate Strings. The familiar syntax and NLP-centric design allows developers to adopt and rapidly develop NLP algorithms in UIMA. The general functionality of the interface is described in relation to the use cases that necessitated its creation.

1 Introduction

In the days when public libraries were the center of information exchange, the job of the librarian was to serve as an interface between the complex library system and the average user. The librarian made it possible for one to access specific sources of information without memorizing the Dewey Decimal System or flipping through the card catalog. Analogous to the great librarians of yesteryear, the Annotation Librarian serves the average Java developer in the creation and management of annotations within natural language processing (NLP) projects built using the open source Apache Unstructured Information Management Architecture (UIMA)[1].

Many NLP tasks are performed in processing steps that build upon one another. Systems designed in this fashion are called *pipelines* because text is processed and then passed from one step to the next like water flowing through a pipe. Each step in the pipeline adds structured data on top of the text called *annotations*. An annotation can be as simple as a classification of a span of text or complex with attributes and mappings to coded values. As pipeline systems have caught on, the ability to standardize functionality in and even across pipelines has emerged. UIMA provides a powerful infrastructure for the storage, transport, and retrieval of document and annotation knowledge accumulated in NLP pipeline systems (Ferrucci 2004). UIMA provides tools that allow testing and visualizing system results, integration with Eclipse[2], and use of standard XML description files for maintainability and interoperability. Because UIMA provides the underlying data model for storing meta-data and annotations with document text and the interface for interacting between processing steps, it has become a popular platform for the development of reusable NLP systems (D'Avolio 2010, Coden 2009, Savova 2008). The most notable example of UIMA capabilities is Watson, the question-answering system that competed and won two Jeopardy! matches against the all-time-winning human champions (Ferrucci 2010).

In addition to its successful implementations in NLP, UIMA supports all types of unstructured information – video, audio, images, etc – and so all UIMA constructs generalize beyond text. While handling multiple data types increases the utility of the framework, developers new to UIMA may feel they need to understand the entire framework before being able to distinguish and focus solely on text. The Annotation Librarian aids both novice and experienced UIMA developers by providing intuitive and NLP-centric functionality.

[1] Apache UIMA is available from http://uima.apache.org/

[2] Eclipse Development Platform is available from http://www.eclipse.org

Proceedings of the ACL-HLT 2011 System Demonstrations, pages 139–144,
Portland, Oregon, USA, 21 June 2011.

2 System Overview

The Annotation Librarian was developed as an interface that synthesizes many of the most frequent annotation management tasks encountered in NLP system development and presents them in a manner easily accessed for those familiar with general Java development methods. It provides convenience methods that mirror Java String manipulation, allowing developers to seamlessly combine document text and annotations with the same commands familiar to anyone who has parsed a String or written a regular expression. Advanced functionality allows developers to examine spatial relationships among annotations and perform annotation pattern matching. In this demonstration, we present the general functionality of the Annotation Librarian in the context of the health care research projects that necessitated the creation of the interface.

The interface does not replace the need for NLP algorithms – developers have a plethora of patterns and decision rules, symbolic grammars, and machine learning techniques to create annotations. The Annotation Toolkit, though, provides a convenient way for developers to use existing annotations in their algorithms. This feeds the pipeline workflow that allows more complex annotations to be built in later processing steps using the annotations created in earlier steps.

The Annotation Librarian was developed and modified in response to four research projects in the health care domain that relied on NLP extraction of concepts from clinical text. The diversity of the different tasks in each of these use cases allowed the interface to include functionality common to different types of NLP system development. Interface functionality will be described as groups of related methods in the context of the four research projects and cover pattern matching, span overlap, relative position, annotation modification, and retrieval. All projects received Institutional Review Board approval for data use and only synthetic documents, not real patient records, are shown in the examples presented in this paper.

