CS 529 - Topics and Tools in Social Media Data Mining

PROJECT TITLE

Sentiment Analysis for Hinglish Code-mixed Tweets by means of Cross-lingual Word Embedding

Submitted by –

- ***Anamitra Mukherjee (214101007)***
- ***Rittick Mondal (214101041)***
- ***Souvik Gorai (214101055)***
- ***Prosenjit Biswas (214101038)***

Phase - I

Introduction: -

Here we investigate the use of unsupervised cross-lingual embeddings for solving the problem of code-mixed social media text understanding. We specifically investigated the use of these embeddings for a sentiment analysis task for Hinglish Tweets, viz. English combined with (transliterated) Hindi. First let us look at the terms word embedding as well as cross lingual embedding. Word Embeddings are projection of words in higher dimensional vector spaces based on their distribution in the corpus. Cross-Lingual embedding methods jointly learn embeddings of words of more than one language in a shared space. Cross-Lingual embeddings also helps in knowledge transfer between languages. Now, let's discuss the term code-mixing, which is often used in the report. Code-mixing is a frequent phenomenon in user-generated content on social media. In linguistics, code-mixing traditionally refers to the embedding of linguistic units (phrases, words, morphemes) into an utterance of another language (Myers-Scotton, 1993). In that sense, it can be distinguished from code-switching, which refers to a "juxtaposition within the same speech exchange of passages of speech belonging to two different grammatical systems or subsystems" (Gumperz, 1982), where the alternation usually takes the form of two subsequent sentences. Here, we have considered code-mixing as a phenomenon where linguistic units in Hindi are embedded in English text, or the other way around, but this can take place both at the sentence and word level. As a consequence, we will use the term code-mixing as an umbrella term that can imply both linguistic phenomena.

The phenomenon of code-mixing frequently occurs in spoken languages, such as for instance a combination of English with Spanish (so-called Spanglish) or English with Hindi (so-called Hinglish). More recently, due to the massive usage of social media platforms, it is increasingly used in written text as well. This social media content is very important to automatically analyze the public opinion on products, politics or events (task of sentiment analysis), to analyze the different emotions of the public triggered by events (task of emotion detection), to observe trends, etc.

Related research: -

Related research on computational models for code-mixing is scarce because of the rarity of the phenomenon in conventional text corpora, which makes it hard to apply data-greedy approaches. Previous research, however, has tried to predict code-switching in English-Spanish (Solorio and Liu, 2008a; Solorio and Liu, 2008b) and Turkish Dutch (Nguyen and Seza Dogruoz, 2013) text corpora. More recently, research has been performed to study codeswitching on social media from a computational angle. Vyas et al. (2014) have compiled an annotated corpus for Hindi-English from Facebook forums, and performed experiments for language identification, back-transliteration, normalization and part-of-speech tagging on this corpus. They identify normalisation and transliteration as very challenging problems for Hinglish. Similar work has been carried out by Sharma et al. (2016), who developed a shallow parser for Hindi-English code-mixed social media text. Rijhwani et al. (2017) introduce an unsupervised word level language detection technique (using a Hidden Markov Model) for code-switched text on Twitter that can be applied to

different languages. Pratapa et al. (2018) compare three bilingual word embedding approaches, bilingual correlation-based embeddings (Faruqui and Dyer, 2014), bilingual compositional model (Hermann and Blunsom, 2014) and bilingual Skip-gram (Luong et al., 2015), to perform code-mixed sentiment analysis and Part-of-Speech tagging. In addition, they also train skip gram embeddings on synthetic codemixed text. Their results show that the applied bilingual embeddings do not perform well, and that multilingual embeddings might be a better solution to process code-mixed text. This is mainly due to the fact that code-mixed text contains particular semantic and syntactic structures that do not occur in the respective monolingual corpora Seminal work in sentiment analysis (SA) of Hindi text was done by Joshi et al. (Joshi et al., 2010), who built a system containing a classification, machine translation and sentiment lexicon module. Bakliwal et al. (2012) created a sentiment lexicon for Hindi, and Das and Bandyophadhyay (2010) created the Hindi SentiWordNet. Joshi et al. (2016) introduced a Hindi-English code-mixed dataset for sentiment analysis and propose a system to SA that learns sub-word level representations in LSTM (Long Short-Term Memory) (Subword-LSTM) instead of character- or word-level representations.

