HCMS at SemEval-2020 Task 9: A Neural Approach to Sentiment Analysis for Code-Mixed Texts

Aditya Srivastava
International Institute of Information Technology, Hyderabad
aditya.srivastava@research.iiit.ac.in

V. Harsha Vardhan
International Institute of Information Technology, Hyderabad
harshavardhan.v@research.iiit.ac.in

Abstract

Problems involving code-mixed language are often plagued by a lack of resources and an absence of materials to perform sophisticated transfer learning with. In this paper we describe our submission to the Sentimix Hindi-English task involving sentiment classification of code-mixed texts, and with an F1 score of 67.1%, we demonstrate that simple convolution and attention may well produce reasonable results.

1 Introduction

Sentiment analysis has gained importance in the current world where social media is crucial in gauging public opinion on products, political campaigns, latest trends and more. A large portion of the world's population is multilingual, and so is the content on social networks. Code-mixing is a natural phenomenon among bilinguals where phrases and words from one language are employed in another. Typically, the underlying grammar of the primary language that is being spoken is kept intact and phrases from another language are embedded into it. India in particular, has 23 officially recognized languages and a majorly bilingual population, requiring robust computational tools to effectively exploit the data it produces.

Deep neural networks have had great success in predicting sentiment encapsulated in text - a big part of this success has been their ability to utilize pretrained word embeddings. In code-mixed tweets however, Hindi words are written in Roman script instead of its native Devanagari script, forcing one to use a transliterator in order to take advantage of pretrained aligned word embeddings, like *fastText* (Bojanowski et al., 2016). These words in Roman script are unnormalized and may have multiple spellings; for example, *bahut* can be spelt in Roman script as *bahut*, *bohot*, *bohut* etc., and such errors propagate from the transliterator to our downstream task, thereby reducing performance.

In this work, we propose a deep convolution network with self-attention which shows promising results, without any pretraining. Convolution neural networks (CNNs) are known to capture local relations (Kim, 2014) or *local context*, and work here as feature extractors acting as a substitute for handcrafted sentiment features. A self-attention layer is then applied over these features (presenting as vectors), which allow each individual feature to attend to all other features (Bahdanau et al., 2014), providing *global context*. We call this the **Hierarchical Context Modeling System** (henceforth referred to as **HCMS**), due to the hierarchical nature of context extraction performed on both levels.

2 Background

Early sentiment analysis systems made use of corpus based statistics (Wiebe and others, 2000), linguistic tools like Wordnet (Miller, 1998) and lexicon based classifiers (Turney and Littman, 2002). More recently, a large portion of work has involved Recurrent Neural Networks (RNNs), Long Short Term Memory Networks (LSTMs) and CNNs at word and character levels (Mulder et al., 2015; Zhang et al., 2015). Usage of sub-word level representations (based on character n-grams) (Joshi et al., 2016), combining both CNNs and LSTMs have also been demonstrated, where the CNN was used to generate sub-word features which were then augmented to word level features. Sharma et al. (2015) also set precedent for

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1253–1258
Barcelona, Spain (Online), December 12, 2020.

normalizing code mixed text by dealing with the multiple transliterated word forms being generated as a result of the Romanization of non-English scripts.

Attention mechanisms were first introduced for neural machine translation, and achieved state-of-the-art results at the time. The idea, which produces significant gains in our own model, has been applied to a variety of tasks such as image captioning, text classification, question answering (Cheng et al., 2016; Xu et al., 2015; Choi et al., 2017), and even aspect level sentiment classification for more fine grained feature extraction (Wang et al., 2016). Attention helps in obtaining better representations of text which in turn boosts downstream performance and thus, is a vital component of HCMS where self-attention is used to boost the performance of the underlying architecture.

3 System Overview

Approaching the problem as one of sequence classification, we present the HCMS architecture as depicted in figure 1. For every tweet (represented as a sequence of word embeddings), our model first performs a convolution and max-pooling operation to get local context between words that are adjacent to one another. In the next step, self-attention allows every context vector to attend to every other context vector in the sentence to get global context, summarizing the tweet. Lastly, the model passes this summary through a dense layer to produce probabilities over the three classes, namely positive, negative and neutral.

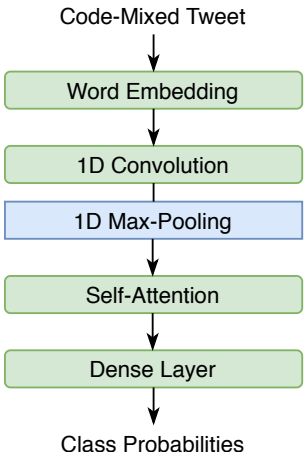


Figure 1: Model Architecture

3.1 Algorithm

Given a tweet X as $[x_1, ..., x_u]$, where every $x_i \in \mathbb{R}^d$ is a d dimensional word embedding. The model first performs a ReLU activated, 1D convolution operation along the tweet followed by a max-pooling operation, to produce localized context vectors C as $[c_1, ...c_v]$, where every $c_i \in \mathbb{R}^{d'}$. The dimensions of C will depend on the parameters of the CNN and max-pooling layers.

$$C = \text{MaxPool}(\text{ReLU}(\text{CNN}(X)) \tag{1}$$

Next, the model performs self-attention between the local context vectors in C. Self-attention lets every $c_i \in C$ attend to every other $c_{i'} \in C$ to produce attention vectors A as $[a_1, ...a_v]$. These vectors are then concatenated to produce global context vector G as $[g_1, ...g_w]$, where every $g_i \in \mathbb{R}$.

For every t^{th} local context vector,

$$\begin{aligned} h_{t,t'} &= \tanh(c_t^T W_t + c_{t'}^T W_c + b_t) && c_t, c_{t'} \in C \text{ and } t \neq t' \\ e_{t,t'} &= \sigma(W_a h_{t,t'} + b_a) \\ q_t &= \text{Softmax}(e_t) \\ a_t &= \sum_{t'} q_{t,t'} c_t && a_t \in A \\ G &= \text{Concatenate}(l) \end{aligned} \tag{2}$$

Finally, the global context vector G is passed through a fully connected layer with Softmax activation to predict probabilities Y' over the three classes.

$$Y' = \text{Softmax}(GW + b) \tag{3}$$

3.2 Loss

The model computes categorical cross-entropy loss L between the predicted class probabilities and the correct class for the sample, as given below

$$L(Y, Y') = -\sum Y \odot \log(Y') \tag{4}$$

4 Experimental Setup

4.1 Dataset

The data, provided by the task organizers Patwa et al. (2020), comprises of tweets that employ the Hinglish mix. The data is in CONLL format, with each word being tagged as belonging to the English or Hindi language if applicable, else tagged as O if neither or EMT if an emoticon. Every tweet has a corresponding ID and sentiment label attached to it.

4.2 Data Preprocessing

We experimented with multiple steps for cleaning the tweets, including but not limited to lowering text case, expanding English contractions, replacing emoticons by their meanings in English, removing character repetitions, and removing hashtags, usernames and links. After cleaning, the data was tokenized and indexed by word and converted into embeddings. Features such as language labels for each word were converted to one hot vectors and concatenated to the word vectors (not present in final submission).

4.3 Additional Details

The 1D CNN layer has 200 filters, each of kernel size 8 with stride of 1. We used randomly initialized word embeddings of size 200. The model was optimized using Adam with a learning rate of 0.01, *beta1* as 0.9, *beta2* as 0.99 and *epsilon* as $1e-7$.

All the experiments were carried out on a single Nvidia GeForce RTX 2080 Ti GPU. The experiments were written in Python3, using the Keras framework (Chollet, 2015). The code itself is hosted on GitHub under the MIT license[1].

4.4 Evaluation Metric

For evaluation we simply use F1 as our metric, emulating the task leaderboard.

5 Results and Analysis

We used the official validation and test sets provided. While our contest results stand at 67.1% F1, we have fine tuned our model since, and the latest scores for all experiments are shown in table 1. (Contest results are visible on the leaderboard under the authors' aliases, *the0ne* and *talent404*.)[2]

S.No.	Model Description	Precision	Recall	Acc.	F1
1.	HCMS	69.51	68.01	68.10	**68.44**
2.	BiLSTM + self-attention	67.67	67.44	67.23	67.49
3.	HCMS w/o self-attention	66.52	66.75	66.17	66.56
4.	fastText Classifier	67.16	65.74	65.77	66.26
5.	BiLSTM w/o self-attention	71.39	65.02	65.67	66.18
6.	SVM	65.86	67.20	66.13	65.97

Table 1: Model descriptions and their results arranged in descending order of test F1 scores.

The SVM (Hearst, 1998) trained on n-grams of sizes 1, 2 and 3 returned an F1 score of 65.97%. Given its ability to handle out-of-vocab words, fastText (Joulin et al., 2017) (also an n-gram based model) outperformed the SVM model, achieving 66.26% F1.

Ablating the self-attention mechanism from the HCMS model left a vanilla 1D CNN with max pooling. Both, the LSTM and the CNN performed similarly to the n-gram based fastText model, but it is clear from the results that applying self-attention on top of features extracted by the respective models improved their

[1]Link to code repository.

[2]Link to contest leaderboard - authors' usernames are *the0ne* and *talent404*.

performance significantly, providing credibility to our hypothesis of local and global features in the data. HCMS can be seen outperforming the models lacking self-attention by a significant ~1%, giving a score of 68.44%.

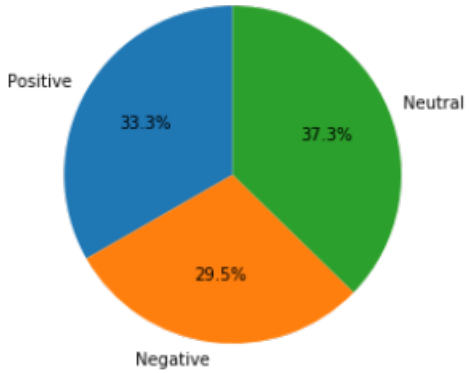
Preprocessing Type	F1
Emoji and contraction replacement	**68.44**
Only contraction replacement	68.25
Only emoji replacement	68.18
No Cleaning	65.97

Table 2: Preprocessing steps arranged in descending order of test F1 scores.

It was observed that cleaning the data improved performance. Table 2 shows how specific forms of data preprocessing affect HCMS. Emoji replacement refers to the substitution of graphical emojis with their ASCII counterparts.[3] Contraction replacement refers to the substitution of English contractions like *can't* or *wouldn't* with their complete forms *cannot* and *would not* respectively. We can see in the table that using both gives the best performance.

We also analysed the data and observed language distribution across the corpus. The numbers have been presented in figure 2. As expected, the vocab is not equally split between the two languages.

(a) Sentiment distribution over the corpus.



(b) Language distribution over the corpus.

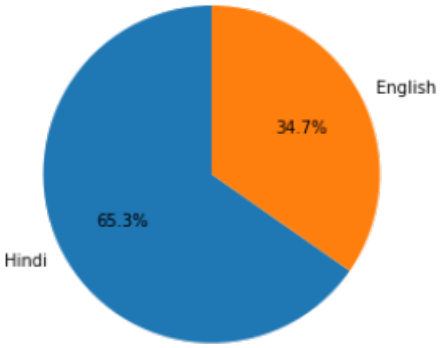


Figure 2: Data distribution charts.

In fact, there is greater presence of Hindi, and that too is Romanized and not in its native Devnagari script. In our experiments we tried to use a pretrained transliterator to de-Romanize the Hindi words, and then tried to employ pretrained word-embeddings, models and sentiment lexicon - due to the noisy Devnagari-to-Roman transliteration by the authors of the tweets however, our reverse transliteration failed and most words did not map into their respective embedding spaces.[4] The results of those experiments are presented in table 3 (note that the other steps of the data cleaning process were preserved).

Experiment	F1
Reverse transliteration + Aligned fastText	55.9
Reverse transliteration + Multilingual BERT	52.3
Reverse transliteration + Sentiment annotation	52.7

Table 3: Failed experiments due to noise in the data.

In our experiments, another noteworthy occurrence was that even though some models showed high validation scores, they were outperformed on the test set by models that produced lower validation score in comparison. This was a recurring theme in our A-B testing over validation and test data, and we feel that this can again be attributed to the noisy nature of code-mixed language.

6 Conclusion

In this paper we have discussed a way to perform sentiment analysis on code-mixed Hindi-English data, and we feel that the system is versatile enough to be applied to any mix of languages, especially when

[3]Links to library used for emoji replacement.

[4]Links to Indictrans Transliterator, aligned fastText embeddings, multilingual-BERT and sentiment lexicon.

resources for the mixture are unavailable or hard to find. We have also looked at ways that such data can be preprocessed for training, and how a multilevel neural architecture is able to extract context from text.

While it is encouraging to see that our model performs consistently, there are a couple of caveats that need to be addressed - even though the lack of pretraining, a low resource setting, and noise in data seem surmountable hurdles to the task, it is evident that the most promising avenues for improvement lie in exploring data cleaning/normalization and transfer learning. Further analysis of the present methodology could also reveal the kind of transfer learning required - whether it be pretrained word embeddings, large scale language modeling or perhaps a better transliteration pipeline to name a few. To do so would require larger quantities of data and the application of the system to other linguistic tasks of settings similar to that of Sentimix. All of these steps qualify within the scope of future work, and committing to them will reward us with more confidence in the hypotheses we propose in this paper, providing further proof of the system's robustness and effectiveness. We hope to undertake efforts for the same soon.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-Fine Question Answering for Long Documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220, Vancouver, Canada, July. Association for Computational Linguistics.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Marti A. Hearst. 1998. Support Vector Machines. *IEEE Intelligent Systems*, 13(4):18–28, July.

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.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1):61 – 98.

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.

S. Sharma, P. Srinivas, and R. C. Balabantaray. 2015. Text normalization of code mix and sentiment analysis. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1468–1473.

Peter D Turney and Michael L Littman. 2002. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. *arXiv preprint cs/0212012*.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Janyce Wiebe et al. 2000. Learning subjective adjectives from corpora.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification.