gundapusunil at SemEval-2020 Task 9: Syntactic Semantic LSTM Architecture for SENTIment Analysis of Code-MIXed Data

Sunil Gundapu
Language Technologies Research Centre
KCIS, IIIT Hyderabad
Telangana, India
sunil.g@research.iiit.ac.in

Radhika Mamidi
Language Technologies Research Centre
KCIS, IIIT Hyderabad
Telangana, India
radhika.mamidi@iiit.ac.in

Abstract

The phenomenon of mixing the vocabulary and syntax of multiple languages within the same utterance is called Code-Mixing. This is more evident in multilingual societies. In this paper, we have developed a system for SemEval 2020: Task 9 on Sentiment Analysis for Code-Mixed Social Media Text. Our system first generates two types of embeddings for the social media text. In those, the first one is character level embeddings to encode the character level information and to handle the out-of-vocabulary entries and the second one is FastText word embeddings for capturing morphology and semantics. These two embeddings were passed to the LSTM network and the system outperformed the baseline model.

1 Introduction

Code-Mixing is a phenomenon which is evident in multilingual societies (Shana Poplack et al., 2003). It reflects the use of distinct grammatical systems and vocabulary of the languages being used simultaneously in a single utterance or conversation. This technique used in communication commonly is widely found today in popular social media platforms like Twitter, Facebook, Instagram in the form of posts, comments, replies, especially in chats. This is evident in multilingual societies like India, Canada, Ireland, South Africa, Switzerland, and many others.

India has officially recognized 22 regional languages[1]. So, in multilingual societies like India, most of the social media users predominantly integrate the well-known language, like English, with their native languages. 560 million Internet users[2] in India exchange information by mixing their regional languages with prominent language like English, which produces a huge amount of code-mixed social media corpus. One such trending combination is the mixing of Hindi and English with the output in **Hinglish**[3] (Hi-En) code-mixed data. Consider the example sentence which illustrates the code-mixing phenomenon being addressed in this paper. "*Congratulations/**Eng** Sir/**Eng** Ji/**Hin** Dobara/**Hin** PM/**Eng** banee/**Hin** ki/**Hin** hardik/**Hin** subhkamnaye/**Hin** aapko/**Hin** ./**O**"*. (Translation into English: Congratulations sir, Best wishes to become Prime Minister again). The words followed by language tags /Hin, /Eng, and /O correspond to Hindi, English and Other respectively.

Sentiment analysis of code-mixed data has become a prominent research area in recent times in the field of NLP. But identifying the sentiment in code-mixed data is hard since it poses the following challenges: (i) Romanized code-mixed data is noisy and ambiguous in nature. (ii) Accessible datasets are smaller in size to tune neural networks. (iii) Frequent occurence of non-standard spellings (such as pyaaarr, goooood). (iv) The phrase/word contractions (cmng for coming, IDK for I don't know). (v) Spelling variations. A single word pyaar (love), can be written as "piyar", "pyaarrrr", "peyar", "pyar", or "piyaar", etc. To handle these challenges we postulate that FastText embeddings enrich the word vectors with

---

[1]https://en.wikipedia.org/wiki/Languages_of_India
[2]https://www.internetworldstats.com/stats7.htm
[3]https://en.wikipedia.org/wiki/Hinglish

sub-word information and that character level embeddings should be able to assist deep learning models to handle unknown Hindi words.

In this paper, we propose models to predict the sentiment label of a given code-mixed tweet/text. The sentiment labels are positive, negative, or neutral and the code-mixed languages selected are Hindi and English. This task is conducted in **CodaLab**[4] website and our CodaLab username is **gundapusunil**. All our models are trained using only the trained dataset provided by SentiMix organizers. We started experiments with traditional machine learning models like Logistic Regression, Support Vector Machines but the f1-score on development set was below 0.63 then we moved to complex models like Long Short Term Memory (LSTM) based models with different types of word embeddings where we were able to conquer the baseline model.

Our paper is divided into the following sections: We begin with an introduction to code-mixing and its challenges in Section 1. Related work of various code-mixing strategies demonstrated in Section 2. Section 3 present the description of SentiMix dataset. We then discuss the pre-processing steps and compare machine learning and deep learning approaches with baseline model results in Section 4. The results are reported in Section 5, and Section 6 concludes the paper.

2 Related Work

Several studies have been made in the areas of sentiment analysis and code-mixed data. One of the earlier studies on code-mixed data was proposed by Gold (1967) with the goal of language identification in which it was stated that the structure of the language is procured by learning the language structure from the given text and informant. Braj B., Kachru (1976) described the structure of multilingual languages and language dependency in linguistic convergence of code-mixing from an Indian Perspective. SentiWordNet for English language introduced by Esuli and Sebastiani (2006), became the primary source for all sense based lexical analysis and opinion classification.

Later, researchers have extended the work on machine learning and sentiment analysis methods for Indian languages. In their study Siersdorfer, Chelaru et al. (2010) used the SVM and Naıve Bayes classifiers to label millions of comments for sentiment polarity. R. Sharma and P. Bhattacharyya (2014) developed a lexicon-based sentiment analyzer for product reviews and the same subjective lexicon-based model has been extended to Punjabi language. For Malayalam movie reviews D. S. Nair, J. P. Jayan et al. (2014, 2015) initially came up with a rule based system for sentiment analysis, later improved the system with the support of the machine learning algorithms. MIKE 2015 sentiment analysis task for Hindi and Bengali used the Multinomial Naive Bayes classifier with the features for building the vector space constrained by filtering the words based on WordNet.

In recent years, researchers have seen a huge improvement in the task of sentiment analysis of English as well as Hinglish using deep neural networks. M.G. Jhanwar, A. Das (2018) proposed an ensemble of character-trigrams based LSTM model and a n-grams based Multinomial Naive Bayes model to classify the sentiments of Hinglish code-mixed data. Shalini K, Barathi Ganesh HB et al. (2018) addressed the performance of distributed representation methods in sentiment analysis and reported comparisons among different machine learning and deep learning techniques. Other attempts include using sub-word level compositions with LSTMs to capture sentiment at the morpheme level (Joshi et al., 2016). We attempted to perform SemEval 2020 task-9 (Patwa et al., 2020) with various classification and deep learning models to analyze the results and also how such models contributed to a great advance in this task.

3 Dataset

In this paper, we used the dataset provided by Task 3: SentiMix in SemEval 2020. The corpus contains a total of 20000 tweets and it is sub-divided into three sets (train, validation and test). Each corpus except test contains code-mixed tweets along with their corresponding sentiment labels. These code-mixed tweets are tokenized into tokens. And the tokens of each tweet are separated by a new line. Each token is manually annotated with a language identification tag which are: Hindi (Hin), English (Eng), Other (O).

---

[4]https://competitions.codalab.org/competitions/20654

Dataset	Positive	Negative	Neutral	Total
Train	4634	4102	5264	14000
Valid	982	890	1128	3000
Test	1000	900	1100	3000
				20000

Table 1: Dataset Statistics.

Table 1 shows the distribution of sentiment classes in the SentiMix dataset. Table 2 shows some examples of code-mixed tweets from the SentiMix dataset. Here, the first column contains Hinglish tweet, the second column contains English translation of tweet, the third column contains sentiment label.

Code-Mixed Tweet	Approximate English Translation	Sentiment Label
All the best Team India Jeet ke aana	Team India all the best, come back with win.	Positive
Aap bhi aisa drama kr do kam se km	You also do this drama and at least.	Negative
Aisa PM naa hua hai aur naa hee hoga	Neither there has been a PM like him, nor there will be	Positive

Table 2: Example Code-Mixed Tweets.

4   System Architecture

In this section, we present our models that are trained and validated on the SentiMix dataset described in the previous section. We compare our approach with Machine Learning and Neural Network based baselines. The full code of system architecture can be found at GitHub[5].

4.1   Preprocessing of Code-Mixed tweets

Our code-mixed data consists of excessive noise in the form of punctuations, Uniform Resource Locator (URL's), few Devanagari script Hindi words, twitter mentions, hashtags, stop words, etc. In the preprocessing step, we take a stab to overcome the noise in the the data by remove/normalize the unnecessary tokens. Figure 1 explains the code-mixed dataset pre-processing pipeline. The input for the pipeline is a tokenized tweet and the output is a cleaned tweet.
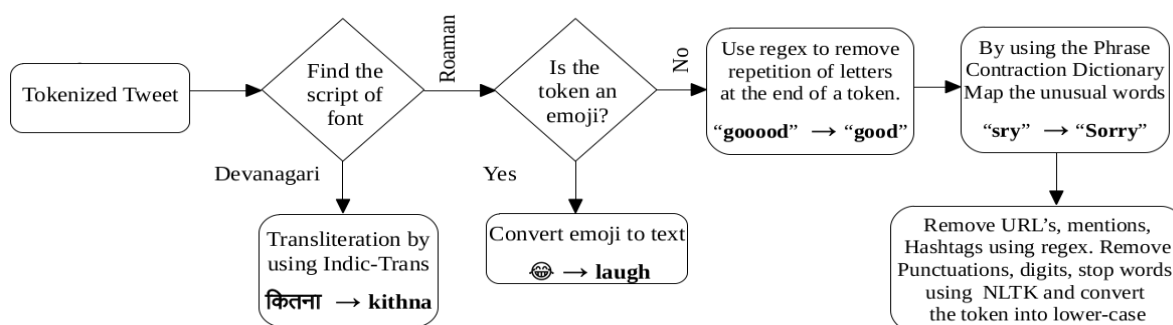

Figure 1: Code-Mixed text preprocessing pipeline.

4.2   Classical Supervised Machine Learning Algorithms

To design the finest system for sentiment analysis in code-mixed data, we begin our experiments with traditional machine learning algorithms like Support Vector Machines (SVM), One-vs-rest classifier with Logistic Regression (OvRLR), Random Forest Classifier (RFC) and Multilayer Perceptron (MLP). The input for these methods is a single d dimensional feature vector of a single code-mixed tweet.

[5]https://github.com/SunilGundapu/SENTIment-Analysis-on-Hindi-English-Code-MIXed-Data

We analyze the results of the above classical algorithms with the combination of two types of vectors. (i) Word level term frequency-inverse document frequency (tf -idf) vector and (ii) Glove word embeddings. For all tokens in the code-mixed tweet a feature vector is created by averaging over d dimensional Glove embeddings and also it is experimented with tf-idf weighted averaging. The code-mixed tweet vector construction scheme is described below:

$$featureVector_{tweet} = \frac{\sum_{n=1}^{N} \textbf{tf-idf}(token_i) \times \textbf{Glove}(token_i)}{total\ no.\ of\ tokens\ in\ tweet\ (N)}$$

Empirically, we found that standard averaging of Glove and tf-idf gave better results than normal tf-idf weighted averaging.

4.3 Deep Neural Networks

In this subsection, we describe the character and word embedding based deep neural network called "Syntactic and Semantic LSTM (SS-LSTM)" that gave better predictions on our corpus. Initially, We tried with Word2Vec (T. Mikolov et al., 2013), GloVe (J. Pennington et al., 2014), FastText (Bojanowski et al., 2016), Character embeddings for each word in the input code-mixed tweet. We train a simple LSTM model using each of these embeddings to test the effectiveness of these embeddings for sentiment classification. FastText and Character level embeddings gave slightly better results than other embeddings. By considering these results we modeled the SS-LSTM architecture given below.

4.3.1 Character Level Embeddings

Character level embeddings use a one-dimensional convolutional neural network (1D-CNN) to find the numeric representation of words by looking at their character-level compositions. 1D-CNN (Yoon Kim, 2014) is an algorithm capable of handling unseen words and also extracting syntactic information from the segments of input. The character embedding step converts tweet tokens into a $d \times T$ matrix. d is the dimension of vector and T is the number of tokens in code-mixed tweet.

4.3.2 Word Level Embeddings

For word level embeddings we used the Facebook's FastText. The main advantage of FastText embeddings is to capture the hidden knowledge about a language, like word analogies or semantic. And it is looking into the internal structure of words, which could be very useful for morphologically rich languages like Hindi. The FastText enrich word vectors with subword information.
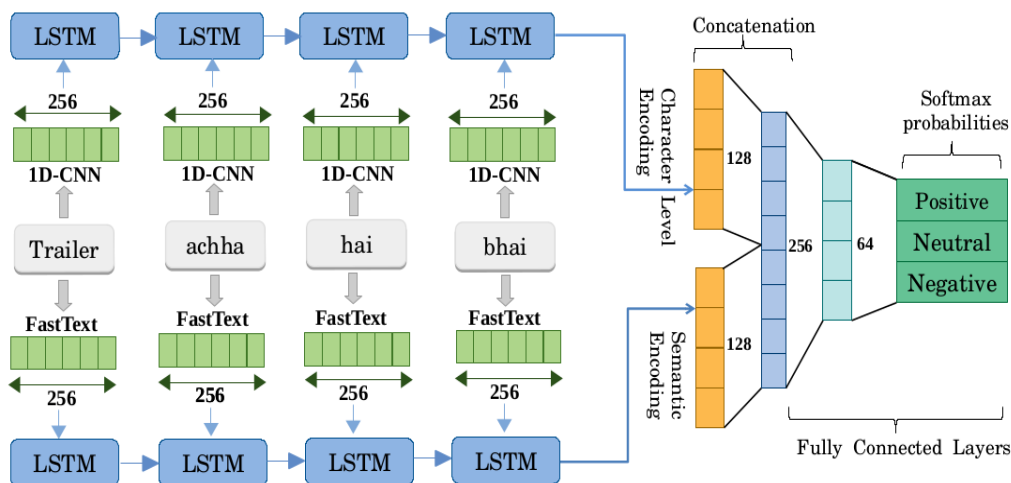
4.3.3 Model Architecture

Figure 2: System (SS-LSTM) Architecture.

1250

We model the task of SentiMix as a multi-class classification problem where given a code-mixed tweet, the model outputs probabilities of it belonging to three output classes - Positive, Negative, and Neutral. The proposed system architecture (SS-LSTM) is shown in Figure 2. The input tweet is fed into 1D-CNN and FastText. These two word embedding models generate two $d \times T$ matrices, one is for the 1D-CNN and the other for the FastText. Here the dimension of each matrix is $256 \times T$. These two word embedding matrices are passed to two LSTM layers. One LSTM layer uses a character level embeddings, whereas the other layer uses a FastText word embeddings. These two layers learn syntactic and semantic feature representation and encode sequential patterns in the tweet. And each LSTM layer gives a 128×1 dimensions vector.

The character embedding LSTM layer output is $C \in R^{128 \times 1}$ and the FastText word embedding LSTM layer output is $W \in R^{128 \times 1}$. These two output feature representations are row-wise concatenated and the output vector dimension is $O \in R^{256 \times 1}$. The output vector O passed to a fully connected network with one hidden layer which models interactions between these features and outputs probabilities per sentiment class. We used **Keras** neural network library to implement this model.

5 Results

A summary of results from various techniques on the SentiMix test dataset is present in Table 3. SS-LSTM gave the best performance on f1-score for each sentiment class as well as on average f1-score. Our results thus indicate that combining syntactic and semantic representations in SS-LSTM outperforms individual LSTM-Character and LSTM-FastText embedding models.

Model	Representations	Precision	Recall	f1-Score
SVM	TF-IDF avg	0.6315	0.6373	0.6308
RFC	TF-IDF avg	0.6260	0.6323	0.6261
OvRLR	TF-IDF avg	0.6391	0.6426	0.6377
MLP	TF-IDF avg	0.6505	0.6433	0.6454
SVM	TF-IDF and Glove avg	0.6409	0.6384	0.6412
RFC	TF-IDF and Glove avg	0.6311	0.6358	0.6356
OvRLR	TF-IDF and Glove avg	0.6357	0.6393	0.6343
MLP	TF-IDF and Glove avg	0.6485	0.6523	0.6445
LSTM	Word2Vec	0.6565	0.6538	0.6523
LSTM	FastText	0.6605	0.6533	0.6554
LSTM	Glove	0.6585	0.6523	0.6545
LSTM	1D-CNN	**0.6825**	0.6610	0.6651
LSTM	BERT	0.6800	0.6712	0.6742
SS-LSTM	FastText and 1D-CNN	0.6819	**0.6773**	**0.6758**

Table 3: Results on test data for Hindi-English.

Machine learning models with tf-idf feature representations gave the approximate baseline results. We observe that the tf-idf weighted average of GloVe performed better than the simple average of vectors. And we used the grid search (Srivastava et al., 2014) to find the better hyper-parameters like number of LSTM layers, learning rate, and the number of epochs. We used GPU for training deep learning models.

6 Conclusion

In this paper, we experimented the code-mixed dataset with various machine learning and deep learning models. We see that the LSTM models performed far better than traditional ML methods. In the first phase of the SentiMix competition (development set), we were able to achieve a score of **0.6357**. But in the second phase (test dataset), our best score was only **0.6758**. After competition we attain the f1-score of **0.6789** by changing few parameters like learning rate and number of LSTM's. Till now we handled problems like unseen words, spelling variations, dataset imbalance, emojis, short form of words, etc.

In future work, we plan to focus on issues like free ordering of words in sentence constructions, short sentences with unclear semantic structure, etc. And we would like to explore more deep neural network architectures that can capture sentiments in code-mixed data.

References

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, December. Association for Computational Linguistics.

Poplack, Shana and Walker, James. 2003. Pieter Muysken, Bilingual speech: a typology of code-mixing. Cambridge: Cambridge University Press, Pp. xvi+306.. Journal of Linguistics. 39. 678 - 683. 10.1017/S0022226703272297.

E. M. Gold and T. R. Corporation. 1967. *Language identification in the limit*, Inf. Control, vol 10, no 5, pp 447–474.

Braj B. Kachru. 1978. Toward Structuring Code-Mixing: An Indian Perspective. *International Journal of the Sociology of Language*, 16:27-46.

A. Esuli, F. Sebastiani, and V. G. Moruzzi. 2006. *SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining*, Proc. Lr, vol 0, pp 417–422.

S. Siersdorfer, S. Chelaru, W. Nejdl, and J. San Pedro. 2011. How useful are your comments?. In *Proceedings of 19th International Conference on World Wide Web 2011*, vol 15, pp 891–900.

R. Sharma and P. Bhattacharyya. 2014. A sentiment analyzer for hindi using hindi senti lexicon. In *Proceedings of 11th International Conference on Natural Language Processing 2014*, p 150.

K. Shalini, H. B. Ganesh, M. A. Kumar and K. P. Soman. 2018. Sentiment Analysis for Code-Mixed Indian Social Media Text With Distributed Representation. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, pp 1126-1131.

D. S. Nair, J. P. Jayan, E. Sherly. 2015. Sentima-sentiment extraction for malayalam. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on. IEEE*, pp 2381-2384.

K. Sarkar and S. Chakraborty. 2015. A sentiment analysis system for indian language tweets. In *International Conference on Mining Intelligence and Knowledge Exploration.*, Springer, pp 694-702.

Madan G. Jhanwar and Arpita Das. 2018. *An Ensemble Model for Sentiment Analysis of Hindi-English Code-Mixed Data*, CoRR, vol abs/1806.04450.

A. Joshi, A. Prabhu, M. Shrivastava, V. Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp 2482-2491.

Patra, Braja and Das, Dipankar and Das, Amitava. 2018. Sentiment Analysis of Code-Mixed Indian Languages: An Overview of SAIL Code-Mixed Shared Task @ICON-2017.

Gupta, Umang Chatterjee, Ankush Srikanth, Radhakrishnan Agrawal, Puneet. 2017. Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. 2016. Bag of tricks for efficient text classification. In *arXiv preprint arXiv:1607.01759*.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp 3111-3119.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, vol 14, pp 1532–1543.

A. Joshi, A. Prabhu, M. Shrivastava, V. Varma. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp 1746-1751.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. *Dropout: a simple way to prevent neural networks from overfitting*, 15(1):1929–1958.