3 Pattern Matching

Name entity recognition and semantic classification tasks often require advanced concept identification techniques. Identifying mentions of prescriptions in a document using regular expressions, for example, would require hundreds of thousands of patterns for names of medicines and have to account for misspelling, abbreviations, and acronyms. Regular expressions are commonly used to solve simple NLP tasks, though, and can be utilized as part of a more complex information extraction strategy, such as understanding the context in which a term is used in the text (Garvin 2011, McCrae 2008, Frenz 2007, Chapman 2001). Negex (Chapman 2001) is an algorithm for identifying words before or after a term that suggest, for example, that a particular symptom is not present in a patient: "the patient has **no** fever." Other methods for understanding the context around terms include the use of an inclusion and exclusion list (Akbar 2009), temporal locality search (Grouin 2009), window search (Li 2009), and combinations of the above techniques (Hamon 2009).

The Annotation Librarian allows patterns to be built using existing annotations along with document text. This functionality combines the power of finding concepts that require complex means with the simplicity of regular expressions. The syntax mirrors that of the Java Pattern[3] and Matcher[4] classes, but allows for an extended regular expression grammar to identify Annotations. Pattern matching is accomplished in three phases: the input pattern is compiled, the document and annotations are analyzed for matches, and matches are returned along with span information.

A project identifying positive microbiology cultures will illustrate the use of pattern matching with the Annotation Librarian. Clinicians order microbiology cultures to determine whether a patient has a bacterial infection and which antibiotics would be most effective at treating the infection. Susceptibility is the measure of whether an antibiotic can effectively treat an organism or whether the organism is resistant to it.

A sample of microbiology report text is shown in Figure 1 and visualized annotations for the same sample are shown in Figure 2.

[3] Documented at http://download.oracle.com/javase/6/docs/api/java/util/regex/Pattern.html

[4] Documented at http://download.oracle.com/javase/6/docs/api/java/util/regex/Matcher.html

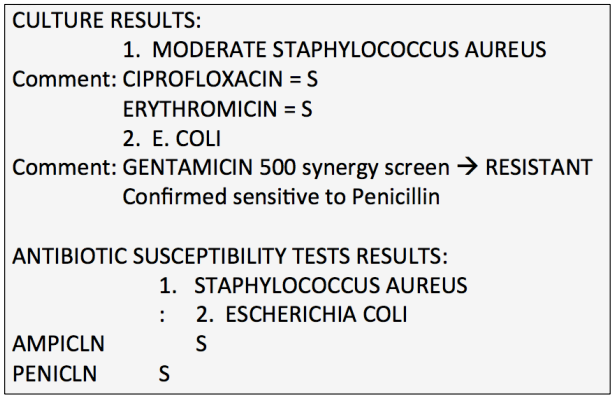


Figure 1: Microbiology Report Text

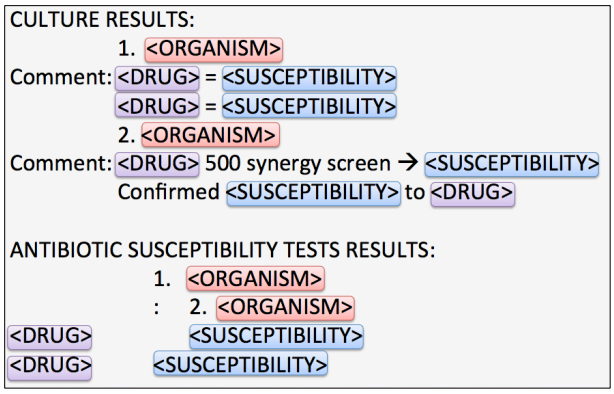


Figure 2: Annotated Report

To demonstrate pattern matching in this sample, the simple pattern of a drug annotation followed by an equals sign and then by a susceptibility annotation will be used.

3.1 Pattern Compilation

The pattern matching process begins when a new instance of an AnnotationPattern is created from the static `compile` method. AnnotationPattern is analogous to the Java Pattern[3] class.

```
AnnotationPattern susceptibilityPattern =
  AnnotationPattern.compile("pattern");
```

The method takes advantage of the UIMA implementation of annotations. Each annotation is an instance of a class that inherits from the UIMA class Annotation[5]. UIMA allows developers to create new types of annotations (in this example Organism, Antibiotic, and Susceptibility) that become Java classes.

