Incremental N-gram Approach for Language Identification in Code-Switched Text

Prajwol Shrestha
Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Nepal
prajwol.shrestha18@gmail.com

Abstract

A multilingual person writing a sentence or a piece of text tends to switch between languages s/he is proficient in. This alteration between languages, commonly known as code-switching, presents us with the problem of determining the correct language of each word in the text. My method uses a variety of techniques based upon the observed differences in the formation of words in these languages. My system was able to obtain third position in both tweet and token level for the main test dataset as well as first position in the token level evaluation for the surprise dataset both consisting of Nepali-English code-switched texts.

1 Introduction

Nowadays, it is common for people to be able to speak in two or more languages. So, the propensity to use code-switching in spoken as well as in written text has increased. Code-switching occurs when a person uses two or more than two languages in a single piece of text. According to Elfardy and Diab (2012), the phenomenon where speakers switch between multiple languages between the same utterance or across utterances within the same conversation is referred to as Linguistic Code Switching. English, being an universal language is highly likely to be code-switched with some other language. This is specially true when English is studied or spoken in the community as the second language by a person. In a such case, the person is likely to use English words with his/her native language to form code-switched, yet, syntactically correct and meaningful sentences.

This paper deals with the code-switching that occurs when English is used with Spanish or Nepali. The problem of identifying code-switching is closely tied with figuring out how a language is acquired or learned. Auer (1988) identified the phenomenon of how Italians, who were raised in Germany developed fluctuation and variation in their native language as well as in German. They were also noticed to have a strong tendency to have a conversation dominated by the German words. This phenomenon was also observed by Dey and Fung (2014). The strong influence of Bollywood in the Indian culture and the high amount of code-switching with English in movie dialogues and song lyrics, led to Hindi-English code-switching, being common for the average Indian. Finding out the points in the text where people are most likely to code-switch, what word of a certain language is more likely to be used than a word with the same meaning of another language and which languages are more likely to be used in code-switching than others are all important research questions. Although my paper deals only with finding out the language a certain token in a code-switched text belongs to, this is a first step towards answering those other questions.

The main aim of this paper is to describe my system submission to the Computational Approaches to Code Switching task (Solorio et al., 2014). The training dataset provided for the classification task were tweets composed of Spanish and English words or Nepali and English words. The test dataset also consisted of similar tweets. In addition to this, there was also a surprise dataset consisting of Facebook posts and comments in the place of tweets. My system for this task performs language identification by using a number of techniques. The first one is based upon an assumption that words of different languages have varying sets of n-gram prefixes that occur predominantly throughout the language. There has been prior research on language identification through the use of n-grams. Cavnar et al. (1994) have approached the task of identifying the language of an electronic mail taken from Usenet newsgroups with the use of n-grams. They obtained training sets for each language to be classified, which acted as language category samples. They computed n-gram frequency profiles on these training sets. They found that the top 300 n-grams of each language are used most frequently to form the words of the language. Nguyen and Dogruoz (2014) have used dictionary search and a n-gram based language model to identify the language on word-level of forum posts with Dutch and Turkish code-switching.

Lignos and Marcus (2013) found that data collected from social media to detect code-switching contained a lot of non-standard spellings of words and unnecessary capitalization. It was also true for this dataset. So, I made use of a lightweight spell checker in the event that the word was not spelled correctly and hence not categorised into any language. I have also used a rule based classification system that can also be used for named entities and non-alphanumeric language classes. With the system that I built based on these ideas, I achieved an accuracy of above 94% for English-Nepali and above 80% for English-Spanish in the token level evaluation. As the system works as a pipeline of smaller systems, it was time consuming. So, in order to improve speed, it is built to run on a multithreaded environment.

Language identification by using these techniques overcomes the drawback of other simpler methods like extracting a token's characters and then using its Unicode value to determine its language. But most of the time the words are not written in its own script by using Unicode, but rather, its Romanized form is used. Some languages like Spanish are almost fully written in roman letters, with exception being only a small subset of accented characters. Precisely these kinds of words require more robust classification techniques. Another alternative is manual classification but it has the downside of being time consuming and an uneconomical alternative. There is a need of an application that can overcome these drawbacks and create a system that can be used for similar sets of data.