Challenges: -

Code-mixing is, however, very challenging for standard NLP pipelines, which are usually trained on large monolingual resources (e.g., English or Hindi). As a result, these tools cannot cope with code-mixing in the data. In addition, social media language is characterized by informal language use, containing a lot of abbreviations, spelling mistakes, flooding, emojis, emoticons and wrong grammatical constructions. In the case of Hinglish, an additional challenge is added because people do not only switch between languages (e.g., English and Hindi), but also use English phonetic typing to write Hindi words, instead of using the Devanagari script.

There are few models proposed in the paper “Sentiment Analysis for Hinglish Code-mixed Tweets by means of Cross-lingual Word Embeddings” by Pranaydeep Singh and Els Lefever for tacking this problem. Below we discuss those methods in details.

Proposed models in the paper: -

In a first step, baseline models, initialized with monolingual embeddings obtained from large collections of tweets in English and code-mixed Hinglish, were trained. In a second step, two systems using cross-lingual embeddings were researched, being (1) a supervised classifier and (2) a transfer learning approach trained on English sentiment data and evaluated on code-mixed data. We demonstrate that incorporating cross-lingual embeddings improves the results (F1-score of 0.635 versus a monolingual baseline of 0.616), without any parallel data required to train the cross-lingual embeddings. In addition, the results show that the cross-lingual embeddings not only improve the results in a fully supervised setting, but they can also be used as a base for distant supervision, by training a sentiment model in one of the source languages and evaluating on the other language projected in the same space. The transfer learning experiments result in an F1-score of 0.556 which is almost on par with the supervised settings and speak to the robustness of the cross-lingual embeddings approach.

To train and evaluate our sentiment analysis system for Hinglish, we use the training data provided for the SemEval 2020 shared task on sentiment analysis in code-mixed social media text (Das et al., 2020). This dataset for Hinglish contains 15,131 instances, which have been labeled as positive, negative, or neutral. Besides the sentiment labels, the organisers also provide the language labels at the word level, consisting of the following tags: en (English), hi (Hindi), mixed and univ (e.g., symbols, @ mentions, hashtags). As mentioned before, the data set contains a mixture of English and romanized or transliterated Hindi. This produces an additional challenge, as this romanized codemixed data contains non-standard spellings like aapke and apke ("your"), non-grammatical constructions like "Wow the amusement never ends even after the election Daily soap bana ke rakh diya" which combines an English sentence with a Hindi sentence mid-way, and words which combine an English word with a Hindi alteration like Jungli ("wild") and Filmy ("glamorous"). Although the data set is tagged with a language label for every word, we did not use this information in our experiments as our aim was to build a common bilingual model that would be applicable for other code-mixed data sets as well.

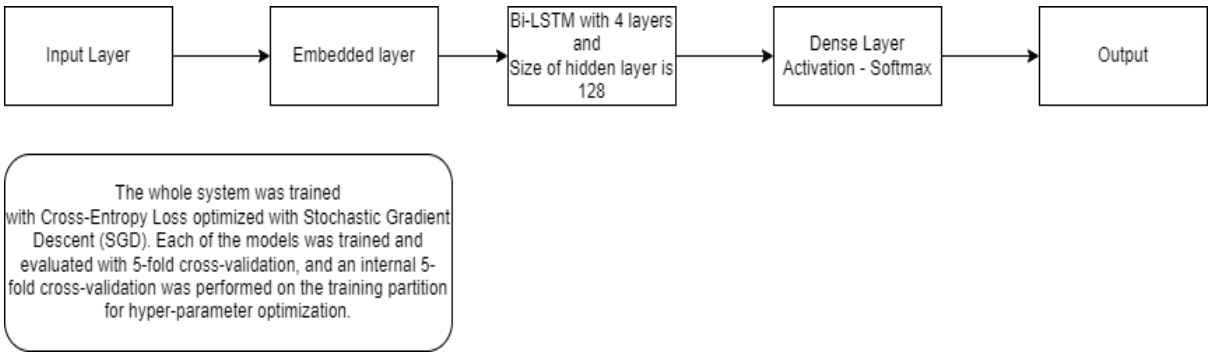

Fig 1: Model Architecture

First let us discuss the architecture of the model discussed in the paper (Fig 1). Here, Bi- LSTM encoder was used followed by a Softmax layer. Pre-trained crosslingual or monolingual embeddings were fed to the LSTM, the size of the hidden layer was 128 and we incorporated 4 layers in our model. This was followed by a single linear layer and the whole system was trained with Cross-Entropy Loss optimized with Stochastic Gradient Descent (SGD). Each of the models was trained and evaluated with 5-fold cross-validation, and an internal 5-fold cross-validation was performed on the training partition for hyper-parameter optimization. The paper investigated two different methods to train our sentiment analysis system for Hinglish code-mixed tweets and compared them with monolingual baseline systems, resulting in the following three experimental setups:

- ***Baseline Monolingual Systems***: Models exclusively trained using monolingual embeddings
- ***Supervised Classification***: Models incorporating cross-lingual English-transliterated Hindi embeddings
- ***Transfer Learning***: Models trained with no supervision on the Hinglish data set but deriving knowledge from the English sentiment data sets

❖ ***Baseline Systems with Monolingual Embeddings: -***

Our baseline models were trained with monolingual embeddings in both languages, viz. code-mixed Hindi (Baseline H) and English (Baseline E). To train these monolingual embeddings, we first scraped tweets by means of the Twitter API in both English and transliterated Hindi. For English 141,566 tweets were scraped, while 252,183 tweets were scraped for Hindi. Hinglish tweets were obtained from the API by querying Hindi tweets and then filtering out tweets containing any Devanagari characters. We were left with 138,589 tweets for Hinglish after removing these 'Devanagari' tweets. Subsequently, monolingual embeddings were trained for both of the above-mentioned corpora with a continuous bag-of-words FastText model (Bojanowski et al., 2017), and used to train a bi-directional LSTM.

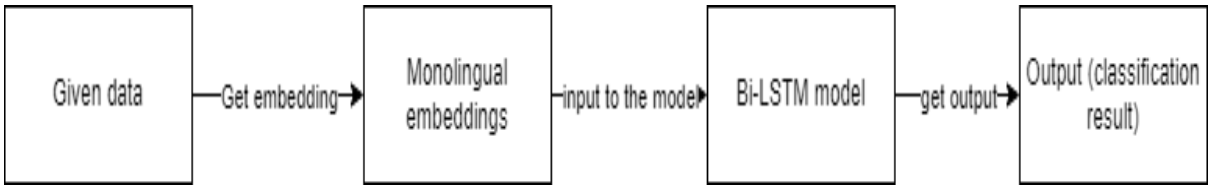

Fig 2: Baseline model

❖ ***Supervised Sentiment Analysis with Cross-lingual Embeddings: -***

Cross-lingual embeddings rely on the inherent similarities in language structure and composition to project multiple monolingual embeddings into the same space, enabling tasks which require knowledge of more than one language (Conneau et al., 2018). This kind of embeddings have been used to solve a variety of tasks like wordto-word translation (Chen and Cardie, 2018), evaluating sentence similarity (Bjerva and O¨ stling, 2017) and detecting cognates across languages (Labat and Lefever, 2019). Most methods to project two or more monolingual embeddings into a shared space require a parallel seed dictionary to initialize an alignment which can then be improved upon (Upadhyay et al., 2016). The latter approach is not feasible, though, in this particular setting, as we aim to align English words with code-mixed (transliterated) Hinglish words, which often have no standardised spelling, but on the contrary occur with many variations in social media data. In recent research, however, a number of methods have been explored that seek to create a projection without any seed dictionary by relying on certain basic characteristics of a language in an embedding space. For our experiments, we evaluated two of these methods, namely the Multilingual Unsupervised and Supervised Embeddings (MUSE) Python library1 and the VecMap toolkit2, to create cross-lingual embeddings. We selected these methods in particular because of high performance in a number of downstream cross-lingual tasks and the lack of parallel data required to train the cross-lingual embedddings. The MUSE ((Multilingual Unsupervised and Supervised Embeddings) toolkit (Lample and Conneau, 2019) uses a domain-adversarial

setting to compensate for the lack of supervision. If the mapping matrix is referred to as W, and the respective monolingual embeddings are referred to as X and Y, then the discriminator is trained to distinguish between WX and Y, whereas W is trained to prevent the discriminator from making accurate predictions by aligning WX and Y as closely as possible. Moreover, an iterative refinement tool using the Procrustes solution is used to further improve the alignment using synthetic dictionaries created from the most frequent words. The VecMap toolkit (Artetxe et al., 2018), on the other hand, starts from the principle that if a similarity matrix of all words in a vocabulary was to be created, then every word would have a unique distribution and that this distribution would be consistent across languages. This principle is used to induct an initial seed dictionary. Optimal orthogonal mappings are then computed using Singular Value Decomposition while iteratively using the improved seed dictionary created by the current mapping. Multiple tweaks to the method, like bi-directional induction of the seed dictionary and symmetric re-weighting of the target language embeddings according to cross-correlation, further improve the quality of the mappings. For our experiments, we tested two variants of both the VecMap and MUSE cross-lingual embeddings: (1) embeddings aligned with an entirely unsupervised dictionary induction method and (2) embeddings aligned using numerals and common tokens like “https” as a bilingual seed dictionary. This method is especially interesting to look at as there is a decent overlap between the vocabulary of both embeddings as Hinglish is a derivative of English. The classifiers were then trained and tested by means of 5-fold cross-validation on the SemEval 2020 data.