[5] Documented at http://uima.apache.org/d/uimaj-2.3.1/api/index.html

The `compile` method input string pattern uses XML tags to represent Annotation classes and tag attributes to denote the name of method calls and return values in the format of:

```
<AnnotationClass methodName="expected value" />
```

When the extra constraint of matching on some method return values is not needed, the tag attribute is left blank. Portions of the pattern that are not contained in XML tags are compiled as Java regular expressions. For our example, the input pattern would be:

```
<Antibiotic /> = <Susceptibility />
```

or further constrained as:

```
<Antibiotic getMedName="ciprofloxacin" /> =
<Susceptibility getValue="S" />
```

which would only match if the particular medication (ciprofloxacin) and susceptibility (S) matched as well.

The pattern is converted into a finite state machine (FSM) in a process described by Fegaras (2005). With our pattern, a four-state FSM would be generated. To arrive in State 1, an Antibiotic annotation must match. To arrive in State 2, a regular expression for "=" must match. The Final State is reached when a matching Susceptibility annotation is found. Any other input would result in a transition back to the Start State.

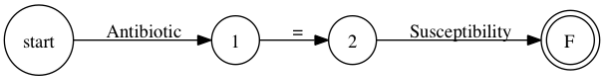


Figure 3: FSM for Antibiotic Susceptibility

3.2 Match Analysis

The second phase of pattern matching processes the document text and annotation set to determine if any matches can be found. This phase is triggered by a call to the static `matcher` method that returns a new instance of an AnnotationMatcher object. AnnotationMatcher is analogous to the Java Matcher[4] class.

```
AnnotationMatcher suscMatcher =
  susceptibilityPattern.matcher(cas);
```

This phase just checks to ensure that each annotation type has at least one instance in the document. Otherwise, a pattern match is not possible. Here, the `cas` parameter refers to the UIMA

Common Analysis Structure, the object containing the document and annotation information.

3.3 Finding Matches

The final phase of pattern matching involves a call to the AnnotationMatcher `find` method. This call results in a FSM traversal at the starting position parameter. Duplicate match candidates starting at the same point are pooled in each state. The candidate pool in each state is traversed with a binary search algorithm, which reduces overall traversal time. Note the following example in which a relationship is created through a new user-defined Annotation class type.

```
int position = 0 ;
while(suscMatcher.find(position))
{
    AntibioticSusceptibility annotation =
      new AntibioticSusceptibility(cas) ;
    annotation.setBegin(suscMatcher.start()) ;
    annotation.setEnd(suscMatcher.end()) ;
    annotation.addToIndexes() ;
    position = matcher.end() ;
}//while
```

Similar to the Java Matcher[4] `find` method, the first match is found from the starting position. The start and end positions are also set within the AnnotationMatcher instance object, which facilitates the creation of new annotations that span the complete pattern. The Annotation Librarian pattern matching functionality allows the inclusion of annotations, which provides an added level of power beyond regular expressions on text data only.

4 Retrieval

The retrieval methods allow developers to interact with annotations and metadata. This set of methods includes the ability to get the file name and path of the document, get all annotations in the document, and get all annotations of just a particular type.

```
getDocumentPath()
getAllAnnotations()
getAllAnnotationsOfType( int type )
```

Ejection fraction is a heart health measurement. An NLP system was developed to identify the ejection fraction from echocardiogram reports. In this project, the Annotation Librarian facilitated the extraction of specific annotation types (the section the concept was found in) in order to discover relevant concept-value pairs.

In Figure 4, ejection fraction annotations are shown in red and quantitative and qualitative values in blue. Because "systolic function" can be used to report ejection fraction, but only when referring to the left side of the heart, it was important to retrieve the section annotations and check the header.

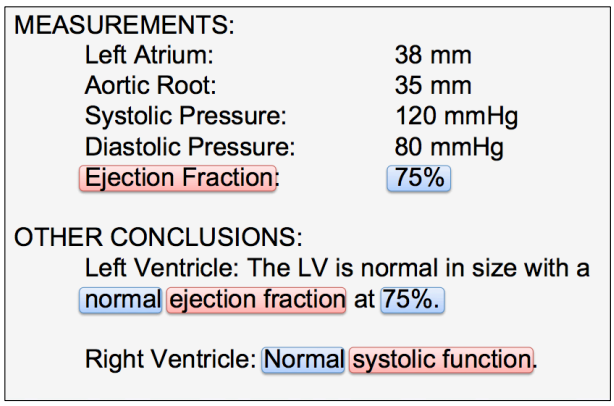


Figure 4: Annotated Echocardiogram Report

5 Annotation Modification

The annotation modification methods allow previous annotations to be altered by trimming whitespace and removing punctuation. While these are trivial tasks performed on Java Strings, an annotation is just a pointer to the text. Updating the annotation with the correct character span requires understanding of UIMA functions and can introduce errors if not done carefully. The Annotation Librarian ensures accuracy by handling these tasks with straightforward programmatic calls.

```
trim( Annotation annotation )
removePunctuation( Annotation annotation )
```

Identifying the organisms from the microbiology reports relied on splitting template text. The project described in Section 3 for pattern matching utilized the Annotation Librarian functionality to clean up spurious characters and whitespace included in annotations.

6 Span Overlap

This set of methods describes how annotations relate to each other spatially by answering questions such as: Does one annotation completely contain the other? Do the annotations overlap in the text? Do they both cover the same span of text?

```
overlaps( Annotation a1, Annotation a2 )
contains( Annotation a1, Annotation a2 )
coversSameSpan( Annotation a1, Annotation a2 )
```

In a system built for identifying medications in discharge summaries, the brand and generic names would often both be listed. Name entity recognition would end up mapping at multiple granularities – brand name only, generic name only, brand and generic name combinations, and even name and dose combinations. The span overlap methods were used to identify and combine overlapping names. Figure 5 shows the annotations that were found and resolved using span overlaps.

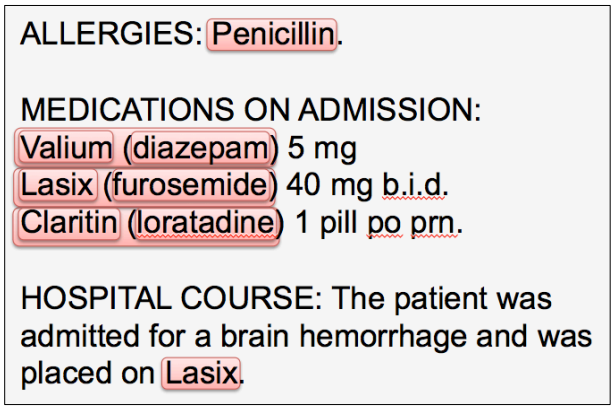


Figure 5: Medication Extraction Use Case

7 Relative Position

The relative position methods allow developers to access annotations based on their position in the text to each other. These methods can determine the next or previous adjacent annotation or the text that exists between two annotations. Often, a task required determining which concepts were found in the same sentence or finding all concepts in a certain section. Methods in this set provide functionality to find annotations that covering the span of another annotations or all annotations contained within the span of another annotation.

```
getContainingAnnotations( Annotation a1 )
getNextClosest( Annotation a1 )
getPreviousClosest( Annotation a1 )
getTextBetween( Annotation a1, Annotation a2 )
```

As part of a project to determine coreference in disease outbreak reports, the ability to determine relative position facilitated coreference resolution. It was also necessary to determine relationships between certain types of annotations from the window of the text. The Annotation Librarian simplified the task of determining co-location by providing the functionality within a single method call. Text between two Annotation objects was similarly identified with a single method call.

