BAKSA at SemEval-2020 Task 9: Bolstering CNN with Self-Attention for Sentiment Analysis of Code Mixed Text

Ayush Kumar* **Harsh Agarwal*** **Keshav Bansal*** **Ashutosh Modi**
Indian Institute of Technology Kanpur (IITK)
{ayushk,harshaga,keshavb}@iitk.ac.in
ashutoshm@cse.iitk.ac.in

Abstract

Sentiment Analysis of code-mixed text has diversified applications in opinion mining ranging from tagging user reviews to identifying social or political sentiments of a sub-population. In this paper, we present an ensemble architecture of convolutional neural net (CNN) and self-attention based LSTM for sentiment analysis of code-mixed tweets. While the CNN component helps in the classification of positive and negative tweets, the self-attention based LSTM, helps in the classification of neutral tweets, because of its ability to identify correct sentiment among multiple sentiment bearing units. We achieved F1 scores of 0.707 (ranked 5^{th}) and 0.725 (ranked 13^{th}) on Hindi-English (Hinglish) and Spanish-English (Spanglish) datasets, respectively. The submissions for Hinglish and Spanglish tasks were made under the usernames *ayushk* and *harsh_6* respectively.

1 Introduction

The research problem of Sentiment Analysis of Code-Mixed Social Media Text appeared as part of the SemEval Shared Challenge 2020 (Patwa et al., 2020). Mixing languages while writing text, also called code-mixing, is a typical pattern observed in almost all forms of communication, including social media text. We only focus on two popular bilingual code-mixing styles namely Hinglish and Spanglish.

Sentiment Analysis is a term broadly used to classify states of human affection and emotion. Interpreting code-mixed languages is difficult not only because the sentences may not fit a particular language model, but also because mixed text on social-media usually contains tokens such as hashtags, and usernames.

In this paper, we present an ensemble of CNN and self-attention based LSTM, utilizing the XLM-R embeddings (Conneau et al., 2019). While CNNs have been used for sentiment analysis before (Wang et al., 2016; Yoon and Kim, 2017), none of the previous works have used a self-attention based LSTM along with it. We found that while the CNN component worked well for positive and negative tweets, the self-attention component worked better for neutral tweets, necessitating an ensemble of the two. The implementation of our system is made available via Github[1].

2 Related Work

Performing standard NLP tasks on code-mixed data has presented significant challenges. Vyas et al. (2014) attempted to find methods for POS tagging of code-mixed social media text.

Another work by Joshi et al. (2016) used CNNs to learn subword level embeddings and then utilized these embeddings in a BiLSTM network to learn subword level information from social media text. Subword level representations are particularly important while dealing with noisy texts containing misspellings and punctuations. However, this work doesn't capture information about word-level semantics.

More recent work by Lal et al. (2019) uses two parallel BiLSTMs, which they call the Collective and Specific Encoder and an additional feature network. This approach combines recurrent neural networks utilizing attention mechanisms, which helps in evaluating the overall sentiment using attention weights when presented with a mixture of local sentiments.

* Authors equally contributed to this work.

[1] https://github.com/keshav22bansal/BAKSA_IITK

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1221–1226
Barcelona, Spain (Online), December 12, 2020.

3 Proposed Approach

3.1 Pre Processing

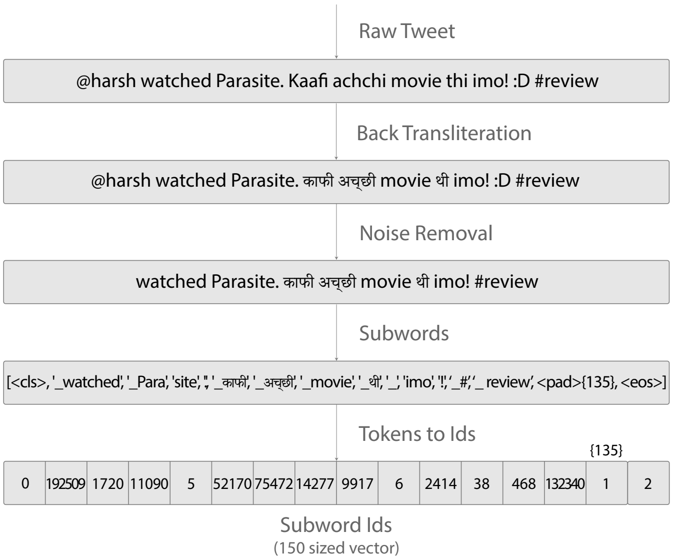
Raw Tweet
@harsh watched Parasite. Kaafi achchi movie thi imo! :D #review
Back Transliteration
@harsh watched Parasite. काफी अच्छी movie थी imo! :D #review
Noise Removal
watched Parasite. काफी अच्छी movie थी imo! #review
Subwords
[<cls>, '_watched', '_Para', 'site', '.', '_काफी', '_अच्छी', '_movie', '_थी', '_', 'imo', '!', '_#', '_ review', <pad>{135}, <eos>]
Tokens to Ids
{135}
0
192509
1720
11090
5
52170
75472
14277
9917
6
2414
38
468
132340
1
2
Subword Ids
(150 sized vector)