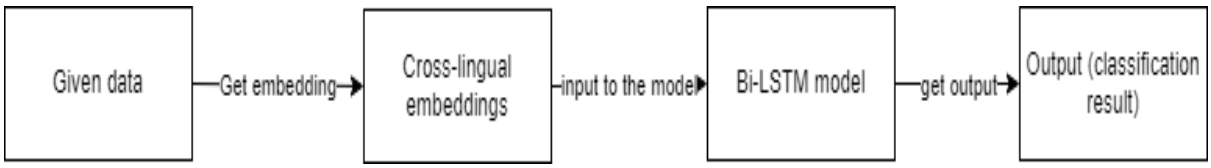

Fig 3: Supervised Sentiment Analysis with Cross-lingual Embeddings

Now let us briefly look into vecMap as well as MUSE.

VecMap (Artetxe et al., 2017) uses an orthogonal transformation over normalized word embeddings. Its semisupervised two-step procedure is specifically designed to avoid the need for a large seed dictionary. For instance, in the original paper, a seed dictionary with 25-word pairs was used. This seed dictionary is then augmented by applying the learned transformation to new words from the source language. The process is repeated until some convergence criterion is met. The unsupervised variant (Artetxe et al., 2018b) obtains the initial seed dictionary automatically by exploiting the similarity distribution of words, and then applies the same method followed by a refinement step that reweights the embeddings based on the cross-correlation of their components, which makes it the only non-orthogonal method tested in this work. The motivation behind vecMap is that two words having same meaning in different languages will have the same distribution. By restricting transformations to orthogonal linear mappings, VecMap and MUSE rely on the assumption that the monolingual embeddings spaces are approximately isomorphic (Barone, 2016). However, it has been argued that this assumption is overly restrictive, as the isomorphism assumption is

not always satisfied (Søgaard et al., 2018; Kementchedjhieva et al., 2018). For this reason, it has been proposed to go beyond orthogonal transformations by modifying the internal structure of the monolingual spaces, either by giving more weight to highly correlated embedding components, as is the case for the unsupervised variant of VecMap in this work (Artetxe et al., 2018a), or by complementing the orthogonal transformation with other forms of post-processing. As an example of this latter strategy, Doval et al. (2018) fine-tunes the initial alignment by learning an unconstrained linear transformation which aims to map each word vector onto the average of that vector and the corresponding word vector from the other language.

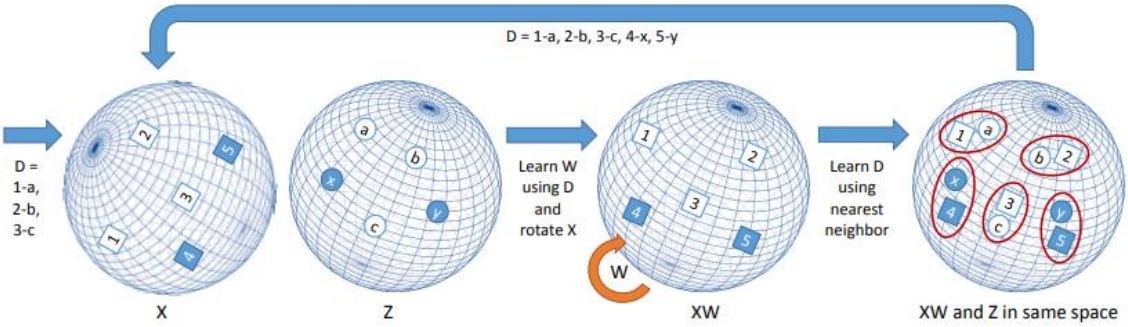

Fig 4: VecMap - Tries to learn a mapping W based on the seed dictionary D, which is then used to learn the full dictionary.