2 Methodology

The classification of a token of a code-switched text into one of the six classes: lang1, lang2, ambiguous, named entity, mixed and other is performed by using four techniques described shortly. But before applying any of these techniques, the first step was the creation of a dictionary for each class by using the tokens from the training set. As a preprocessing step, for any token that starts with #, the # is removed. Also, any token that starts with @ is given the 'other' class label. The techniques used in my system are detailed below. They are applied in a pipeline, in the same order as they are mentioned.

2.1 Incremental N-Gram Occurrence Model with Dictionary Search

This model is used for test tokens whose length (L) is greater than three in the case of Nepali-English code-switching task and is greater than two in the case of Spanish-English code-switching task. Tokens that are shorter are classified by using a simple dictionary lookup. If the occurrence count of the token in the dictionary of class C is the highest, then the token is classified as belonging to class C.

In order to assign a class label to a particular token, this model uses only the first ngram of each size n ranging from 3 (for Spanish-English) or 4 (for Nepali-English) to L-1. The count of occurrence of this ngram in each class dictionary is taken as the score. The size n is increased iteratively and the score from each iteration is added at the end to obtain the final score. For named entity (NE) and ambiguous dictionary search, the whole token is used instead of just the ngram since the size of these dictionaries is small. Since a whole token lookup was performed, the occurrence count scores from these dictionaries are rated to be three times higher. After obtaining the final scores for each class, the one with the highest score gets assigned as the class label of the token.

This method is based on the hypothesis that tokens belonging to the same language will have more overlap of the preceding characters. If two tokens are from different languages, they might start the same way but will start deviating in the use of characters faster than two tokens of the same language. The Incremental N-Gram Model for Nepali-English Classification is shown in Algorithm 1.

Consider that we have to find the language of the Test token Parsin. The following assumptions are made:

Algorithm 1 Incremental N-gram Classification

```
if $len(token) > 3$ then
    $n = 4$
    while $n < len(token) - 1$ do
        if $token \in dict$[ambiguous, ne] then
            Increment Respective Language
            Occurrence Count by 3
        end if
        if FirstN-Gram $\in$ Remaining Classes then
            Find the number of words in
            each class dictionary that starts
            with the First N-Gram.
            Add this number with the previous
            occurrence count for the
            particular class
        end if
    end while
end if
```

N-gram Size	First N-gram	English	Nepali	Ambiguous
4	PARS	2	6	3
5	PARSI	2	6	3
6	PARSIN	1	0	3
Total		7	12	9

Table 1: Incremental N-gram Classification Example

- The Word Parsing occurs twice and Parsimony once in the English Language Dictionary.
- Word Parsi occurs 6 times in the Nepalese Language Dictionary (Parsi means the day after Tomorrow).
- Test token Parsin occurs 0 times in Other Language and Named Entity Dictionary
- Test token Parsin occurs once each in Ambiguous words Dictionary

The algorithm works as shown in Table 1.

2.2 Rule Based Classification

A small fraction of test tokens are left unclassified by the above method. These tokens are further processed by using a rule based classification system. It consists of the following handwritten rules:

- Check if the token is an emoticon against an emoticon list. If the token is found in the list, it is of the class, 'other'.
- It was hard to find an off-the-shelf named entity recognizer for code-switched text. So, a simple named entity recognition rule was used. For a token consisting of only alphabetic characters, if there are more than one uppercase letters in the token or if the token starts with an uppercase letter, it is an NE.
- If the difference in the occurrence score of a token in lang1 dictionary vs lang2 dictionary is higher than three, the token is classified as belonging to the language with the higher score.
- If the token occurs in lang1 and lang2 dictionaries equally, the token is 'ambiguous'.

2.3 Lightweight Spell Checker

The test tokens that are still not classified are checked for spelling errors using a simple spelling checker, complementary to the idea of edit distance. If the above two classifiers were unable to classify a token, it might be because these tokens were misspelled. This method is based upon the idea that misspelled tokens are still similar to the language that they belong to. The spell checker checks the test token against every token in the dictionaries for similarity (defined below).