Figure 1: Preprocessing Pipeline

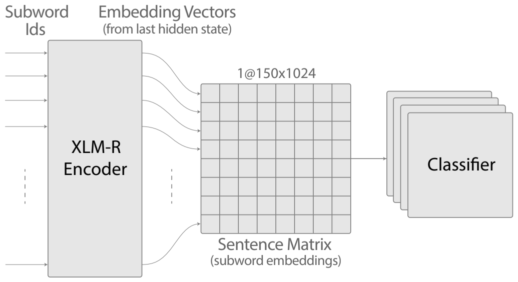

Figure 2: XLM-R Encoder

The tweets have been originally provided in the Latin script with their corresponding language tags. Before feeding the tweets to any training stage, they are preprocessed using the following procedure (Figure 1):

1. Back-Transliteration: All the words with "Hindi" language tags are converted into Devanagari words using phonetic transliteration. Google's Transliteration API[2] was used for this purpose. The words with "Spanish" language tags are not transliterated.

2. Noise removal: Usernames (annotated as @username), URLs, and emoticons present in the tweets are removed altogether, while hashtags (annotated as #hashtag) are left as it is. We also experimented with replacing emoticons by their corresponding textual meaning, but removing them led to better performance.

3. Tokenization: Tweets after noise removal are tokenized into subwords using the XLM-R (Conneau et al., 2019) vocabulary and later converted into their corresponding IDs.

3.2 Embedding layer

Since our data comprised of code-mixed tweets, it was essential to use a multilingual model. For our proposed architecture, we used the XLM-R embeddings. XLM-R is a transformer-based masked language model trained on one hundred languages, using more than two terabytes of filtered CommonCrawl data (Conneau et al., 2019).

The subword IDs from the pre-processing stage are fed to the XLM-R encoder. The final hidden state corresponding to each token is used for the classification task as inputs to the proceeding components (See figure 2). The XLM-R encoder is fine-tuned during training to generate better encodings for the code-mixed text.

We also experimented with the Multilingual BERT (henceforth, M-BERT), released by Devlin et al. (2018). We found that XLM-R performed much better than M-BERT for our dataset.

[2] https://www.google.com/inputtools/services/features/transliteration.html

3.3 Architecture

We propose an ensemble model comprising of two main components.

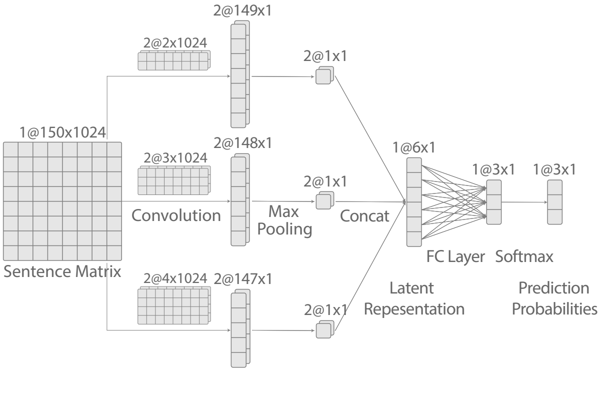

Figure 3: CNN Classifier

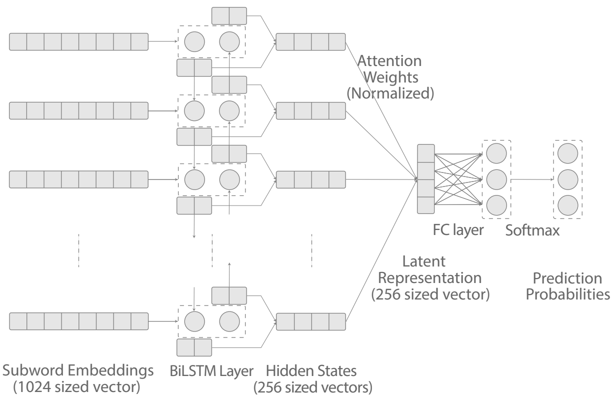
Attention Weights (Normalized)
FC layer
Softmax
Latent Representation (256 sized vector)
Prediction Probabilities
Subword Embeddings (1024 sized vector)
BiLSTM Layer
Hidden States (256 sized vectors)

Figure 4: Self-Attention Classifier

3.3.1 CNN Classifier

The first component is a convolutional neural network (Lecun, 1989) (henceforth, CNN). CNNs, to some extent, take into account the ordering of the words and the context in which each word appears.

We generate the required embedding by passing the subword embeddings of a sentence individually into 1-D CNN. We perform a convolution with 3 different filter sizes (2, 3 and 4), before adding a bias and applying a non-linear RELU activation.

The idea behind using several filter sizes was to capture contexts of varying lengths. The convolution layer is used to extract local features around each word window, while the max-pooling layer is used to extract the essential features in the feature map. XLM-R embeddings are passed through this component and, ultimately, through a softmax function to obtain the predictions of the first component. We call these predictions p_{CNN}.

3.3.2 Self-Attention Classifier

The second component is a self-attention based classifier (See figure 4). It helps in choosing the overall sentiment when presented with a mixture of sentiments. We use soft-attention (Xu et al., 2015), a deterministic, differentiable attention mechanism, where a softmax gives the weights for each subword, and the output of the attention module is a weighted sum of hidden representations at each location.

The self-attention component comprises a BiLSTM (Hochreiter and Schmidhuber, 1997) layer, which takes as input the output of the XLM-R encoder. The hidden state obtained from the BiLSTM layer for each subword is used to calculate the attention scores.

Suppose a sequence is given by the subwords $(w_1, w_2, ..., w_n)$. Let the i^{th} forward hidden state in the BiLSTM be represented by $\overrightarrow{h_i}$ and i^{th} backward hidden state by $\overleftarrow{h_i}$. The combined annotation k_i is obtained by concatenating $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$. We first concatenate the forward and backward hidden states to obtain a combined annotation $(k_1, k_2, ..., k_n)$.

$$k_i = [\overrightarrow{h_i}; \overleftarrow{h_i}] \tag{1}$$

The attention mechanism gives a score e_i to each subword i in the sentence S, as given by (2).

$$e_i = {k_i}^T k_n \tag{2}$$

Then the attention weight a_i of each k_i is computed by normalizing the attention score e_i

$$a_i = \frac{exp(e_i)}{\sum_{j=1}^{n} exp(e_j)} \tag{3}$$

We then calculate the sentence latent representation vector h using equation (4)

$$h = \sum_{i=1}^{n} a_i \times k_i \tag{4}$$

The representation is thus a weighted combination of all the hidden states. The representation vector h is then passed through a fully connected layer followed by a softmax to obtain predictions p_{att}.

The predictions from the first and second components are aggregated (See figure 5) using element wise product (denoted by $\circ$) to obtain the final predictions ($p_{final} = p_{CNN} \circ p_{att}$). We experimented with other aggregating techniques like linearly weighted average, but element-wise product worked out better.

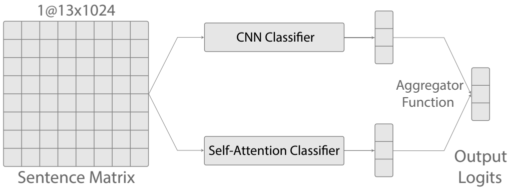

Figure 5: Ensemble Classifier

	Dataset	Positive	Neutral	Negative
Train	Hinglish	5264	4634	4102
	Spanglish	6005	3974	2023
Validation	Hinglish	982	1128	890
	Spanglish	1498	994	506
Test	Hinglish	1000	1100	900

Table 1: Statistics of training and development data

4 Data Description

We used the dataset provided by the organizers of Task-9 of SemEval 2020 (Patwa et al., 2020) for training both Hinglish and Spanglish models. The data has been annotated semi-automatically. The statistics of the dataset are shown in Table 1. The dataset for Hinglish is balanced while that of Spanglish is highly unbalanced. For hyperparameter tuning, we used the validation set provided by the organizers.

5 Experiments and Results

We first trained a vanilla CNN model on the provided dataset using the XLM-R embeddings. The CNN model seemed to be confused on neutral data points but worked well on positive and negative tweets.

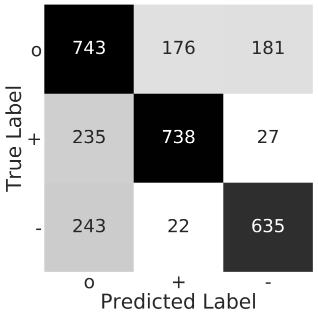

Figure 6: Confusion matrix for Ensemble on Hinglish test data

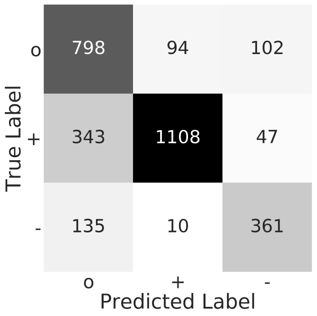

Figure 7: Confusion matrix for Ensemble on Spanglish validation data [3]

The self-attention model outperforms the previous model on neutral data points though it performs worse on the positive and negative samples. The good performance on neutrals can be attributed to the fact that neutral tweets may contain multiple sentiment bearing units which the model is capable of handling.

Combining the results of CNN with those of the Self-Attention model was the primary motivation for using an ensemble of the two. The ensemble outperforms all our previous models, achieving a recall of 0.705 with an F1-score of 0.707 on the Hinglish test dataset and a recall of 0.696 with an F1-score of 0.725 on the Spanglish test dataset (See table 2). The confusion matrices for the ensemble on both datasets are shown in figure 6 and 7 (o : neutral, + : positive, - : negative). Our team was ranked 5^{th} among 62 teams in Hinglish and 13^{th} among 29 teams in Spanglish.

[3] Validation data was used for constructing the confusion matrix for spanglish as true labels for test data were not available

	F1				Macro	Macro
	o	**+**	**-**	**Macro**	**Precision**	**Recall**
Hinglish	0.640	0.762	0.729	0.707	0.712	0.705
Spanglish	0.135	0.825	0.375	0.725	0.763	0.696

Table 2: Performance of Ensemble system on Hinglish and Spanglish test datasets

6 Analysis

6.1 Visualization of the individual components

To visualize the sentence embeddings learned by the model for the Hinglish test dataset, we projected the sentence vectors obtained before the final fully connected layer onto a lower-dimensional subspace using the t-SNE algorithm (van der Maaten and Hinton, 2008) for the two components (See figure 8).

For CNN, the positive and negative tweets seem to form two distinct clusters, while the neutral tweets are scattered among them. In contrast, for the self-attention component, neutrals seem to form a distinct cluster, while the positive and negative classes are partially dispersed in a wide region. Thus, the two components, in a way, complement each other for better predictions over all the three classes.

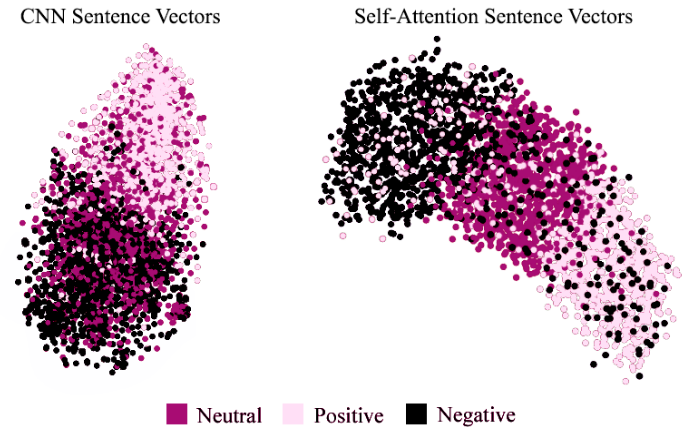

Figure 8: Visualisation of CNN and Self-Attention Sentence Vectors

6.2 Error Analysis

Most of the misclassifications were made by our model on the following three types of tweets -

1. Neutral - Despite the improvement due to the self-attention classifier, the performance on neutral tweets still lags much behind positive and negative tweets.

2. Sarcastic - Sarcasm is the use of irony to mock or convey contempt. Tweets such as ***Best wishes to pseudo atheist In new country in advance. Bon voyage*** are challenging to classify due to their hidden context and are falsely predicted as positive by our model.

3. Mildly negative - Due to exorbitant amount of abusive tweets in the data, some mildly negative ones like ***South africa team bekar h jab tak ushme ABD villers na ho*** are falsely predicted as neutral.

7 Conclusion

For our system, we use an ensemble of CNN and Self Attention architectures with XLM-R multilingual embeddings. We analyze which models work better for different classes of tweets. Our self-attention system helps in better classification of neutral tweets, which are difficult to classify due to multiple sentiment bearing units. Creating an ensemble with CNN helps in better classification of all the three classes. We also visualize how our model performs on different classes of tweets using the t-SNE algorithm. Our results show an improvement over some of the previous works in this field.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.

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 *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491, Osaka, Japan, December. The COLING 2016 Organizing Committee.

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, July. Association for Computational Linguistics.

Yann Lecun, 1989. *Generalization and network design strategies*. Elsevier.

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.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

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, Doha, Qatar, October. Association for Computational Linguistics.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 225–230, Berlin, Germany, August. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

Joosung Yoon and Hyeoncheol Kim. 2017. Multi-channel lexicon integrated CNN-BiLSTM models for sentiment analysis. In *Proceedings of the 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017)*, pages 244–253, Taipei, Taiwan, November. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).