On the other hand, MUSE (Multilingual Unsupervised and Supervised Embeddings) is a Python library that enables faster and easier development and evaluation of cross-lingual word embeddings and natural language processing. This library enables researchers and developers to ship their AI technologies to new languages faster. MUSE takes a novel approach to natural language processing. Rather than relying on language-specific training or intermediary translations in order to classify text, it utilizes multilingual word embeddings to enable training across many languages to help developers scale. MUSE is compatible with fast Text, and offers large-scale, high-quality bilingual dictionaries for training and evaluation. It's available on CPU or GPU, in Python 2 or 3.

Objective of MUSE is that we try to Learn a mapping W (matrix) between X (Source Lang embeddings) and Y (Target Lang embeddings). Ex. want to align the embedding spaces so that cat in English matches with gato in Spanish when you do a nearest neighbour search.

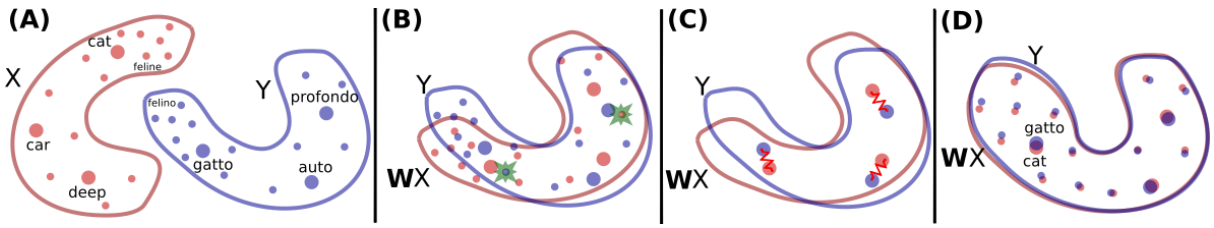

Fig 5: MUSE (Multilingual Unsupervised and Supervised Embeddings)

X, Y = English, Italian embedding spaces. W is the rotating/aligning matrix, which rotates X so that WX is similar to Y

❖ ***Transfer Learning with Cross-lingual Embeddings*: -**

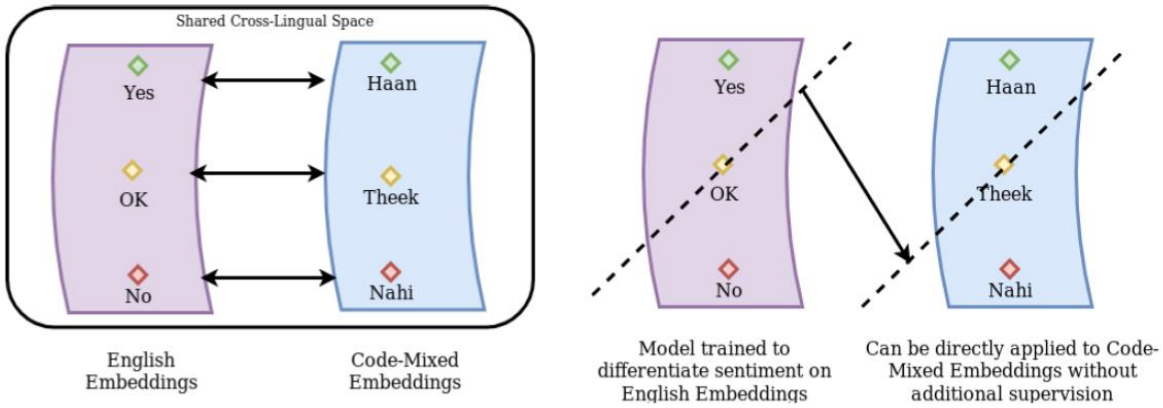

Fig 6: Transfer Learning based Sentiment Analysis for Hinglish, using cross-lingual embeddings

Approaches like VecMap and MUSE allow us to find an alignment which transforms monolingual embeddings into a shared space. Since this projection is done with no supervision (or minimal supervision in the case where numerals and identifiers are used as a seed dictionary), it should also be possible to train sentiment models for one of the languages and evaluate them on the other language. This can work if we assume that the model learns the sentiment-related information in the shared space in which both languages reside. To test these assumptions, we train a bi-directional LSTM on the English sentiment data of the SemEval-2016 “Sentiment Analysis in Twitter” task (Nakov et al., 2016) using English embeddings in the same shared space as code-mixed Hinglish embeddings. We then evaluate the model on the SemEval-2020 Hinglish data set, using the Hinglish embeddings pre-aligned with English embeddings.

Figure 6 illustrates the intuition behind this experiment. Since the model learns to associate particular words to particular sentiments in English during the supervision step, it should ideally also pick up the corresponding words and their sentiments in the code-mixed data due to the shared space, and by consequence be able to perform sentiment analysis with no direct supervision in the code-mixed data. As in the supervised setting, we test for embeddings aligned with VecMap and MUSE, using both (1) the completely unsupervised (Unsupervised) and (2) the numerals and special characters seed methods (SeedDict).

Phase – II

Limitations of proposed models in research paper: -

There are some critical gaps in the proposed models in the research paper. First of all, VecMap method performs better than all other methods only when initialized with seed dictionary (referred to as vecMap seedDict) but it's not always possible to obtain seed dictionary for non-lexically overlapping languages, for e.g., English and Russian have only 24% overlap of vocabularies. Also, learning cross-lingual mapping matrix **W,** has following assumptions- Isometry Condition, Source and target languages have distribution about the same mean, embeddings of source and target languages are zero-centered, which may not reflect in the dataset.

So, these are some of the drawbacks of the proposed models for this challenging problem. Later we discuss our proposed model to make further improvements on this.

Objective for phase–II of project: -

We have already mentioned some drawbacks in the original approaches in the paper. So, our proposed model for improvement is XLM-RoBERTa. XLM-R, uses self-supervised training techniques to achieve state-of-the-art performance in cross-lingual understanding, a task in which a model is trained in one language and then used with other languages without additional training data. XLM-R improves upon previous multilingual approaches by incorporating more training data and languages - including so-called low-resource languages, which lack extensive labeled and unlabeled datasets. XLM-R is different from XLM, avoiding TLM (Translation language model) objective, it just trains Roberta on a huge multilingual dataset at a large scale. Around 100 languages were extracted from CommonCrawl datasets, i.e., 2.5TB of text data (unlabeled). XLM-R is trained only with an objective of the Masked language Model (MLM) in a Roberta way. XLM-R is a transformer-based multilingual masked language model (MLM) pre-trained on a text in 100 languages, which obtains state-of-the-art performance on cross-lingual classification, sequence labeling, and question answering.

Our goal is to beat the benchmark scores set by the paper by using the SOTA model of XLM-R.

Intuitions behind the objective for phase–II of project: -

XLM-R has achieved the best results to date on four cross-lingual understanding benchmarks, with increases of 4.7 percent average accuracy on the XNLI cross-lingual natural language inference dataset, 8.4 percent average F1 score on the recently introduced MLQA question answering dataset, and 2.1 percent F1 score on NER. After extensive experiments and ablation studies, it's shown that XLM-R is the **first multilingual model to outperform traditional monolingual baselines that rely on pretrained models**. This is the reason why we have decided to use SOTA model of XLM-R to tackle this very challenging problem.

Now let us discuss about the architecture of XLM-RoBERTa.

Architecture of XLM-RoBERTa: -

We can say that XLM-R follows the same approach as XLM, only introducing changes that improve performance at scale. XLM-R is a scaled-up version of XLM-100.The main training objective of XLM-R is the masked language model, as the name Roberta suggests, XLM-R is trained in a ROBERTa fashion i.e., using Masked language model objective.

XLM-R uses a Transformer model trained with the multilingual MLM objective using only monolingual data. It samples streams of text from each language and train the model to predict the masked tokens in the input, using sentence piece with a unigram language model applying subword tokenization on the raw text, and sample batches from different languages using the same sampling distribution of ($\alpha = 0.3$).

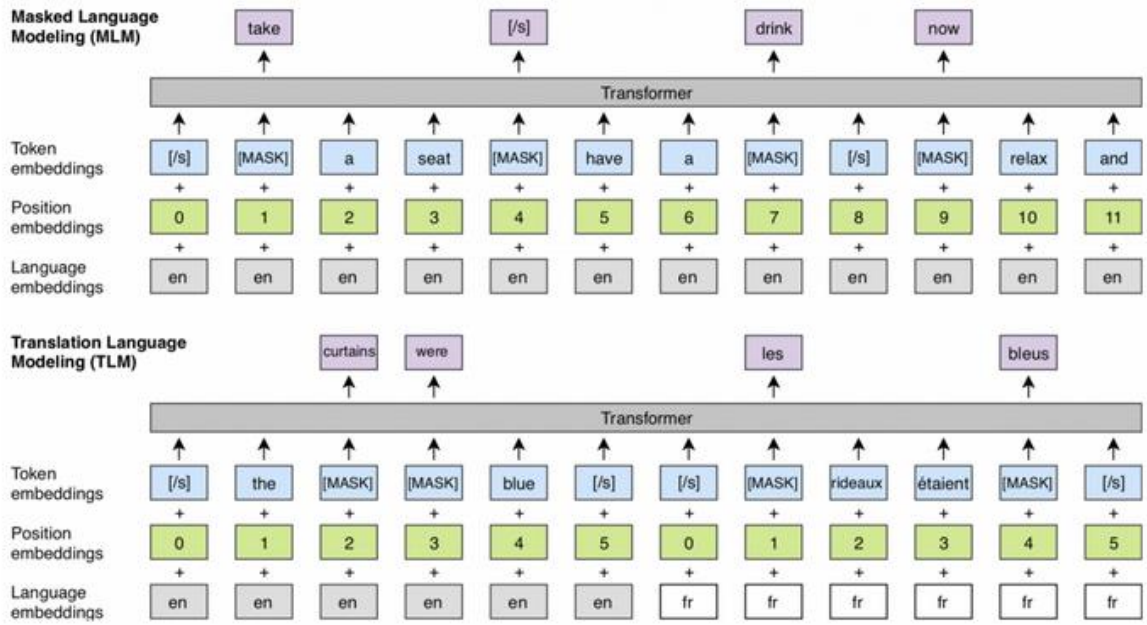

Fig 7: MLM and TLM

MLM main aim was to mask one or more words in a sentence and give the model to predict the masked tokens given others words in the sentence. The TLM objective extends

MLM to pairs of parallel sentences. Now to predict a masked English word, the model can attend to both the English sentence and its French translation, encouraged to align English and French representations. Now the model can leverage the french context if the English one is not sufficient to infer the masked the English words.

One of the biggest updates that XLM-R offer is the increase amount of training data. XLM-R was trained on 2.5TB of newly created clean Common Crawl data in 100 languages. It outperforms previously released multi-lingual models like mBERT or XLM on tasks like classification, sequence labelling and question answering.

One of the biggest advantages of using XLM-R is that it can be fine-tuned on just one language and zero-shot transfer to the other languages. XLM-R handles 100 languages and still remains competitive with monolingual counterparts.

Supporting experimental setup: -

1. **Baseline Monolingual Systems**: Models exclusively trained using monolingual embeddings

2. **Supervised Classification**: Models incorporating cross-lingual English-transliterated Hindi embeddings

3. **Transfer Learning**: Models trained with no supervision on the Hinglish data set but deriving knowledge from the English sentiment data sets

All the above methods are seen in the given paper. Apart from these models we have proposed the XLM-RoBERTa model as improvement. The reason for that choice as well as its architecture are mentioned above.

***Observed outputs*: -**

Below are the results from conducting the experiments. Here, we have only used F1 score as our evaluation metric, which is consistent with the given paper.

```
F1 Score:  0.5892191834051321
```

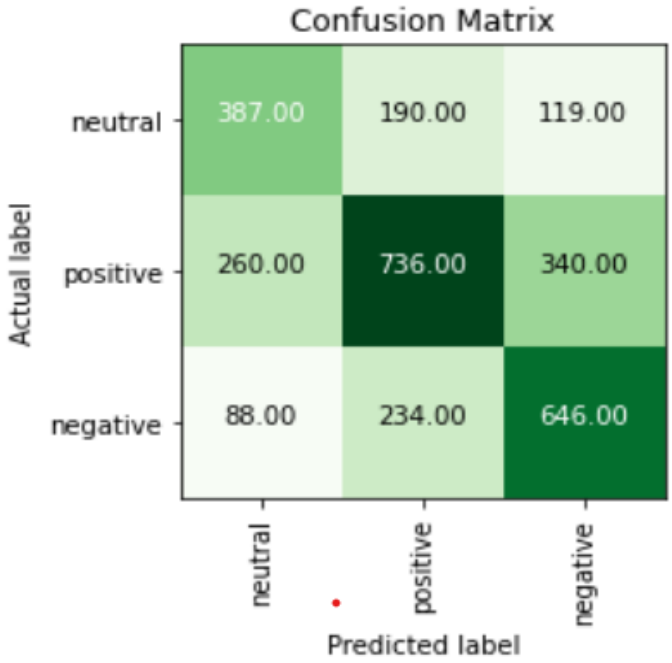

Fig 8: Results for the Baseline model

F1 Score: 0.6145317481773718

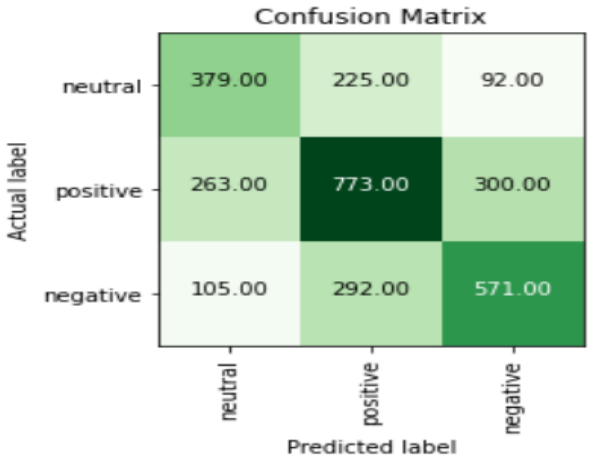

Fig 9: Results using VecMap

F1 Score: 0.6250145362398444

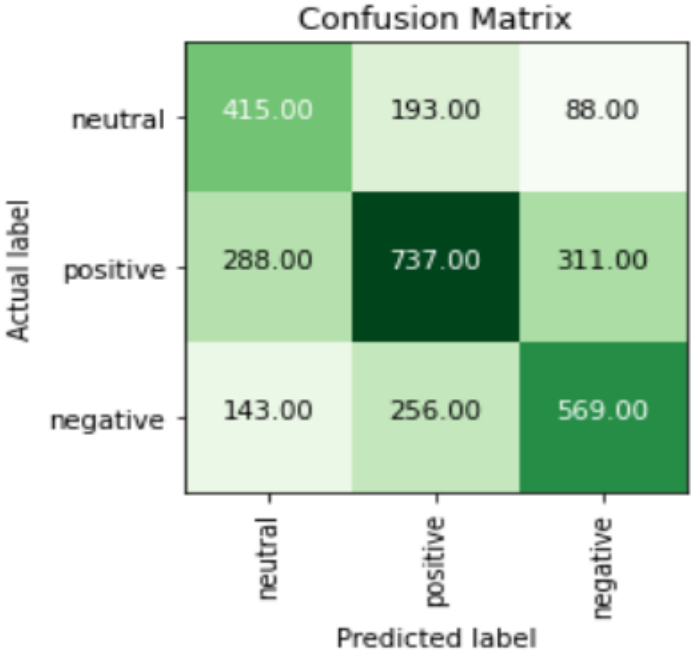

Fig 10: Results using XLM-RoBERTa

Observations from experimental results: -

The experimental results for our architectures reveal a number of interesting outcomes. Firstly, it can be noted that the SeedDict VecMap approach consistently outperforms other types of cross-lingual embeddings. While for the supervised experiments, the cross-lingual embeddings do not outperform classical embeddings by a large margin, there are small improvements which can be accounted for by the fact that we can use both English as well as code-mixed embeddings to classify a sentence, whereas only one of those can be used at a time in standard monolingual approaches. While the quality of the embeddings may have diminished due to the alignment process, the results are still better due to the increased vocabulary at our disposal.

A tweet like “One India sabka saath sabka vikas sabka visvas” (One India, with togetherness, progress and trust) is misclassified by the model incorporating monolingual english embeddings as “Neutral” since it cannot pick up the positive code-mixed Hindi words, while a tweet like “FF Have a great weekend” is misclassified by the monolingual code-mixed embeddings because of lack of knowledge of English words. Both of these tweets are, however, correctly classified by the VecMap embeddings using a seed dictionary. Also, the transfer learning-based model will also be able to perform sentiment analysis with acceptable accuracies without needing code-mixed supervision of any degree. As expected, XLM-R works the best among the given options.

Regarding the baseline approaches, it is also worth noting that the Code-Mixed Baseline does not perform a lot better than the English baseline as one would expect. This can probably be attributed to the quality of the monolingual embeddings, since the English embeddings were trained on the vast Common Crawl data while the Code-Mixed embeddings were trained on a little more than 100,000 scraped tweets. While the classification is understandably accurate for tweets containing a majority of English words like “Exclusive censor reports of Bharat is world class Words like movie of the year” and less reliable for sentences predominantly containing code-mixed words like “YouTube views ko vote samjhne wale agar is bar Nahi jita to Kabhi Nahi jitega”, the performance could be improved with better alignments and possibly a hybrid approach with minimal supervision.

Future work: -

As of now for these kind of challenging problems XLM-Roberta provides state-of-the-art solution. So, in future we can tackle this problem from a different direction. Also, we can play around with how the word embeddings are created and see how it affects our results. These are the next possible steps for this project.