'Similarity' is defined as follows: First, a 'similar count' score (SC) is calculated as the number of characters that match between two tokens in order. A test token of length $L1$ is said to be similar to a dictionary token of length $L2$ if: SC>max($L1$,$L2$)-1 when $L1$<7 or SC>max($L1$,$L2$)-2 when $L1 \geq 7$

Here, when the test token is checked against a token in the Nepali dictionary, the characters 'x' and '6' in both tokens are replaced with the character sequence 'ch'. This normalization is performed because it is very common for the latter character sequence to be replaced by either of the former two characters, in the Nepali language. If a test token is found to be similar to a token in a dictionary of a certain class, the similarity score to the class is incremented. The class with the maximum similarity score is considered to be the class of the test token.

2.4 Special Characters Check

At this stage, only a minimal number of tokens are left to be labeled. These tokens are checked to see if they contain characters not belonging to English Unicode or modifiers. If one such character is found, the token is said to be from lang2, either Spanish or Nepalese. All the remaining tokens are categorized as 'other'.

3 Experimental Settings

For all my experiments, I divided the training data into a ratio of 70:30 for training and cross-validation. In order to tune the different parameters, I had to repeat the experiments multiple times. So, in order to improve the runtime performance, I made use of multithreading.

I tested the application by setting the first n-gram length in the Incremental N-Gram Model to 3 and 4. I varied the criteria of the least number of characters that should match between two tokens, in order for the two tokens to be similar. I observed the highest accuracy of above 94% in Nepali- English classification when the First n-gram length was 4. In the case of Spanish-English token classification, I observed the highest accuracy of 88% when the n-gram length was 3. The spellchecker gave the best results when it had the above mentioned similarity criteria.

The whole classifying task was sure to take a long time so I built it to scale with the increasing number of CPUs. I performed the experiments on a 1st Generation Core i7 (Eight Logical Cores) CPU and a Core 2 Duo CPU (2 logical Cores).

I observed the best performance when the application created the number of threads equal to the number of available CPU cores. The classification task completed in the i7 CPU with 8 active threads in 13 minutes compared to almost 35 minutes with 2 active threads on the Core 2 Duo CPU. The task completed in around 38 minutes in the i7 CPU with 2 active threads.

4 Results and Analysis

Language Pair	Recall	Precision	F1-Score	Accuracy
NE-EN	0.980	0.968	0.974	0.951
ES-EN	0.883	0.489	0.630	0.699

Table 2: Tweet level results on the test data.

My system obtained an accuracy of 95.1% in the tweet-level evaluation and 79.4% accuracy in the Facebook post-level evaluation of English-Nepali test tweets. Although, it was third in tweet-level evaluation, it was only 0.7% behind the best tweet-level system in terms of accuracy. My system was second in Facebook post-level evaluation by 6.9%. It had an accuracy of 94.6% and 86.5% in the token level evaluation of English-Nepali test tweets and Facebook posts respectively. The model was third in the tweet-token evaluation but stood first in the Facebook-post token evaluation. These results align with the hypothesis of the Incremental N-Gram Occurrence Model that token belonging to the same language will have more overlap of the preceding characters.

Category	Recall	Precision	F1-Score
lang1	0.944	0.949	0.947
lang2	0.965	0.964	0.965
mixed	0.000	1.000	0.000
ne	0.510	0.657	0.574
other	0.968	0.935	0.951

Table 3: Token level results on the test data for Nepali-English.

Category	Recall	Precision	F1-Score
lang1	0.866	0.761	0.810
lang2	0.750	0.861	0.802
mixed	0.000	1.000	0.000
ambiguous	0.000	0.000	0.000
ne	0.155	0.554	0.242
other	0.847	0.823	0.835

Table 4: Token level results on the test data for Spanish-English.

My system obtained an accuracy of 69.9% in the tweet-level evaluation and 70.0% accuracy in the Facebook post-level evaluation of the English-Spanish test data. It was the least effective in both the evaluation tasks. My system had an accuracy of 80.3% and 87.6% in the token level evaluation of English-Spanish test tweets and Facebook posts respectively. The model was again the least effective in both the token level evaluation task but by a smaller margin. The results do not exactly follow the hypothesis, but we can say it supports it because English and Spanish languages share a lot of common word prefixes. Hence my method is more likely to incorrectly predict some Spanish words as English and vice-versa.