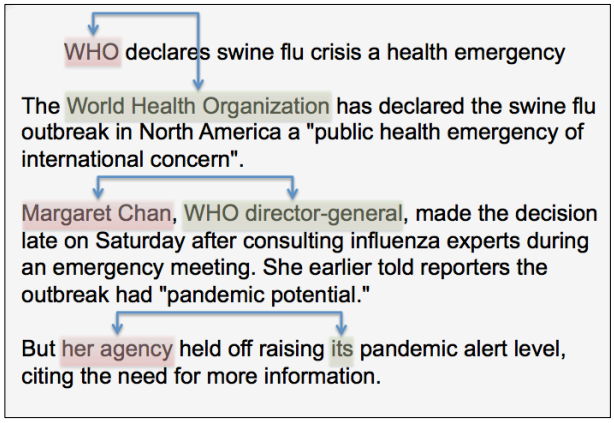


Figure 6: Disease Outbreak Reports Use Case

8 Conclusion

The Annotation Librarian was developed and modified over a number of different NLP use cases. Because of the diversity of tasks in each of these use cases, the toolkit includes functionality common to various types of NLP system development. It includes over two-dozen functions that were used more than one hundred times in each of the four systems listed above. Use of this interface reduced the amount of repeated code; it simplified common tasks, and provided an intuitive interface for NLP-centric annotation management without requiring the presence of an NLP developer who has intimate knowledge of the UIMA data structure. The extended capability provided by the pattern matching methods allows system developers to capitalize on the pipeline approach to NLP development in determining patterns. The ability to use annotations along with text significantly increases the types of patterns that can be identified without complex regular expressions.

9 Future Plans

The Annotation Librarian has been enhanced over the course of a number of biomedical NLP use cases and we plan to continue to enhance the interface as new use cases arise. Some planned enhancements include performance improvements and expanding the AnnotationPattern input pattern syntax to include regular expressions for method return values and annotation class names. We plan to provide additional functionality such as pattern frequency counts.

We see the ability for the Annotation Librarian to help identify patterns through active learning or

unsupervised techniques. In this way, relationships between annotations could be inferred based on those existing in the document set. Such functionality would also provide the ability for more intelligent analysis of future document sets or observation systems by allowing previously identified relationships to be utilized in other use cases.

Acknowledgments

This work was supported using resources and facilities at the VA Salt Lake City Health Care System with funding support from the VA Informatics and Computing Infrastructure (VINCI), VA HSR HIR 08-204 and the Consortium for Healthcare Informatics Research (CHIR), VA HSR HIR 08-374. Views expressed are those of the authors and not necessarily those of the Department of Veterans Affairs.

References

Annin Coden, Guergana K. Savova, Igor L. Sominsky, Michael A. Tanenblatt, James J. Masanz, Karin Schuler, James W. Cooper, Wei Guan, Piet C. de Groen. 2009. Automatically extracting cancer disease characteristics from pathology reports into a Disease Knowledge Representation Model. J Biomed Inform. 2009 Oct;42(5):937-49.

Christopher M. Frenz. 2007. Deafness mutation mining using regular expression based pattern matching. BMC Med Inform Decis Mak. 2007 Oct 25;7:32.

Cyril Grouin, Louise Deléger, and Pierre Zweigenbaum. 2009. COKAINE, A Simple Rule-based Medication Extraction System. i2b2 Workshop in conjunction with the AMIA Annual Symposium, San Francisco, CA; November 13, 2009.

David Ferrucci and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. Natural Langage Engineering 10(3–4): 327–348.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. AI Magazine. Vol 31. No 3.

Guergana K. Savova, Karin Kipper-Schuler, James D. Buntrock, Christopher G. Chute. 2008. UIMA-based clinical information extraction system. LREC 2008: Towards enhanced interoperability for large HLT systems: UIMA for NLP.

Jennifer H. Garvin, Brett R. South, Dan Bolton, Shuying Shen, Scott L. DuVall, Bruce Bray, Paul Heidenreich, Matthew H. Samore, and Mary K. Goldstein. 2011. Automated Extraction of Ejection Fraction (EF) for Heart Failure (HF) from VA Echocardiogram Reports. Department of Veterans Affairs Health Services Research and Development National Meeting. 2011 Feb 16.

John McCrae, Nigel Collier. 2008. Synonym set extraction from the biomedical literature by lexical pattern discovery. BMC Bioinformatics. 2008 Mar 24;9:159.

Leonard W. D'Avolio, Thien M. Nguyen, Wildon R. Farwell, Yong Chen, Felicia Fitzmeyer, Owen M. Harris, Louis D. Fiore. 2010. Evaluation of a generalizable approach to clinical information retrieval using the automated retrieval console (ARC). J Am Med Inform Assoc. 2010 Jul-Aug;17(4):375-82.

Leonidas Fegaras. 2005. Converting a Regular Expression into a Deterministic Finite Automaton. http://lambda.uta.edu/cse5317/notes/node9.html. Pulled February 2011.

Saiful Akbar, Thomas Brox Røst, Laura Slaughter, and Øystein Nytrø. 2009. Extracting Medication Information from Patient Discharge Summaries. i2b2 Workshop in conjunction with the AMIA Annual Symposium, San Francisco, CA; November 13, 2009.

Thierry Hamon and Natalia Grabar. 2009 . Concurrent linguistic annotations for identifying medication names and the related information in discharge summaries. i2b2 Workshop in conjunction with the AMIA Annual Symposium, San Francisco, CA; November 13, 2009.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. Chapman WW, Bridewell W, Hanbury P, Cooper GF, Buchanan BG. J Biomed Inform. 2001 Oct;34(5):301-10.

Zuofeng Li, Yonggang Cao, Lamont Antieau, Shashank Agarwal, Qing Zhang, and Hong Yu. 2009. Extracting Medication Information from Patient Discharge Summaries. i2b2 Workshop in conjunction with the AMIA Annual Symposium, San Francisco, CA; November 13, 2009.

Author Index

Abu-Jbara, Amjad, 121
Agarwal, Nitin, 115
Alabau, Vicent, 68
AR, Balamurali, 127

Bandyopadhyay, Sivaji, 50
Bär, Daniel, 74
Bechet, Frederic, 86
Benotti, Luciana, 62
Berman, Alexander, 92
Bhattacharyya, Pushpak, 127

Casacuberta, Francisco, 68
Ceylan, Hakan, 103
Chang, Jason S., 26
Chen, Bo-Nian, 133
Chen, Mei-hua, 26
Chiticariu, Laura, 109

Das, Amitava, 50
Denis, Alexandre, 62
DuVall, Scott, 139

Engkoo Team, Microsoft, 44
Erbs, Nicolai, 74
Erdogmus, Deniz, 38

Favre, Benoit, 86
Ferschke, Oliver, 97
Fried-Oken, Melanie, 38

García-Varea, Ismael, 68
Garner, Philip N., 80
Ginter, Thomas, 139
Goyal, Vishal, 1
Gurevych, Iryna, 74, 97
GVR, Kiran, 115

Hao, Yanfen, 14
Hild, Kenneth, 38
Huang, Chung-chi, 26
Huang, Shih-ting, 26

Joshi, Aditya, 127

Kiefer, Bernd, 7

Larsson, Staffan, 92
Le Roux, Joseph, 86
Leiva, Luis A., 68
Li, Cheng-Te, 32
Li, Yunyao, 109
Lin, Shou-De, 32, 133
Liu, Xiaohua, 44

Mihalcea, Rada, 103
Mohanty, Rajat, 127

Nanchen, Alexandre, 80
Nasr, Alexis, 86
Neumann, Günter, 20
Nezamfar, Hooman, 38

Oken, Barry, 38
Orhan, Umut, 38
Ortiz-Martínez, Daniel, 68

Popescu-Belis, Andrei, 80
Purwar, Shalini, 38

Radev, Dragomir, 121
Reddy, Ravi Shankar, 115
Reiss, Frederick, 109
Rey, Jean-Francois, 86
Roark, Brian, 38
Rosé, Carolyn Penstein, 115

Schäfer, Ulrich, 7
Schmeier, Sven, 20
Scott, Matthew R., 44
Singh Lehal, Gurpreet, 1

Soundrarajan, Balaji, 139
Spurk, Christian, 7
Steffen, Jörg, 7
Stymne, Sara, 56

Tseng, Chien-Lin, 32

Veale, Tony, 14
Villing, Jessica, 92

Wang, Chien-Yuan, 32
Wang, Rui, 7
Wang, Yen-Kai, 133
Weng, Jui-Yu, 133

Yang, Cheng-Lun, 133
Yazdani, Majid, 80

Zesch, Torsten, 74, 97
Zhou, Ming, 44