Emotion Detection of Hindi-English Code-Mixed Data

Joy Barnes, Paresh Nayyar, Shubha Manohar,
Sruthi Venkatavaradan, Vishal Kumar Vasnani
jtbarnes@usc.edu, pnayyar@usc.edu, manohars@usc.edu,
sruthiv@usc.edu, vasnani@usc.edu

1 Introduction

With growth in communication over social media like Twitter and Facebook, huge amounts of natural language data is available that can be leveraged to create useful applications. Emotion detection is one such popular application that has been widely researched. Emotion detection helps in social media monitoring, product analysis, market research, customer satisfaction analysis, etc. A lot of work has been done in sentiment analysis of monolingual data as enormous amounts of labelled data is available for this. However, emotion detection for multilingual data is still a largely unexplored space.

Multilingual data in social media is a norm. One such widely used language in social media is Hinglish - a hybrid of English and Hindi. According to census 2011, 43.63% population of India speak Hindi and a detailed analysis of Hindi-English bilingual users on Facebook showed that 17.2% of all posts ,which accounted for around one-fourth of the words in their dataset revealed some form of code-mixing(Bali et al., 2014) Dealing with code-mixed data in itself is challenging due to lack of well annotated data, ambiguities in spellings, no grammar rules, abbreviations, etc.

Our project is an attempt to to assemble an annotated dataset for this, represent them appropriately with embeddings for code-mixed data, and further detect emotion from such bilingual Hindi-English (Hinglish) code-switched texts. We have collected and compiled a dataset of code-mixed tweets with labelled emotions. After pre-processing and code-mix normalization, we generated word embeddings using FastText(Bojanowski et al., 2016). We experimented with multiple models to predict the emotion results showed that Attention based Bi-directional LSTM performed the best with 86% accuracy.

2 Related Work

While sentiment analysis involves detecting the polarity of text, work on emotion analysis usually detects a subset of these emotion categories : happiness, anger, sadness, fear, disgust and surprise. RBEM-Emo (Tromp and Pechenizkiy, 2014) is a rule based emotion detection algorithm for the eight emotions defined by Plutchik's model, where emission scores are calculated using predefined rules and the final emotion is predicted using these. (Gaind et al., 2019) use a hybrid approach to detect emotion from English tweets by selecting words that match a generated Emotion-Words-Set and calculating an emotion score. Delving deeper into emotion categories, (Liew and Turtle, 2016) explore fine grain emotion detection in tweets where an SVM was trained to predict as many as 28 emotion categories.

While dealing with code-mixed data, there has been research on sentiment analysis with such datasets. (Lal et al., 2019) generate subword level representations for the sentences using CNN and use a Dual Encoder Network to predict the sentiment of Hindi-English code mixed data. The baseline paper (Wadhawan and Aggarwal, 2021) we have compared our work use a BERT model to detect emotion in Hindi-English code mixed data with an accuracy of 71.43%. Work in other languages, such as Malayalam-English(Chakravarthi et al., 2020) code mix data includes calculating TF-IDF feature vectors and experimenting with different machine learning models to perform sentiment analysis.

3 Method

3.1 Data

We collected around 140k tweets and labels from two different sources. The first dataset was provided by (Wadhawan and Aggarwal, 2021). This dataset gave us the tweet IDs and labels in CSV format. The challenge was to obtain the tweets from the tweet IDs. We used a JavaScript script to pull the tweets from tweet IDs using Twitter API. The

challenge we faced here was that the Twitter APIs have rate-limiting, which means we would not be able to extract all tweets in one go. To tackle this we implemented a batch processing mechanism which would call the Twitter APIs in batches and retrieve tweets in batches calling the API in fixed intervals. This took a significant amount of time and there were some tweets that were deleted since the time the IDs were generated, so we excluded those IDs. We were able to get the tweets and corresponding labels in JSON file.

Our observation of the data showed us that the data was skewed towards more political tweets. Thus, to balance this, we tried obtain more tweets. The second source (Sharma) gave us tweets directly with their labels and we merged these with our previous tweets hence creating our dataset.

3.2 Pre-processing

To clean the dataset and use it in the model pipeline, multiple pre-processing steps were applied.

Since the collected data was essentially tweets, first tweet-specific cleaning was required. This included removal of hashtags, URLs and other special characters like emojis. We then converted the text to lowercase and restricted the tweet length to avoid longer length texts.

Code-mix specific preprocessing was an integral part of our work. While dealing with tweets, normalizing them , i.e, converting misspelt and abbreviated words to a standardized form is a a necessity. Though there has been previous work in normalizing English tweets (Gupta and Joshi, 2017), there is limited research done on code-mixed data. We performed code-mix normalization in a rule-based fashion. An initial exploratory analysis was done to understand possible misspellings of frequently occurring words. On this, a set of rules were applied to translate them to their standardized forms. Apart from English stopword removal, we also performed code-mixed stop word removal. As there is no predefined list of stopwords for such a dataset, once again, analysis of most frequently occurring words was used to perform this filtering.

3.3 Embeddings

Before running the Machine Learning models, the preprocessed corpus has to be converted to embeddings. Word Embeddings are vector representations to encode the meaning of words such that similar words are closer in distance. Several word embedding methods are available for monolingual languages. However Hindi-English code-mixed pretrained bilingual word embedding is unavailable. Hence, we trained FastText (Bojanowski et al., 2016) on our corpus to obtain word embeddings for Hinglish.

FastText is a library for learning of word embeddings created by Facebook's AI Research lab in 2016. FastText is particularly useful as it not only learns the vector representations for the words present in the corpus but also has the ability to learn for Out Of Vocabulary (OOV) words. This is because FastText learns the weights for the entire word and also for the character n-grams of the word. Since the same word can have different similar spellings in social media (for example : chlo, chalo, chaalo), FastText would be the perfect choice for learning word embeddings for such a dataset.

We trained FastText on our corpus using the Gensim module which is a fast native C implementation of FastText with Python interface. It was trained using negative sampling with a window size of 10 for 10 epochs. This is then used to build the embedding matrix which is used as the input layer for the various deep learning models outlined in the following section.

3.4 Models

CNN Convolutional Neural Network is sustainable to data of matrix form and hence has been very successful for image classification problems where images are passed as input. For our problem of multi-class text classification, the embedding matrix will be passed as the input. Keras, an open-source neural network library (using Tensorflow backend) was used to build our Convolutional Neural Network architecture. The FastText word embeddings obtained for the tweet is passed as the input layer for CNN. The model is summarised as follows:

1. FastText Embedding layer (28 * 300)
2. 1D convolution layer (Relu Activation)
3. MaxPool layer
4. 1D convolution layer (Relu Activation)
5. MaxPool layer
6. Global MaxPool layer (upon which dropout is applied)
7. Dense layer (128 units)
8. Dense layer (64 units)
9. Output layer (Softmax Activation with 6 classes)

The hyperparameters were tuned and finally Adam optimizer and categorical cross entropy provided the best results.

LSTM LSTMs have been often successfully applied to binary text classification problems with good results. We considered LSTM mainly for two reasons, the dependency of a word's context on the previous words and not suffering from the vanishing gradient problem. Our Architecture consisted of a LSTM layer followed by two dense layers. The model is summarised as follows:

1. Embedding layer (28 * 300)
2. LSTM layer (150 units)
3. Dense layer (64 units)
4. Output layer (Softmax Activation layer with 6 classes)

BiLSTM In comparison to LSTM, BLSTM or BiLSTM has two networks, one accesses past information in forward direction and another accesses future information in the reverse direction. Because of this behavior, the context of a word depends not only on the previous words but also the following words. To perform sentiment analysis on hinglish dataset, we made use of model that consisted of:

1. Input layer (Sequence Length:28)
2. Embedding layer (28 * 300)
3. Bidirectional (each 150 units)
4. TimeDistributed (10 units)
5. Flatten
6. Dense layer (64 units)
7. Output layer (Softmax Activation layer with 6 classes)

BiLSTM with attention The technique of attention is based on learning the words which contribute the most towards the over all emotion of the sentence, and filtering out the words which contribute the least. The attention based model helps to overcome the long-range dependency problem of RNN and LSTMs (Figure 1). Attention provides a mechanism where output can 'attend to' (focus on) certain input time step for an input sequence of arbitrary length. The model is summarised as follows:

1. Input layer: Input Sequence (Sequence Length: 28)
2. Embedding layer (Fast Text, Dimension: 200000 x 300)
3. BiLSTM layer (64 units)
4. Attention layer
5. Output layer (Softmax Activation layer with 6 classes)

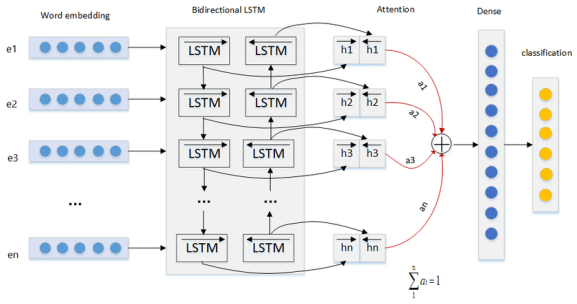

Figure 1: BiLSTM - Attention Architecture

4 Experiment

4.1 Experimental Setup

Dataset Our dataset comprises of around 140k tweets labelled with six standard human emotions - *disgust, anger, happiness, sadness, surprise and fear*.

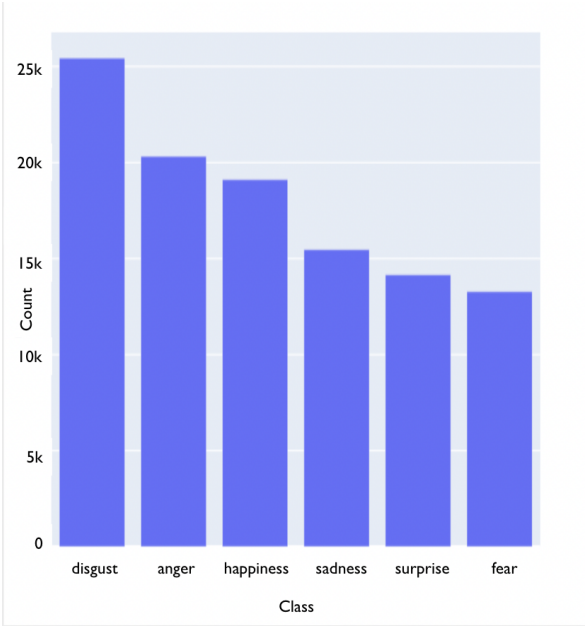

Figure 2: Emotion Classes in dataset and their counts

Baseline Methods A previous work (Wadhawan and Aggarwal, 2021) was used as baseline while building our models. The baseline model tried both word2vec and fasttext based embedding as inputs to CNN, LSTM, BiLSTM, BiLSTM with Attention models. They achieved higher accuracy while using fasttext as the embedding with 74.32%, 75.55%, 77.12% and 78.06% respectively.

Evaluation Protocols Used accuracy as the evaluation metrics to see the model's performance on the validation data-set. All the described models are compiled and trained multiple times over distinct hyper parameter configurations such as number of epochs, batch size and learning rate. We monitored the overall accuracy gain after each epoch is successfully completed and used early stopping to prevent over-fitting. As it is a multi-class problem we also focused on getting a good precision rate for each of the six classes.

4.2 Results and Discussion

We experimented with four different methodologies, i.e., CNN, LSTM, BiLSTM and BiLSTM with Attention. CNN model gave us an accuracy of 81% on the validation set. We observed that the validation loss started increasing as the number of epochs increased. This is a sign of overfitting the model. We tried LSTM and found the accuracy to be 83% on the validation set. In this case as well, we could see an uptick in the validation loss thus indicating overfitting.

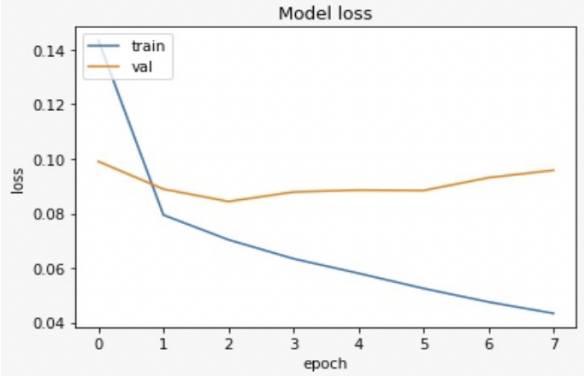

Figure 3: Train and Validation loss for CNN

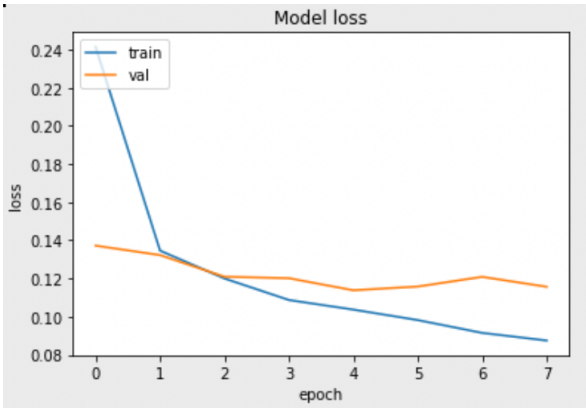

Figure 4: Train and Validation loss for LSTM

BiLSTM model proved to give us a satisfactory result with an accuracy of around 83%. We then added attention mechanism on top of BiLSTM which increased our accuracy to 86% on the validation set. Both The validation loss training loss are decreasing. This gave us a better insight into the tweet data. Compared to the reference paper's accuracy of around 72%, our implementation gave a better performance on the dataset.

Models	Accuracies
Attention-BiLSTM	86%
BiLSTM	83%
LSTM	83%
CNN	81%

Table 1: Accuracy Measures for each modeling library.

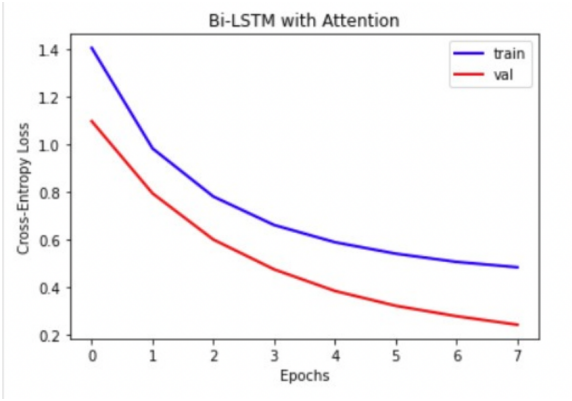

Figure 5: Train and Validation loss for Attention based BiLSTM

5 Conclusion

The results obtained from our models are satisfactory and provide a better insight on the dataset. Emotion detection on code-mixed data is a conundrum for a lot of social media sites to handle. It requires domain specific knowledge on the language we are dealing with.

Currently our code normalization is handled by a rule-based engine using regular expressions. In future, we would like to implement a transformer that could help us reduce the manual work and give better results on our normalized data. We also plan to extend our model to incorporate different languages apart from just Hindi and make it a general solution that can be used by many applications to solve problems that are currently a bit difficult to tackle.

Team responsibilities

We have effectively divided the tasks amongst ourselves. Data collection from Twitter and labelling was handled by Vishal, training FastText to obtain bilingual word embeddings was done by Shubha, preprocessing of the obtained text was carried out by Sruthi, construction and training of the CNN model was finished by Sruthi and Shubha, LSTM and BiLSTM models were built and trained by Joy, and BiLSTM model with attention was taken care of by Paresh. All of us carefully evaluated and compared the performance of the different models. Simultaneously all of us contributed equally to the research that was used to implement the project.

References

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. "I am borrowing ya mixing ?" an analysis of English-Hindi code mixing in Facebook. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 116–126, Doha, Qatar. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bharathi Raja Chakravarthi, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. 2020. A sentiment analysis dataset for code-mixed malayalam-english. *CoRR*, abs/2006.00210.

Bharat Gaind, Varun Syal, and Sneha Padgalwar. 2019. Emotion detection and analysis on social media. *CoRR*, abs/1901.08458.

Itisha Gupta and Nisheeth Joshi. 2017. Tweet normalization: A knowledge based approach. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pages 157–162.

Yash Kumar Lal, Vaibhav Kumar, Mrinal Dhar, Manish Shrivastava, and Philipp Koehn. 2019. De-mixing sentiment from code-mixed text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 371–377, Florence, Italy. Association for Computational Linguistics.

Jasy Suet Yan Liew and Howard R Turtle. 2016. Exploring fine-grained emotion detection in tweets. In *Proceedings of the NAACL Student Research Workshop*, pages 73–80.

Shashank Sharma. Linguistic resources. https://linguisticresources.weebly.com/about.html.

Erik Tromp and Mykola Pechenizkiy. 2014. Rule-based emotion detection on social media: Putting tweets on plutchik's wheel.

Anshul Wadhawan and Akshita Aggarwal. 2021. Towards emotion recognition in hindi-english code-mixed data: A transformer based approach. *CoRR*, abs/2102.09943.