It is evident from the results that this model is suitable when the languages being classified are

Language Pair	Recall	Precision	F1-Score	Accuracy
NE-EN	0.900	0.486	0.632	0.794
ES-EN	0.882	0.493	0.633	0.700

Table 5: Tweet level results on the surprise data.

Category	Recall	Precision	F1-Score
lang1	0.913	0.802	0.854
lang2	0.936	0.911	0.923
ne	0.394	0.833	0.535
other	0.886	0.696	0.780

Table 6: Token level results on the surprise data for Nepali-English.

Category	Recall	Precision	F1-Score
lang1	0.853	0.756	0.801
lang2	0.746	0.839	0.789
mixed	0.000	1.000	0.000
ambiguous	0.000	0.000	0.000
ne	0.145	0.550	0.230
other	0.826	0.808	0.817

Table 7: Token level results on the surprise data for Spanish-English.

highly dissimilar in syntax and structure. As English and Nepali language do not have the same ancestry they have very different syntax and structure. The word prefixes used frequently to form Nepali words and the syntax of forming various parts of speech in Nepali language is quite different than in the English language.

In both the training and test datasets, the ratio of code-switched to monolingual tweets is higher in Nepali than in Spanish, which probably led to my system performing worse on tweet level for Spanish. Although, this distribution can be anticipated because English is taught from primary schooling levels in Nepal. Almost all the literate population can communicate pretty well in English. Nepal is a country that relies heavily in the tourism industry, and English being a universal language is a second language in major cities and travel destinations of the country. All these factors have led to a lot of code switching in tweets Nepali tweets. On the other hand, Spanish is a widely spoken language itself. The people who know Spanish rarely need to learn a second language. This might be the reason that there are less code-switched tweets for Spanish.

My model also has a drawback, which is also demonstrated by my evaluation results. Spanish and English languages do share a lot of common prefixes. This maybe due to their shared Indo-European ancestry and the fact that English language has borrowed a significant number of words from the French language, which is very similar to the Spanish language. The word ”precious” and ”bilingual” in English is spelled ”precioso” and ”bilingue” in Spanish. This similarity of prefixes leads the Incremental N-gram model to classify tokens wrongly based upon the recurrence of the same prefixed words documented more frequently in one language than the other. It further results in a large number of English-Spanish tweets and Facebook posts to be verified as code switched because, just one token in a tweet that is wrongly classified as belonging to another language class, will validate the tweet as code-switched. To counter this drawback, when classifying words of the language that have the same ancestry and similar structure and syntax, only the prefixes should not be considered.

Another important thing to note is that the task of evaluation is very taxing on the CPU and takes a lot of time. Various evaluation techniques are applied to a token before its correct class is determined. This time consuming process can be accelerated significantly by designing a system that follows the data and task parallelism principles i.e. multithreading. The redesign of the system to support multithreading made the training process almost 3 times faster.

5 Conclusion and Future Work

The method described in this paper is useful in language identification of code-switched text. It works especially well when the two languages in question have different word formation syntax and structure. For the languages that are similar in ancestry and when one language contains many words derived from the other language, like Spanish and English, this method is not very reliable. For these types of languages, considering that they have similar syntax and structure, the use of all the possible n-grams of the tokens in the training set and their frequencies might be useful. Also considering the suffixes of the word rather than just the prefixes might provide greater accuracy for prediction of these types of languages. These tasks are left as future improvements.

Acknowledgment

I would like to thank the organizers of the Computational Approaches to Code Switching Workshop at EMNLP'14 who gave me an opportunity to participate in this task.

References

Peter Auer. 1988. A conversation analytic approach to code-switching and transfer. *Codeswitching: Anthropological and sociolinguistic perspectives*, 48:187–213.

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175.

Anik Dey and Pascale Fung. 2014. A hindi-english code-switching corpus. In *The 9th International Conference on Language Resources and Evaluation (LREC)*, Reykjavik.

Heba Elfardy and Mona T Diab. 2012. Token level identification of linguistic code switching. In *COLING (Posters)*, pages 287–296, Mumbai, India.

Constantine Lignos and Mitch Marcus. 2013. Toward web-scale analysis of codeswitching. In *Annual Meeting of the Linguistic Society of America*.

Dong Nguyen and A Seza Dogruoz. 2014. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gonheim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar.