Hierarchical classification for Multilingual Language Identification and Named Entity Recognition

Saatvik Shah
CSE, MNIT, Jaipur
Rajasthan-302017
saatvikshah1994@gmail.com

Vaibhav Jain
CSE, MNIT, Jaipur
Rajasthan-302017
vj.25071996@gmail.com

Sarthak Jain
ECE, MNIT, Jaipur
Rajasthan-302017
sarthakssj@gmail.com

Anshul Mittal
EE, MNIT, Jaipur
Rajasthan-302017
anshulmittal712@gmail.com

Jatin Verma
EE, MNIT, Jaipur
Rajasthan-302017
jatin.verma.205@gmail.com

Shubham Tripathi
EE, MNIT, Jaipur
Rajasthan-302017
stripathi1770@gmail.com

Rajesh Kumar
EE, MNIT, Jaipur
Rajasthan-302017
rkumar.ee@gmail.com

ABSTRACT

This paper describes the approach for Subtask-1 of the FIRE-2015 Shared Task on Mixed Script Information Retrieval. The subtask involved multilingual language identification (including mixed words and anomalous foreign words), named entity recognition (NER) and subclassification. The proposed methodology starts with cleaning the data and then extracting structural and contextual features from the text for further processing. A subset of these features is selected (based on validation) for training supervised classifiers, separately for language identification and NER. Finally, they are applied hierarchically to annotate the entire text. The detected named entities are further subclassified by a novel unsupervised technique based on query refinement and keyword based scoring. The proposed approach on the testing dataset of the shared task showed promising results with a weighed F-measure of 0.8082. However, it is worth noting that the classifiers have been sub-optimal with respect to discriminating between certain linguistically similar languages (for e.g., Gujarati in Hindi and Gujarati pairs). The proposed approach is flexible and robust enough to handle additional languages for identification as well as anomalous foreign or extraneous words. The implementation of the approach has also been shared for the purpose of future research usage.

Keywords

Language Identification; NER;Information Retrieval; Wikipedia; Query Refinement; Ensembling;

1. INTRODUCTION

The ubiquitous use of indigenous languages in social media and search engines has led to the increasing need for techniques to tackle mixed script information retrieval. In such an environment, language and named entity identification has become both a necessary and challenging task. At this juncture, the shared task for mixed script information retrieval for the FIRE workshop serves as an excellent source to obtain multiple insights to the solution to such a problem. This research paper addresses language identification (LI) and Named Entity Recognition (NER) for text in social media. Such kind of text could involve, for example, switching of language in between a sentence (code switching), or even include words of mixed languages (code mixing). In addition to this, people also use phonetic typing and irregular spellings in social media ('2nyt' in place of 'tonight' or 'y' in place of 'why'). All these challenges make the correct identification of language in data from social media a tough task.

The task required rigorous preprocessing of both the training (validation) as well as testing dataset. Section 2 describes, in detail, the approach followed for it. Rest of the paper is organized as follows. Section 3 describes all the features extracted. Section 4,5 and 6 describe the hierarchical classification approach that we implemented for tagging of tokens into their respective categories. Section 7 and 8 cover the Results and Conclusion. A Web Tool has been developed for the implemented approach and can be accessed at **https://mixscian.herokuapp.com**. The source code has been shared on the link, **https://github.com/saatvikshah1994/hline** for future research purposes.

2. PREPROCESSING

During training, the following steps were performed to achieve preprocessing:

1. Began preparing the data by skipping all the X's present in the Input file, which are punctuation, numerals, emoticons, mentions (words starting with @), hashtags (words starting with #) and acronyms; with respect to the given annotation file directly.
2. Then Unicode based cleanup of words was performed, to bring Unicode characters to their closest matching ASCII representatives. (Eg. **"Zine$^{\mathbf{TM}}$"** to **ZineTM**)
3. After that, each utterance was formatted to remove any punctuation marks in between letters.
4. Finally, erroneous utterances (with unequal number of words and labels) were discarded.

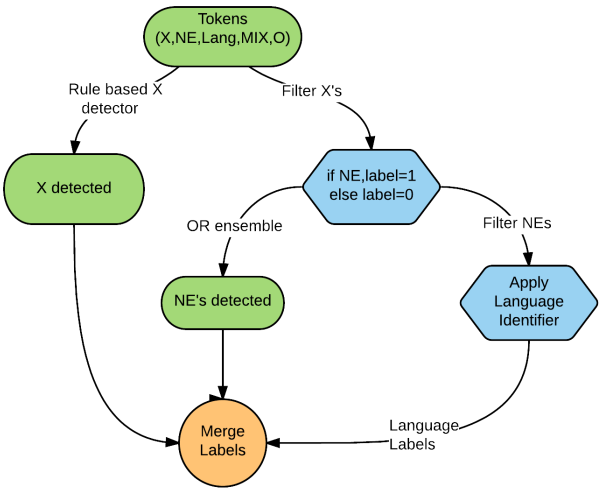

Figure 1: Hierarchical Classification Workflow

During testing, the process started with performing Unicode based cleanup, followed by removal of punctuations or any other invalids in between letters from tokens.

