IIT Gandhinagar at SemEval-2020 Task 9: Code-Mixed Sentiment Classification Using Candidate Sentence Generation and Selection

Vivek Srivastava, Mayank Singh
Indian Institute of Technology Gandhinagar
Gujarat India
{vivek.srivastava, singh.mayank}@iitgn.ac.in

Abstract

Code-mixing is the phenomenon of using multiple languages in the same utterance of a text or speech. It is a frequently used pattern of communication on various platforms such as social media sites, online gaming, product reviews, etc. Sentiment analysis of the monolingual text is a well-studied task. Code-mixing adds to the challenge of analyzing the sentiment of the text due to the non-standard writing style. We present a candidate sentence generation and selection based approach on top of the Bi-LSTM based neural classifier to classify the Hinglish code-mixed text into one of the three sentiment classes positive, negative, or neutral. The proposed approach shows an improvement in the system performance as compared to the Bi-LSTM based neural classifier. The results present an opportunity to understand various other nuances of code-mixing in the textual data, such as humor-detection, intent classification, etc.

1 Introduction

Code mixing is one of the most frequent styles of communication in multilingual communities, such as India. This pattern of communication on various platforms such as social media, online gaming, online product reviews, etc. makes it difficult to understand the sentiment of the text. Sentiment classification of the code-mixed text is useful in the scenarios of socially or politically driven discussions, fake news propagation, etc. Some of the major challenges with the text in the code-mixed language are:

- **Ambiguity in language identification**: *is, me, to* are some examples of the words that are ambiguous to classify as English and Hindi without proper knowledge of context.
- **Spelling variations**: E.g., *jaldi, jldi, jldiii,..* are some variations for the word *hurry* in English.
- **Misplaced/ skipped punctuation**: E.g., *Aap kb se cricket khelne lage..never saw u bfr.*
  The sentence in the example misses a question mark(?) apart from other necessary modifications to make the structure of the sentence correct.
- **Missing context**: E.g., *Note kr lijiye.. Bandi chal rahi h ;)* is a code-mixed sentence and demonetisation (notebandi) is the hidden context.

With the increasing popularity of using code-mixing on social media platforms, the interest to study the various dynamics of code-mixing is seeking a boom. Multiple works on language identification (Barman et al., 2014; Das and Gambäck, 2014), POS tagging (Vyas et al., 2014; Ghosh et al., 2016), named entity recognition (Singh et al., 2018a; Singh et al., 2018b), etc. shows the challenges and the opportunities with the code-mixed data. Pang et al. (2008) presents a survey of the approaches to understand the opinions and sentiments on various platforms. Dos Santos and Gatti (2014) performs the sentiment analysis task of the short text messages on two corpora from different domains and present their findings. Kouloumpis et al. (2011) presents multiple experiments to understand the sentiment of Twitter messages using linguistic features and lexical resources. Sentiment analysis of the code-mixed Tweets using a sub-word level representation (Prabhu and Verma, 2016) in the LSTM can improve the performance of the system. Swami

et al. (2018) presents a corpus to understand and detect the sarcasm of 5250 code-mixed English-Hindi tweets.

Contributions: We present a candidate sentence generation and selection based procedure on top of the Bi-LSTM neural classifier. We observe the increase in the system performance by using the proposed architecture as compared to the Bi-LSTM classifier.

2 Dataset

We use the dataset (Patwa et al., 2020) provided by the task organizers for building our system (Codalab username: vivek_IITGN). Each sentence in the dataset has a sentiment label as positive, negative, or neutral. Table 1 shows the distribution of the sentences in the train, validation, and test dataset for each class. We have 15131, 3000, and 3000 sentences in train, validation, and test set, respectively. On manual inspection of the dataset, we observe ambiguity in the annotation of the sentences. To examine this further, we extract the top 20 most frequently used words in the dataset. We remove the English stopwords, and we set a threshold of 4 characters on the length of the tokens to filter out the Romanized Hindi stopwords. Table 2 shows the percentage overlap of most frequent 20 words of length more than four characters in the train, validation, and test set. The high percentage overlap of most frequent neutral words with positive and negative words also indicates the presence of ambiguity as a challenge in the annotation. Ambiguity in the label for the sentence is one of the major challenges for understanding the sentiment of the sentence. Figure 1 shows some of the example sentences in training set with ambiguous sentiment label. There could be multiple reasons for the ambiguity in the annotation of the sentences such as hidden sarcasm, targeting individual or institution, unclear intent, etc. It leads to human bias due to the annotator's perception of the event or the individual in the sentence. To preprocess the dataset, we remove hyperlinks, mentions, hashtags, emoticons, and special characters from the sentences. We lowercase the sentences. To identify and remove the emoticons from the sentences, we use the emoji sentiment dataset[1].

