IIITG-ADBU at SemEval-2020 Task 9: SVM for Sentiment Analysis of English-Hindi Code-Mixed Text

Arup Baruah[1], Kaushik Amar Das[1], Ferdous Ahmed Barbhuiya[1], and Kuntal Dey[2]*
[1]IIIT Guwahati, India
[2]Accenture Technology Labs, Bangalore
arup.baruah@gmail.com, kaushikamardas@gmail.com,
ferdous@iiitg.ac.in, kuntal.dey@accenture.com

Abstract

In this paper, we present the results that the team IIITG-ADBU (codalab username 'abaruah') obtained in the SentiMix task (Task 9) of the International Workshop on Semantic Evaluation 2020 (SemEval 2020). This task required the detection of sentiment in code-mixed Hindi-English tweets. Broadly, we performed two sets of experiments for this task. The first experiment was performed using the multilingual BERT classifier and the second set of experiments was performed using SVM classifiers. The character-based SVM classifier obtained the best F1 score of 0.678 in the test set with a rank of 21 among 62 participants. The performance of the multilingual BERT classifier was quite comparable with the SVM classifier on the development set. However, on the test set it obtained an F1 score of 0.342.

1 Introduction

Sentiment analysis has been defined as the computational study of opinions, sentiments, and emotions expressed in the text (Liu, 2010). In its basic form, sentiment analysis is used to determine the polarity of a given text where the polarity may be *negative*, *neutral*, and *positive*. Thus, sentiment analysis can be viewed as a text classification problem.

With the advent of social media platforms, the use of non-standard language has increased. Now-a-days, it is very common to use emoticons, mentions, acronyms, and ungrammatical sentences while communicating in social media. All these factors make the traditional tools used for natural language processing fail on social media text. Another new style of communication in social media is the use of code-mixed text. Code-mixing means the mixing of words from more than one language in the same sentence or between sentences. Code mixing makes the task of sentiment analysis more challenging.

The objective of *SentiMix* (Task 9), organized as part of the International Workshop on Semantic Evaluation 2020 (SemEval 2020), is to detect the sentiment of code-mixed tweets (Patwa et al., 2020). This task was a three-way classification problem with the labels being *negative*, *neutral*, and *positive*. The task was held for both Hindi-English and Spanish-English code-mixed tweets.

We participated in this task for the Hindi-English language. In this task, we experimented with SVM and multilingual BERT classifiers.

2 Related Work

Joshi et al. (2010) performed the first work on the detection of sentiment analysis of Hindi text. In their work, unigram and bigram based SVM classifiers were used. As another approach, the Hindi text was machine translated to English and the translated text was then classified using a classifier trained on English text. This work also led to the creation of Hindi-SentiWordNet sentiment lexicon. The lexicon was also used to perform a lexicon-based sentiment analysis. The classifier trained on the Hindi text performed the best with an accuracy of 78.14%. Sharma et al. (2015) used a lexicon-based approach to determine the

*This work was done when the author was affiliated with IBM Research India, New Delhi

Negative	533 (29%)
Neutral	754 (40%)
Positive	582 (31%)
Total	1869

Table 1: Trial

Negative	4102 (29%)
Neutral	5264 (38%)
Positive	4634 (33%)
Total	14000

Table 2: Train

Negative	890 (30%)
Neutral	1128 (38%)
Positive	982 (32%)
Total	3000

Table 3: Development

Negative	4992 (29%)
Neutral	6392 (38%)
Positive	5616 (33%)
Total	17000

Table 4: Combined

sentiment of code-mixed Hindi-English text. The language of each word was first determined. The Hindi words written using Roman scripts were then transliterated into Devanagari script. The sentiment of each word was then determined through a lookup of the lexicons - Hindi SentiWordNet, Opinion Lexicon, and AFINN. The sentiment of the text was then determined based on the count of positive and negative words. Joshi et al. (2016) also worked on detecting sentiment on code-mixed Hindi-English text. A sub-word level LSTM was used in their study. The sub-word level representations were generated by performing a convolution operation on 128-dimensional character embeddings. The sub-word level LSTM was found to perform better than character based LSTM.

3 Data Set

The Hindi-English data set for this task was provided in the CoNLL format. For each tweet, the following information was provided: (1) the id for the tweet, (2) its label (negative, neutral, or positive), and (3) the language id of each token (HIN for Hindi, ENG for English, and O if neither Hindi nor English). In our experiments, we did not make use of the language id information provided in the data set.

Tables 1 to 4 show the statistics of the trial, train, development, and the combined data sets respectively. The combined data set was obtained by combining the trial, train, and development data sets. The duplicate entries were removed from the combined data set. The combined data set was used in our experiments. As can be seen from the tables, the data sets were quite balanced.

4 Methodology

4.1 Preprocessing

In our work, before performing tokenization, the text was converted to lower case. This conversion to lower-case was performed through the BERT tokenizer and the TFIDF vectorizer. In one of the experiments, the URLs were removed from the text. Emoticons, hashtags, and mentions were not removed from the text.

4.2 Classifiers

4.2.1 Multilingual BERT

BERT (Devlin et al., 2019) is a bi-directional model based on the transformer architecture. The transformer architecture is an architecture based solely on attention mechanism (Vaswani et al., 2017). The transformer architecture overcomes the inherent sequential nature of Recurrent Neural Networks (RNN) and hence they are more conducive for parallelization.

Multilingual BERT is BERT trained for multilingual tasks. It was trained on monolingual Wikipedia articles of 104 different languages. It is intended to enable multilingual BERT fine-tuned in one language to make predictions for another language. In our study, we used the multilingual BERT model having 12 layers and 12 heads [1]. This model generates a 768-dimensional vector for each word. We used the 768-dimensional vector of the Extract layer as the representation of the tweet. Our classification layer consisted of a single Dense layer. The dense layer consisted of 3 units and the *softmax* activation function

[1] `https://github.com/google-research/bert`

was used. The loss function used was *sparse categorical crossentropy*. The *Adam* optimizer with a learning rate of 2e-5 was used for training the model. The model was trained for 15 epochs. Early stopping with patience of 5 was used and *Sparse categorical accuracy* was monitored for early stopping.

4.2.2 SVM

We also used the Support Vector Machine (SVM) model for this task. The SVM implementation provided by Scikit-learn library (Pedregosa et al., 2011) was used in our experiments. The SVM model was trained using TF-IDF features of word and character n-grams. TF-IDF of n-grams in a document is calculated by multiplying the *term frequency* (Luhn, 1958) of the n-gram in the document with the *inverse document frequency* (Jones, 2004) of the n-gram. The *term frequency* of an n-gram in a document is the count of the number of times the n-gram appears in the document. The count may be normalized by dividing the count with the total number of n-grams in the document. The *inverse document frequency* of an n-gram *t* is calculated as $log(N/N_t)$, where N is the total number of documents and N_t is the number of documents in which the n-gram *t* appears. Word n-grams of size 1 to 3 and character n-grams of size 1 to 6 were used in our study. The *linear* kernel was used for the classifier and hyperparameter C was set to 1.0. The hyperparameter C is the regularization parameter. Larger values for C leads to a narrower margin and less misclassified instances. However, C should be set to a smaller value to reduce overfitting. We experimented using the SVM model on both the uncleaned data (SVM Run 1) and on the cleaned data where the URLs were removed from the text (SVM Run 2).

5 Results

Table 5 shows the results of our classifier obtained on the development set. As was mentioned in section 3, we combined the trial, train, and dev data sets. 20% of this combined data set was used as the development set. Run 2 of the SVM classifier was on a cleaned data set where the URLs were removed. The uncleaned data set was used for BERT and run 1 of the SVM classifier. As can be seen from the table, the best score was obtained by the SVM classifier when the cleaned data set was used. SVM trained on the character n-grams performed better than those trained on word n-grams or a combination of character and word n-grams.

Metric	**SVM Run 1**			**SVM Run 2**			**BERT**
	Char n-gram (1 to 6)	**Word n-gram (1 to 3)**	**Char + Word**	**Char n-gram (1 to 6)**	**Word n-gram (1 to 3)**	**Char + Word**	
Precision	0.6288	0.6127	0.6191	**0.6379**	0.6169	0.6204	0.6103
Recall	0.6209	0.6100	0.6183	**0.6298**	0.6135	0.6200	0.6104
F1	0.6241	0.6112	0.6186	**0.6330**	0.6150	0.6201	0.6101

Table 5: Dev Set Results

	SVM1 (Char n-gram)			**SVM1 (Word n-gram)**			**SVM1 (Char + Word n-gram)**		
	Pred NEG	**Pred NEU**	**Pred POS**	**Pred NEG**	**Pred NEU**	**Pred POS**	**Pred NEG**	**Pred NEU**	**Pred POS**
True NEG	631	300	56	623	310	54	652	270	65
True NEU	252	763	272	274	713	300	260	722	305
True POS	74	342	710	91	309	726	93	320	713

Table 6: Confusion Matrix for SVM Run 1 on Dev Set

Tables 6 to 8 show the confusion matrices for our classifiers on the development set. As can be seen, the character n-gram based SVM classifier's strength was its ability to predict the *neutral* class. The word n-gram based classifier predicted the *positive* class better. Whereas the classifier trained using the combination of character and word n-gram features predicted the *negative* category better.

	SVM1 (Char n-gram)			SVM1 (Word n-gram)			SVM1 (Char + Word n-gram)		
	Pred NEG	**Pred NEU**	**Pred POS**	**Pred NEG**	**Pred NEU**	**Pred POS**	**Pred NEG**	**Pred NEU**	**Pred POS**
True NEG	636	293	58	627	304	56	652	265	70
True NEU	239	784	264	271	726	290	256	723	308
True POS	78	332	716	91	313	722	94	314	718

Table 7: Confusion Matrix for SVM Run 2 on Dev Set

	BERT		
	Pred NEG	**Pred NEU**	**Pred POS**
True NEG	600	304	83
True NEU	247	699	341
True POS	99	261	766

Table 8: Confusion Matrix for BERT on Dev Set

	SVM1	SVM2	BERT	Best System
F1	0.674	**0.678**	0.342	0.75
Rank	-	21/62	-	1/62

Table 9: Official Results on Test Set

While comparing BERT with SVM, it can be seen that the BERT classifier predicted the *positive* category better than SVM. However, it did not predict the *negative* and *neutral* classes well.

Table 9 shows the scores our classifier obtained on the official run. The SVM classifier trained on the cleaned data using character n-gram features was our best performing classifier. It obtained F1 score of 0.678 and obtained the 21st rank out of 62 participants. The BERT classifier's performance on the development set was quite comparable to the SVM classifiers. However, on the test data set, the BERT classifier did not perform well and obtained an F1 score of only 0.342.

6 Conclusion

BERT has been a very successful model in many of the natural language processing tasks. In our study, we used multilingual BERT for the detection of sentiment in code-mixed Hindi-English text. Its performance on the development set was comparable with the SVM classifier. However, it produced an F1 score of only 0.342 in the test data. The SVM classifier trained on character n-gram was our best performing classifier on the test set with an F1 score of 0.678.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 60(5):493–502.

Aditya Joshi, Balamurali A R, and Pushpak Bhattacharyya. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. In *Proceedings of the 8th International Conference On Natural Language Processing (ICON)*.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2482–2491. ACL.

Bing Liu. 2010. Sentiment analysis and subjectivity. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*, pages 627–666. Chapman and Hall/CRC.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165.

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.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830.

Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In Jaime Lloret Mauri, Sabu M. Thampi, Michal Wozniak, Oge Marques, Dilip Krishnaswamy, Sartaj Sahni, Christian Callegari, Hideyuki Takagi, Zoran S. Bojkovic, Vinod M., Neeli R. Prasad, Jose M. Alcaraz Calero, Joal Rodrigues, Xinyu Que, Natarajan Meghanathan, Ravi Sandhu, and Edward Au, editors, *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015, Kochi, India, August 10-13, 2015*, pages 1468–1473. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan. N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.