3. FEATURE EXTRACTION

The proposed technique uses a comprehensive set of features [1] for both language identification and NER. These are explained in brief below:

1. **Word Context:** Word Context features that can be extracted from a token (word) being labeled and its surrounding context. Considering w_0 as current token, we took Subset of all features for upto 2 context words, both in the left and right (w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}).
2. **Prefix and Suffix:** Prefix and suffix, defined as variable length character sequences (here, 3 and 2), are stripped from each token and used as the features of classifier.
3. **Stemming/Lemmatization:** A step applying stemming or lemmatization of input tokens or both before extracting features was added to the feature extraction pipeline and tested.
4. **Character level n-gram:** Character n-gram is a contiguous sequence of n character extracted from a given word. We extract character n-grams of length two (bigram) and three (trigram), quad(four), penta(five) and use these as features.
5. **POS tags:** POS tagging(or grammatical tagging), marks part of speech of a word present in corpus based on both definition and context.
6. **Relative Position:**The relative position of specific token in its utterance was considered and added as a feature.
7. **Word Normalization:** Word Normalization is done to capture similarity between two words with common properties like "HahaHHa123" is normalized to "AaaaAAa000", with "A" denoting a capital, "a" denoting non capital and "0" denoting a numerical character.
8. **Compressed Word Normalization:** The output of Word Normalization is compressed to remove repetitions of similar characters appearing together. For example "AaaaAAa000" becomes "AaAa0".
9. **Composition features:**A feature was defined whose value depended upon the presswork of the word. The following four values were possible:
   - AllCaps (if the current word is made up of all capitalized letters),
   - AllSmall (word is constructed with only lowercase characters),
   - WordDigit (word contains digits and alphabets both)
   - InitCaps (whether the first character is in uppercase) and

These features, having variable length for every token under consideration, had to be further processed into a Bag-Of-Words formation. Term Frequency Matrix was computed for the same, resulting in the formation of sparse matrices, which were finally passed onto the classifiers.

4. PUNCTUATION (X) RECOGNITION

This section describes the methodology used for tagging Xs, or tags which are punctuation marks, numerals, mentions, hashtags, acronyms or emoticons. The process begin by a rule based approach for isolating the hashtags, mentions, and numerals. This is done by checking for tokens that start with '**@**', or '**#**', or tokens that are composed entirely of digits. Then, the tokens are checked for web URLs or email addresses, and punctuations. A regular expression based classification was implemented to search for tokens starting with '**http://**' or '**https://**', or any token of the format '**abc@pqr.xyz**'. Also, any tokens that are composed only of punctuation marks are isolated and tagged as X. Another regular expression check is applied to catch any emoticon such as '**:-)**', '**;)**', '**:}**', etc. A dictionary based approach is then applied to check for any token to exist in a list of popular acronyms. If there is a match, the corresponding token is also classified as an X. Once these 'X's have been filtered out, a cleaning operation was performed on the remaining tokens. Cleaning is defined as converting Unicode characters to the nearest matching ASCII characters. For example, converting **"Zine™"** to **ZineTM**. After this preprocessing, these 'cleaned' tokens are sent for Named Entity Recognition.

5. NAMED ENTITY RECOGNITION

This task focuses on tagging those tokens as named entity which can be names of persons, locations, organizations etc [2] in the given text. For each named-entity, we can:

1. Classify it as **{NE}**, OR
2. Sub-classify it as **{NE_L, NE_P, NE_O, NE_PA, NE_LA, NE_OA, NE_Ls, NE_X, NE_XA}**

Methodology: Tagging of named entities has been divided into two steps:(1)Supervised approach for binary classification and (2)Unsupervised approach for sub-classification.

5.1 Supervised Approach

The Supervised approach includes training of Classifiers [3], namely, Linear-kernel SVM(using liblinear), Logistic Regression and Random Forest Model on the extracted features explained in section 3. The input data for training included this year's training data along with dataset of FIRE 2014. All these were implemented by use of Transform Pipelines containing Clean_Transform(to clean data) and Feature_Extractor. Finally, the use of Term Frequency Matrix was done to obtain numeric features. Based on individual feature cross-validation, the most important features that contributed in identifying the Named Entities were:

- Word(and Compressed) Normalization
- Local Context(with a smaller window)
- Relative Position
- Prefix/Suffix
- POS Tags.
- Composition Features.

An Ensemble Model of NER was developed which performed a "LOGIC OR" on the predicted results of each classifier to decide the final tag for a particular token. One(1) was used to denote a token being a named entity and Zero(0) for non-named entity.

$$Tag_{\text{out}} = 0 \ OR \ (Tag_{\text{classifier\_1}}) \ OR \ (Tag_{\text{classifier\_2}}).... \ OR \ (Tag_{\text{classifier\_n}}) \quad (1)$$

where
Tag_{out} is the predicted result for each token,
$Tag_{classifier\_k}$ is the predicted tag by classifier k, and
n is number of classifiers used.

5.2 Unsupervised Approach

An Unsupervised approach based on parsing Wikipedia to find out whether the predicted named entity is a person, location or organization was employed [6]. Searching the Wikipedia page of a named entity provides information comprising of the title of the Wikipedia Page, the summary, the section headings, the infobox contents, the categories section, etc. indicating whether it is a person, location or organization. The prediction is done using three reference sets each containing some keywords that are commonly used in the context of a person, location or organization. For a person, keywords can be *'born', 'Education', 'President', etc.*; for a location, they can be *'place', 'nation', etc.*; for an organization, they can be *'party', 'company', etc.* While parsing the Wikipedia page of *'Narendra Modi'*, we will surely get across the word *'President'* declaring *'Narendra Modi'* as **'NE_P'** or person. However, it was found that for some named entities like AAP, Mark, etc., it displays a disambiguation page whereas some named entities like *Sonia, Pradep*, etc. didn't even have a Wikipedia page. Thus, a systematic approach overcoming all the shortcomings of only Wikipedia parsing was formulated which is described in the following paragraph.

Assuming the named entity to be of one word, let the named entity that is to be sub-classified be denoted by a variable *token*. The following procedure prevails:-

1. First of all, any of the punctuation marks, numerals or special characters are removed from *token*.
2. The *token* is searched using Wikipedia API for python that returns an object containing the information about token extracted from its Wikipedia page.
3. Using the object, all the above-mentioned information is concatenated to form a single string denoted by *wiki-extract*. For convenience, the information is first converted into lowercase, and any of the punctuation marks, numerals, Unicode characters, etc. are removed. Each of the words in *wiki-extract* is lemmatized, and to counter the internet dependence and decrease the processing time, the final *wiki-extract* is cached in a file with the corresponding token against it.
4. The reference set keywords are searched in *wiki-extract*. The number of times a keyword is spotted in *wiki-extract* will be the score of the subclass **('NE_P', 'NE_L' or 'NE_O')** to which the keyword belongs.
5. After each of the keywords' search is completed, we get a final score for each subclass. If the difference between the scores of the subclass with the maximum score and that with the second maximum score is greater than or equal to a confidence threshold (5 in this paper), then *token* is annotated as the subclass with the maximum score, otherwise it is annotated as 'en'. If the final scores of all the subclasses is zero, it is marked as 'en'.

If in the 2nd step, Wikipedia shows disambiguation error, then *token* is searched on Bing using its API for python named 'py_bing_search'. As long as the data under process belongs to social networking sites including popular topics like politics, movies, etc., the results by Bing will help in finding out the most popular/probable meaning of *token* out of many ambiguous cases. Steps to be followed are:-

i Using 'py_bing_search', *token* appended with the word 'Wikipedia' (for e.g. 'AAP wikipedia') is searched on Bing so that it returns URLs of Wikipedia pages at top priority. Only first three URLs are considered.

ii Only those URLs that point towards a Wikipedia page except disambiguation pages are considered.

iii As soon as the desired URL is obtained, the proper term that would not cause disambiguation error and could replace *token* is extracted out of that URL. For e.g. if the token was *'AAP'*. Bing will return a URL 'https://en.wikipedia.org/wiki/Aam_Aadmi_Party' for this token and then the term *'Aam_Aadmi_Party'* extracted out from this URL will replace *token*.

iv Now, rest of the steps of searching the new token on Wikipedia are same as step 2 to step 5 discussed above.

If, among the three URLs, none of them satisfies the condition of being a Wikipedia page except disambiguation pages, then under disambiguation error there is an option to fetch the options offered by Wikipedia disambiguation page. Only first five options are considered in this paper. These five options are concatenated to form *wiki-extract* and rest of the procedure is same as step 3 to step 5 keeping the confidence threshold zero.

If in the 2nd step, Wikipedia shows page error, then to replace *token* with a proper term that would not cause page error, Bing search is used again. Hence, the steps to be followed are same as step i to iv. Again, if among the three URLs obtained from 'py_bing_search', none of them satisfies the condition of being a Wikipedia page except disambiguation pages, then the token is annotated as 'en'.

Apart from the above algorithm, it is constantly sought that if the term succeeding *token* also comes out to be named

P, R, F-S	Bn	En	Gu	Hi	Kn	Ml	Mr	Ta	Te	X	NE	NE_L	NE_P
P	0.878	0.958	0.097	0.817	0.575	0.394	0.705	0.937	0.431	0.961	0.368	0.722	0.2121
R	0.966	0.848	0.5	0.74	0.829	0.752	0.79	0.708	0.687	0.966	0.528	0.124	0.25
F-S	0.838	0.9	0.163	0.776	0.679	0.517	0.745	0.806	0.529	0.964	0.433	0.214	0.229

Table 1: Strict Precision (P), Recall (R), F-scores (F-S) for languages and NEs.

entity, provided that token is not positioned last in the current utterance, then they are treated as a single entity and then searched on Wikipedia. For e.g. the tokens are 'White' 'House' which are marked as named entity. If we search 'White' on Wikipedia we won't get desired results. But if we search 'White House', we may surely get it as a location, and thus both 'White' and 'House' are annotated as **'NE_L'**. If the tokens are misspelled or ambiguous, then Wikipedia may cause page or disambiguation error. In that case, the combined token is first searched on Bing to find out the term that could replace the combined token. If this results in failure in finding out a Wikipedia page except disambiguation pages, then the process is reverted and the two tokens are searched individually again following the procedure discussed earlier.

6. LANGUAGE IDENTIFICATION

This section addresses the problem of identification of language of origin of a query term in the code-mixed query [4]. For Language Identification, linear-kernel SVM(using liblinear) classifier was used with important features being:

- Word n-grams(bi-grams, tri-grams, quad-grams and penta-grams)
- Local Knowledge
- Part of Speech Tags
- Composition Features

Parameter optimization was performed using parameter tuning based on cross-validation. Supervised approach[5] was implemented with the classifier being trained on current year's training dataset and previous year's dataset. A similar transform pipeline as used for NER was implemented for the training of classifiers. A multilevel classification was performed with possible tags for each token:

1. One of Languages in L={(**en**), (**hi**), (**bn**), (**gu**), (**ka**), (**ml**), (**mr**), (**ta**), (**te**)}
2. Token made of two languages **{MIX}** and its subcategory **{MIX_L_r_L_s}** where r and s are the language of root and suffix respectively
3. None of the above **{O}**

For the purpose of training, the datasets were cleaned and tokens annotated as NE or its subcategories and X's were removed. The trained classifier was then applied on testing set to get the prediction for each token.

7. RESULTS

In this section, results are reported to proposed experiments. To investigate and find the most effective methods as described in upper sections, results are initially obtained by development(Training) set and finally, the model was selected using 4-fold cross validation. Thereafter, a run was submitted for Subtask-1. The overall performance was measured in terms of Utterance, Token and NE accuracies along with precision, recall and F-measure scores for individual languages and sub-divided NE. The **Average F-measure** and **Weigthed F-measure** scores obtained was **0.654** and **0.808** respectively. While generally the Mean **Average F-measure**, **Weigthed F-measure** score was **0.5395**, **0.6990** and Max **Average F-measure**, **Weigthed F-measure** score was **0.6917**, **0.8299**.

8. CONCLUSION

In this paper, a system was described for Subtask-1 i.e. Query Word Labelling in FIRE Shared Task 2015 on Mixed Script Information Retrieval. The validation procedure indicated that the context of a token, its POS tag, its structure and normalization play key roles in both NER and Language identification. It is interesting to note that the performance on linguistically dissimilar languages such as English and Bengali has been top notch. On the other hand, performance on similar language pairs such as Hindi and Marathi have invariably lead to the classifier getting confused. The use of Search engines to refine query terms followed by use of publically available encyclopedias such as Wikipedia for subclassifying named entities is a newly proposed technique that has given promising results.

9. REFERENCES

[1] Gupta, D. K., Kumar, S., & Ekbal, A. Machine Learning Approach for Language Identification & Transliteration: Shared Task Report of IITP-TS.
[2] Abhinaya N., Neethu John, Dr. M. Anand Kumar, Dr. K.P. Soman (2014). Amrita @ FIRE-2014: Named Entity Recognition for Indian Languages
[3] Dubey, S., Goel, B., Prabhakar, D. K., & Pal, S. ISM@ FIRE-2014: Named Entity Recognition Indian Languages.
[4] King, B., & Abney, S. P. (2013, June). Labeling the Languages of Words in Mixed-Language Documents using Weakly Supervised Methods. In HLT-NAACL (pp. 1110-1119).
[5] A. Das and B. Gamback. (2014). Code-Mixing in Social Media Text:The Last Language Identification Frontier? Traitement Automatique des Langues (TAL): Special Issue on Social Networks and NLP , TAL Volume 54 no 3/2013, Pages 41-64
[6] Nothman, J., Ringland, N., Radford, W., Murphy, T., & Curran, J. R. (2013). Learning multilingual named entity recognition from Wikipedia. Artificial Intelligence, 194, 151-175.