Sentiment Classification on a Code-Mixed data

Introduction

According to a study done in 2021, there are approximately 43% of the population. A further 17% of people are multilingual. More than half of the world's population can speak more than one language. Another trend seen in the multilingual speaking area is that the people in the age group of 16-24 years are more likely to speak more than one language. This is the same age group that is seen to be the most active on social media platforms like Twitter, Instagram, and Facebook.
Although information retrieval systems and natural language processing systems have seemed to have gained high accuracy in the cross-lingual text classification area, the code mixed text datasets themselves seem to be very scarce. 30% of queries to Ruuh (a Microsoft chatbot) are code mixed. Alexa gets code-mixed speech inputs. Around 2-30% of posts on Twitter and Facebook are code mixed. Observing this shift in the trend of linguistics, having a multilingual retrieval model seems to be a necessity more than an added feature to a search engine. A study of 830k tweets from the Hindi-English bilinguals also indicated that the native language, Hindi was more preferred for expressing negative sentiments while English was preferred for the positive ones. These statistics also tell us that multilingual retrieval models are not only restricted to understanding the meaning of the query, but also the sentiment.

The few challenges faced for sentiment analysis with a code-mixed data are:

- In code-mixed data, word order is completely lost.
- There are no proper spelling rules like the word 'super' could be written as 'sooper', 'suuupar' which makes it difficult to normalize.
- Users create their spellings like 'YouTube' written as 'UTube' or 'great' as 'gr8' which poses a challenge for sentiment analysis.
- Abbreviations commonly used on social media like 'FDFS' meaning 'First Day First Show'
- capitalization is not followed while writing informally on social media

Cross-lingual sentiment classification state of art systems have received an accuracy of 83% in the English dataset, while a 92.5% accuracy for positive reviews on French data. While the code mixed data is scarce, there are several attempts made to create a model giving a good accuracy on the multilingual text.

Survey

There has been a rise in the research done in the area of code-switching ever since 2014. I have reviewed three other proposed models.

The paper, 'Robust Deep Learning Based Sentiment Classification of Code-Mixed Text', proposes an ACCMT (Attention-based deep learning technique for Sentiment Classification on Code-Mixed Text) model on a Hindi-English dataset. This architecture gives a good accuracy of 71.97% exceeding the baseline accuracy. There have been similar research works done but have

lacked the capacities of capturing information related to word levels semantics or handling misspelled words and a wide variety of code-mixed texts. The phonetic similarity of various words across the languages in the Code-Mixed text increases the challenge due to the ambiguity of the meaning of a word. This problem could be solved by using clean data we have, dependent on the word-embedding-based input. The proposed model consists of two parts -
i. preparing a suitable word-embedding of code-mixed text
ii. then building a robust deep learning architecture for the classification of code-mixed data
For the first step, we need to prepare a corpus of the Hindi Romanized text, preparing the word embeddings using fastText due to the presence of multiple homo-phonic representations. Lastly, we have to ensure that the words from both languages that are similar have representations nearby. SVD is used for learning a linear transformation. The alignment of two monolingual vectors (from 2 languages) into a single vector is done by SVD.
Next, we build the ACCMT model that learns from both the character and word-based representations. Out of the two, the first part learns the sub-word level features from the input. This part is the baseline model and helps resolve issues like code mixed text like non-standard spelling and phrasal contraction. The second part captures features for word-level semantic representation to counter the limitations of the first part.
There are two separate attention layers over the LSTM output of character-based and word-based sides. This is for inferring the prominent features from the two representations. These two outputs are then concatenated.

Another problem faced while dealing with code-mixed sentiment analysis with not much work focused on it is the imbalanced class label distribution in the data. The paper, 'Sentiment analysis imbalanced code-mixed data using machine learning approaches' deals with this issue using various ways. The expected outcomes of this paper are -
i. to separate the non-code-mixed data from the code-mixed data using an enhanced spell check algorithm
ii. making a lexicon dictionary and words with spelling variations are normalized using Levenshtein distance
iii. extracting the sentiments using machine learning techniques
A Tamil-English and Malayalam-English dataset scraped from YouTube comments has been used in this paper. The sentences are divided into Tamil-English and English scripts. The English script sentences are eliminated from the data to increase the F1 score. Another thing done is normalizing recurrent characters and recurrent words. Recurrent characters entail words like "wowww" for the word "wow" to express their increased degree of emotion. Similarly, sentences like, "Thala vera vera level panitaru" (meaning the person has performed extremely well) have the word "vera", meaning "very", used repeatedly for expressing the degree of emotion.
Levenshtein distance is used for normalizing words. The edit distance is kept to two since it is found that when the distance is more than two, the word transforms into a different word not matching the meaning of the base word. The use of Levenshtein distance helps in reducing the computational complexity and results in an increased F1 score.
Resampling is the core of solving the problem the paper deals with - the imbalance in the data. Oversampling and undersampling techniques have proven to help solve the imbalanced class label distribution for multi-class classification. Oversampling is preferred over undersampling and the SMOTE (Synthetic Minority Over-Sampling) and ADASYN (Adaptive Synthetic) techniques.

Among the various feature extraction techniques, TF-IDF is chosen since code mixed data usually contains words that have less frequency and need to be given weightage. BOW on the other hand gives higher weightage to the frequently occurring words in the documents. The machine learning models used for classification were probabilistic (like Naive Bayes), linear classifiers (SVM), decision-based (Random Forest classifier, XGBoost classifier) and statistical models (logistic regression).
Without separating code mixed and non-code-mixed data Logistic regression performs the best. Naive Bayes fails to predict other classes except positive since most algorithms seem to be more biased towards the positive class. This imbalance can be attributed to the non-separation of the code-mixed and non-code mixed data. Another interesting observation made is that resampling techniques improve the F1 score by 50%.
After removing the non-code-mixed data the F1 score increased by 2%. Levenshtein distance overcame the spelling mistakes and many strong features of a class had alleviated the class imbalance problem to an extent. And yet the class imbalance was not solved completely due to the ambiguous muti word contexts. One class pulls down the other class. Out of all Logistic regression proves to be the best with multi-class handling capabilities.

In the paper, 'Supervised sentiment analysis in the multilingual environment', a Spanish dataset and an English dataset scraped from tweets, is used separately for monolingual models, then fused and another code-mixed 'ES-EN' dataset is used for experimentation. This paper deals with comparing existing state-of-art models. A multilingual model trained on a multilingual dataset (by fusing two existing monolingual datasets), a dual monolingual model with a perfect language detection on monolingual texts, and a monolingual model that acts based on the decision provided by a language identification tool.
11,400 Spanish-English tweets were used as the training set for the models. Tweets were then divided into four categories:
i. the ones that show sentiments in both the languages
ii. the ones that had sentiments only in the English part
iii. the ones that contained the sentiments in the Spanish part of it
iv. the ones in which sentiment was independent of the language, eg, using emoticons
The tweets were annotated by three speakers fluent in English and Spanish. Two scores were provided for each tweet, one for positive strength and the other for negative. These two scores were averaged to get the final sentiment score. Around 200 tweets faced a contradiction in the annotations by the annotators, with at least one being annotated as positive and the others negative. These could case of strong disagreement, irony, mixed feelings in one sentence, or the overuse of subjective acronyms. To give a few samples of these, 'lol miss you too!!! :p mmmmm hahahaha.' got a score of (4,1), (1,2), (4,1) - positive and negative scores. 'This movie is badass like damn and makes me cry lol' was scored (4,1), (1,5), (4,1). The ubiquitous use of subjective clauses and abbreviations like 'lol' and 'lmao' is controversial given that they are sometimes used as the negative part of a message without any positive connotations to it, for example, 'He is so stupid, lmao'.
The L-2 regularized logistic regression has been shown to work the best for this task. The basic features utilized in this model are the words, lemmas, psychometric properties and part-of-speech tags which are all combined to obtain a rich linguistic model that improves performance. Along with these syntactic features and N-grams are used as well for getting the dependencies between words and features.

The experiments show that out of the three models discussed earlier, the multilingual model is the most robust. It gives similar results on a monolingual corpus as given by the monolingual approaches. The models based on language detection don't seem to be optimal on the code-switched dataset.

Conclusion

After reviewing these research papers a few things about the topic of sentiment analysis using code-mixed data are certain. Data cleaning is one of the most important steps in the problem. If we want to classify code-mixed text only we should filter out the non-code-mixed text before training the model which has shown to improve the F1 score significantly by 50%. If we choose to have a monolingual model as well, then separating the monolingual languages in their respective places and then training the model accordingly would give good results. Normalization is also an important step, which is done using various basic features like words, lemmas, POS tagging, and Levenshtein distance in the above papers along with syntactic features and N-grams for word and feature dependencies. Class imbalance is an important factor considered which is solved by resampling techniques like oversampling and undersampling which increases the F1 score by 2%. Another observation is that the less accuracy got could be attributed to the less data we have.
Logistic regression is the state-of-art model for this data. Another model used is the one that consists of the LSTM attention-based model giving a comparable performance. The F1 score and accuracy are the evaluation metrics that are chosen for this problem. The future scope of research in this area could be done by experimenting with the data on deep learning models like the CNN architecture due to its ability to exploit spatially-local correlations and it is seen that code-switched sentences tend to contain continuous slices of text written in the same language. We could also try to work on the challenges mentioned in the introduction that mainly deal with the data cleaning and normalizing. We could also try to work on this problem with unsupervised learning since there seems to be a scarcity of labeled data.