	Train	Validation	Test
Positive Sentences	5034	982	1000
Negative Sentences	4459	890	900
Neutral Sentences	5638	1128	1100

Table 1: Distribution of the sentences in the train, validation, and test set.

	Training Set			Validation Set			Test Set		
	Positive	**Negative**	**Neutral**	**Positive**	**Negative**	**Neutral**	**Positive**	**Negative**	**Neutral**
Positive	-	20	55	-	20	50	-	15	40
Negative	20	-	55	20	-	45	15	-	40
Neutral	55	55	-	50	45	-	40	40	-

Table 2: Percentage overlap of the most frequent 20 words of length more than 4 characters in the training, validation, and test set.

3 Experiments

The availability of code-mixed embedding is a challenging task due to the scarcity of large scale code-mixed corpora. We are using the Glove embedding (Pennington et al., 2014) for the English words, and we train the embedding on the PHINC dataset (Srivastava and Singh, 2020) for the Romanized Hindi words. We are using the code-mixed sentences from PHINC to train the code-mixed embedding.

Initially, we train the system using Bi-LSTM based neural architecture. The architecture of the Bi-LSTM classifier has the embedding layer followed by the Bi-LSTM layer and then two dense layers and, finally, the softmax prediction for the three sentiment classes. For prediction on the test set, we pre-filter

---

[1]https://www.kaggle.com/thomasseleck/emoji-sentiment-data

Figure 1: Example sentences from the training set with ambiguous labels.

the sentence based on the list of abusive words. If a sentence contains any words from this list, we label that sentence as negative. In the pre-filtering process, we identify 123 sentences in the test set containing one or more of the abusive words from the list. Post pre-filtering step, we generate 15 candidate sentences for each of the remaining test instance using the Candidate Sentence Generation (CSG) procedure. We then select the best sentiment prediction for the sentence using the Candidate Sentence Selection (CSS) procedure. Algorithm 1 shows the CSG procedure. Algorithm 2 shows the CSS procedure. Figure 2 shows the flow diagram of the proposed approach. In the CSG procedure, we try to confuse the model with nearly similar sentences with additional phrases. We generate five similar sentences to the original code-mixed sentence for each of the three buckets (positive, negative, and neutral). We detect the degree of confusion in the sentiment prediction using the CSS procedure. We also keep track of the degree of confusion by sentences in each bucket. If the degree of confusion is significantly higher, we change the previous prediction by the model using the rules (as discussed in Algorithm 2).
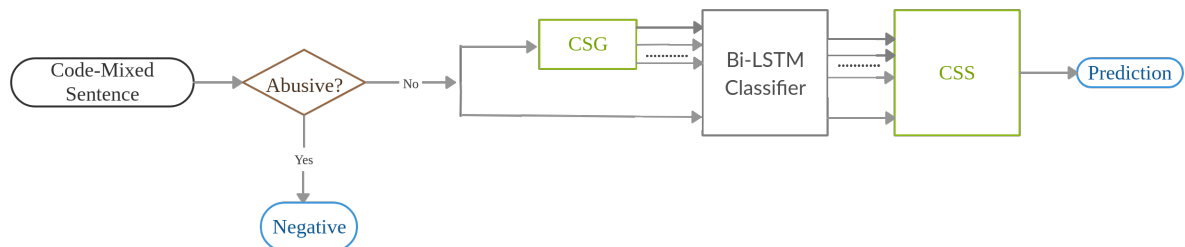
Figure 2: Flow diagram of the proposed approach.

