Hindi-English Language Identification, Named Entity Recognition and Back Transliteration: Shared Task
System Description

Navneet Sinha¹, Gowri Srinivasa²
Dept. of Information Science and Engineering, PES Center for Pattern Recognition,
PESIT Bangalore South Campus, Bengaluru.
¹navneet.sinha27@gmail.com, ²gsrinivasa@pes.edu

Abstract

This paper presents an algorithm for word level language identification, named entity recognition and classification, and transliteration of Indian language words written in the Roman script to their native Devanagari script from bilingual textual data. We propose the construction of an extensive, hierarchical structured dictionary and hierarchical rule-based classifier to expedite word search and language identification. The proposed method uses lexical, contextual and special character features particular to Hindi and English. With a few modifications to the system, the present solution can be replicated for other languages. The system we have submitted shows the best performance in English token level precision (0.895) and the second best in Indian language token recall (0.915). The transliteration level f-measure is relatively low (0.15); this can be significantly improved with a more representative and exhaustive training data.

1. Introduction

Hindi is one of the official languages of the Federal Government of India and the fourth most commonly used language in the world; the first three being Chinese, English and Spanish. It has been widely observed that a large number of Indians tend to communicate using an admixture of words from English and Hindi and/or other native languages. Most of these native languages have their own script, including the Devanagiri script for Hindi. Yet, due to various socio-cultural and technological reasons, most writers of dual or multiple tongues use the Roman script transliteration of words in the native language in blogs, tweets, posts, etc. Transliteration between the Devanagari script and Roman script is difficult due to difference in the encoding of the scripts, number of alphabets, capitalization of leading characters, phonetic characteristics, character length and various modifications. In this paper, we propose to parse natural language sentences written in Hindi and English using the Roman script from sources such as blogs, tweets, posts, etc. and, use multiple hierarchical pre-processed dictionaries and a match based classifier to classify the language and subsequently transliterate words in Hindi to Devanagari.

The paper is organized as follows – in section 2, we describe the dataset used for training and designing the system. In section 3, we present our approach and experimental setup, the experimental results in section 4 and an error analysis in section 5. Finally, we conclude with a few possible directions for further work in Section 6.
2. Dataset

To build the system, we used the English word list and the annotated output together with the file of Hindi word and transliteration pairs from the FIRE 2013 dataset. We also used several lists off the internet for the names of location and other named entities to build various dictionaries to expedite the classification of each word [2-4].

3. Approach

**Problem:** The task involves reading an input file of bilingual text and tagging each word with a label: Hindi, English, Named Entity (proper nouns such as names of people, organizations, etc.), Location (places such as cities, countries, etc.), Abbreviation or Other (numerals, emoticons or smileys, etc.). And, for those words labelled “Hindi”, it is required to transliterate the word written in Roman script to Devanagiri script.

**The proposed solution:** The foundation of our approach is based on the formation of dictionaries for words in each category, except the “Other” category, which is determined based on evaluating regular expressions. The dictionaries formed are used for language tagging, back transliteration as well as for naming entities.

For every input token, rather than search exhaustively in the various dictionaries, we define a flag to keep track of the last accessed language dictionary: Hindi or English. This determines the priority with which the Hindi dictionary will be searched for the next input token. The other labels, such as “Abbreviation”, Location”, etc. do not alter the language label. The assumption is that the probability of finding a Hindi word succeeding another word in Hindi is higher than an English word succeeding a word in Hindi. Likewise, the probability of finding an English word succeeding another word in English would be higher than a Hindi word succeeding a word in English. Proper nouns such as names of people or places, numbers, smileys and abbreviations are generally language independent and hence do not contribute to altering the language flag.

The steps of the approach are as follows: a file with bilingual text is input and parsed by the system. For an input token (word) from the bilingual text file, the system first evaluates various regular expressions to determine if it belongs to the “Other” category. If the token does not belong to the “Other” category, the system proceeds to search for the word based on the language flag. Suppose the language flag is set to “Hindi”, the token that is read is first matched with the Hindi dictionary. If it is found, the system checks the sub-dictionary for the Devanagiri transliteration of the word. If the transliteration is found, the category is output as “H” for Hindi and the transliteration is also noted. If the transliteration is not found, only the label is output. In case the word is not found in the Hindi dictionary, the system proceeds to check for the word in the “Named Entity”, “Location” and “Abbreviation” dictionaries in turns. If they the token is found in any of these categories, the annotation is output. If not, a word is tagged as “E” for English by default. This accounts for anomalous entries and typographical errors in the input text. If the token is tagged as “E”, the flag is reset to “English” and the search order of the dictionaries in the next iteration would the “Other” category followed by the English dictionary and various named entities before searching through the Hindi dictionary.
3.1 Dictionary formation

The dictionaries are based on lists of words in each of the categories. To create the Hindi dictionary, first a python script is run on the various “every transliteration pairs” (populated from the FIRE 2013 data archive) in accordance with their structure of representation (Hindi word in Roman script followed by the Devanagiri transliteration). The script scraps the required text from all these files as a prerequisite for the Hindi annotation and transliteration dictionaries. The English dictionary is built from word lists available online, although these have presently not been used to search for a word as any input not found in the other categories is labelled as “E” for English by default.

The dictionary follows a tree-like hierarchical structure. It was constructed so as to reduce the number of comparisons over a linear search through all the possible dictionaries. The lists of words are sorted by their alphabetical order (Roman script) and a sub-dictionary is created for every alphabet. In the case of Hindi words, a sub-dictionary stores the Hindi words written in Roman script with their Devanagari equivalent; this is also alphabetically sorted (based on the Roman script).

For example, a Hindi sub-dictionary for the alphabet “b” would contain the entry:

```[
“b”, (“bharat”:“भारत”)[…]
]```

The ellipses in the above example indicate other words with their transliteration pair starting with the alphabet ‘b’.

The lists for English words, Hindi words and named entities form separate dictionaries of the ‘defaultdict’ structure which is already provided by the collections package in python. An entry in the English dictionary for the alphabet a might be as follows:

```[
“a”,(abase, advance, attest,…)]
```

These styles of dictionaries help in the quicker processing of the output file as they drastically reduce the number of comparisons.

3.2 Regular Expressions, Tagging, Back Transliteration, Named Entity Representations

A python script loads the different dictionaries and, if the straight forward alphabet search does not yield a class label, the script proceeds with the evaluation of regular expressions.

First, the system uses manually written regular expressions for checking whether the input word belongs to the “Other” category which includes numbers, symbols, smileys, web page links and web page references. The expressions also screen for shortened words (such as “they’re”), hyphenated words, etc. These are annotated as “English”. A list of the regular expressions is presented in Table 1.

Next, the system checks for the presence of an input word in the Hindi dictionary through nested `if-elif` ladder. If the input word is found in the Hindi dictionary, then the system proceeds to check whether the token’s Devanagari equivalent is available. If the pair is found, the label and the transliteration are printed in the output. If not, the word is labelled but not transliterated. In case the word is not found in the Hindi dictionary, the systems searches in
the “Named Entities”, “Location” and “Abbreviation” dictionaries. Then the word is checked whether if it’s an abbreviation or not. If an input word is not found in any other category, it is assumed to be an English word and suitably labelled.

Table 1: Regular expressions to screen for the “Other” category and compound words

<table>
<thead>
<tr>
<th>Expression</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>r'[a-zA-Z0-9]+]</td>
<td>Others</td>
</tr>
<tr>
<td>r'[a-zA-Z]+[-]+[a-zA-Z]*'</td>
<td>English</td>
</tr>
<tr>
<td>r'http+'</td>
<td>Others</td>
</tr>
<tr>
<td>r'[a-zA-Z]+-[a-zA-Z]*'</td>
<td>Others</td>
</tr>
<tr>
<td>r'[A-Za-z0-9]+.com'</td>
<td>Others</td>
</tr>
</tbody>
</table>

Disambiguation of words common to both languages: An advantage of using a language flag is the ability to delineate bilingual words as belonging to English or Hindi based on the context of the previous word. The underlying assumption is that if the previous word was found in the Hindi dictionary, the current word is more likely to be a word in Hindi. The language flag is used at the start of the conditional ladder and thus, effectively handles the evaluation of words which could be used in both languages such as “to”, “in”, “me”, “so”, “us”, “par” and others.

The system is not sensitive to the case of the input with regard to searching through the language dictionaries. Thus, bilingual words in either case are handled in the same way. However, some of the regular expressions evaluated for the “Abbreviation”, etc. categories are sensitive to case and annotate the input appropriately.

4. Results

A hierarchical rule-based search driven by the language flag yields promising results for the tagging precision, recall and f-measure for both languages and for named entity tagging. Results for transliteration are relatively low due to a paucity of training data used to populate the dictionary as well as errors in the test data. The results are summarized in Table 2.

Table 2: Results: various metrics computed by MSRI team:

<table>
<thead>
<tr>
<th>LP</th>
<th>LR</th>
<th>LF</th>
<th>EP</th>
<th>ER</th>
<th>EF</th>
<th>TP</th>
<th>TR</th>
<th>TF</th>
<th>LA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.853</td>
<td>0.915</td>
<td>0.883</td>
<td>0.895</td>
<td>0.822</td>
<td>0.857</td>
<td>0.091</td>
<td>0.427</td>
<td>0.15</td>
<td>0.855</td>
</tr>
</tbody>
</table>

EQMF All (NT) EQMF w/o NE (NT) EQMF w/o Mix (NT) EQMF w/o Mix and NE(NT) ETPM

0.231 0.354 0.231 0.354 1086/2235

5. Error Analysis

The system yields promising results for the tasks of classifying words based on language and recognizing named entities. However, the results of the transliteration are relatively low. We conjecture this is based on the creation of the dictionary: the transliteration pairs were directly scrapped from the files in the dataset and were not checked for errors. Thus, the erroneous data output by the system for the test data did not match with the ground truth that was used to evaluate the performance of the system.

A few examples of such errors would be:

\[
\begin{align*}
\text{baat/H=} & \text{भारत} \\
\text{ab/H=} & \text{आब} \\
\text{mujhse/H=} & \text{मुझे} \\
\text{kiya/H=} & \text{क्या} \\
\end{align*}
\]

Further, for a majority of the words in the test data, Devanagiri equivalents were not available in the dictionary (due to the training data not being exhaustive). Since there was no other way for the proposed system to transliterate Hindi words in the Roman script to Devanagari script, the transliteration accuracy has been low.

6. Conclusion and Future Work

In this paper we presented a brief overview of a hierarchical rule-based classifier for Hindi-English language identification of words, back transliteration of Hindi words to the Devanagari script and named entity tagging for people, location, organization and abbreviations. The experimental results demonstrate that having sound dictionaries with an efficient architecture and suitable regular expressions and search strategies yield promising results for the tagging. In the future, we would like to improve the performance of the annotation on unknown words with the help of the addition of n-grams, dictionaries that are dynamically updated and other search strategies (such as sub-dictionaries for a pair of letters). Furthermore, the transliteration task could be improved based on Unicode equivalents for the Roman script rather than rely on a pre-existing dictionary of transliterated words.

References