	Phrases
Positive	good enough, sure thing, undoubtedly, theek hai, even so, indubitably, tathaastu, of course, savaida, certainly, sakaaraatmak, gladly, affirmative, apanee marjee, abhee to, sabase adhik aashvast, amen, good, yakeenan, bahut achchha, theek, precisely, by all means, beyond a doubt, surely, yeah, unquestionably, very well, exactly, positively, khushee se, har tarah se, most assuredly, definitely, achchha
Negative	nowhere, nahin kar sakate, koee bhee nahin, nahin, wouldn't, won't, nahin hai, nahin karana chaahie, nahin kiya ja saka, kabhee naheen, never, don't, neither, couldn't, nothing, doesn't, koee nahin, barely, mushkil se, kuchh bhee to nahin, wasn't, shouldn't, scarcely, nahin karata hai, hardly, nahin hoga, kaheen bhee nahin, no, not, na, nobody, can't, shaayad hee, no one, none

Table 3: List of positive and negative phrases.

4 Results and Analysis

To evaluate the system performance, we use accuracy, precision, recall, and f-score as the evaluation metric. Table 4 shows the distribution of the successfully and unsuccessfully modified sentences for the

---
Algorithm 1 Candidate Sentence Generation (CSG) procedure

1: **procedure** CSG(CM_sent_i)
2:       Load the set of positive and negative phrases P. Table 3 shows the set P.
3:       Set $bucket_1$=[], $bucket_2$=[], and $bucket_3$=[]
4:       **for** i=1→5 **do**
5:           Create a set $spots_{pos}$={$sp_1, sp_2, ...$} of randomly selected spots in CM_sent_i
6:           Create a set $spots_{neg}$={$sn_1, sn_2, ...$} of randomly selected spots in CM_sent_i
7:           Create a set $spots_{neu}$={$su_1, su_2, ...$} of randomly selected spots in CM_sent_i
8:           Set $sent_{pos}$=CM_sent_i, $sent_{neg}$=CM_sent_i, and $sent_{neu}$=CM_sent_i
9:           **for** each spot sp_j in $spots_{pos}$ **do**
10:               Randomly select a positive phrase p_{pos} from the set P
11:               Replace the phrase p_{pos} at spot sp_j in $sent_{pos}$
12:           Add the new sentence $sent_{pos}$ in the list $bucket_1$
13:           **for** each spot sn_j in $spots_{neg}$ **do**
14:               Randomly select a negative phrase p_{neg} from the set P
15:               Replace the phrase p_{neg} at spot sn_j in $sent_{neg}$
16:           Add the new sentence $sent_{neg}$ in the list $bucket_2$
17:           **for** each spot su_j in $spots_{neu}$ **do**
18:               Alternatively select phrase p_{pos} and p_{neg} from the set P
19:               Replace alternatively the phrase p_{pos} and p_{neg} at spot su_j in $sent_{neu}$
20:           Add the new sentence $sent_{neu}$ in the list $bucket_3$
21:       Return $bucket_1$, $bucket_2$, and $bucket_3$

Algorithm 2 Candidate Sentence Selection (CSS) procedure

1: **procedure** CSG($pred\_sent_i$, $pred\_bucket_1$, $pred\_bucket_2$, $pred\_bucket_3$)
2:       Set $pred_{final}$=[]
3:       **if** $pred\_sent_i$ is *Positive* **then**
4:           **if** most frequent prediction in $pred\_bucket_1$ is *Positive* **then**
5:               Set $pred_{final}$= *Positive*
6:           **else**
7:               Set $pred_{final}$= most frequent prediction in $pred\_bucket_2$ and $pred\_bucket_3$
8:       **else if** $pred\_sent_i$ is *Negative* **then**
9:           **if** most frequent prediction in $pred\_bucket_2$ is *Negative* **then**
10:               Set $pred_{final}$= *Negative*
11:           **else**
12:               Set $pred_{final}$= most frequent prediction in $pred\_bucket_1$ and $pred\_bucket_3$
13:       **else**
14:           **if** most frequent prediction in $pred\_bucket_3$ is *Neutral* **then**
15:               Set $pred_{final}$= *Neutral*
16:           **else**
17:               Set $pred_{final}$= most frequent prediction in $pred\_bucket_1$ and $pred\_bucket_2$
18:       Return $pred_{final}$

> CODE-MIXED PRE-PROCESSED SENTENCE: rohit bhai i am your big fan want to meet you
> SENTENCE WITH SPOTS (MARKED AS <SPOT>): rohit bhai i am <spot> your big fan <spot> want to meet you
> SENTENCE IN BUCKET 1: rohit bhai i am *undoubtedly* your big fan *by all means* want to meet you
> SENTENCE IN BUCKET 2: rohit bhai i am *scarcely* your big fan *shaayad hee* want to meet you
> SENTENCE IN BUCKET 3: rohit bhai i am *positively* your big fan *kaheen bhee nahin* want to meet you

Figure 3: Example CSG procedure for sentences in all the three buckets. We select phrases in *italics* at random from the list of positive and negative phrases and replace with <spot> as per the rule for each bucket.

final prediction by the Bi-LSTM + CSG + CSS model. We use the prediction by the Bi-LSTM classifier as the baseline. We observe relatively better successful modifications for the neutral sentences to and from the positive and negative sentences. This result can be attributed to the high overlap in the most frequent words in the neutral sentences with both the other classes (as discussed in section 2). Table 5 shows the system performance of the two models on the test dataset. We observe an increase in the system performance with the use of CSG and CSS procedures on top of the Bi-LSTM classifier. Table 6 shows the system performance on the test dataset with the classwise F-score as the evaluation metric.

	Successful modification			Unsuccessful modification		
	Positive	**Negative**	**Neutral**	**Positive**	**Negative**	**Neutral**
Positive	-	10	77	-	20	96
Negative	7	-	120	34	-	149
Neutral	189	95	-	153	100	-

Table 4: Distribution of the successful and unsuccessful modification of the test sentences by the Bi-LSTM + CSG + CSS model. We use Bi-LSTM classifier as the baseline. Prediction labels in the rows shows the prediction by the Bi-LSTM classifier whereas the prediction labels in the columns are for the Bi-LSTM + CSG + CSS model.

	Accuracy	Precision	Recall	F-score
Bi-LSTM	0.587	0.607	0.590	0.595
Bi-LSTM + CSG + CSS	0.608	0.618	0.609	0.612

Table 5: System performance on the test dataset. We use the macro score for evaluation.

	Positive	Negative	Neutral
Bi-LSTM	0.621	0.643	0.520
Bi-LSTM + CSG + CSS	0.754	0.625	0.459

Table 6: Evaluation of the system performance based on classwise F-score.

5   Conclusion and Future Work

We present a Bi-LSTM based sentiment classifier for the classification of code-mixed Hinglish sentences. We also propose a candidate sentence generation and selection based approach on top of the Bi-LSTM based classifier to improve the system performance. Up to a certain extent, the proposed approach is able to detect the ambiguous labels in the dataset. We can extend the proposed method to solve other challenges relevant to code-mixing, such as sarcasm detection, fake-news identification, intent classification, etc.

References

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 13–23, Doha, Qatar, October. Association for Computational Linguistics.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387.

Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 90–97.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Fifth International AAAI conference on weblogs and social media*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Joshi Aditya Shrivastava Manish Prabhu, Ameya and Vasudeva Verma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. *arXiv preprint arXiv:1611.00472*.

Kushagra Singh, Indira Sen, and Ponnurangam Kumaraguru. 2018a. Language identification and named entity recognition in hinglish code mixed tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 52–58.

Vinay Singh, Deepanshu Vijay, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018b. Named entity recognition for hindi-english code-mixed social media text. In *Proceedings of the Seventh Named Entities Workshop*, pages 27–35.

Vivek Srivastava and Mayank Singh. 2020. Phinc: A parallel hinglish social media code-mixed corpus for machine translation. *arXiv preprint arXiv:2004.09447*.

Sahil Swami, Ankush Khandelwal, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A corpus of english-hindi code-mixed tweets for sarcasm detection. *arXiv preprint arXiv:1805.11869*.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.