Looking Beyond the Obvious: Code-Mixed Sentiment Analysis (CMSA)

by

Vaibhav Kumar, Mrinal Dhar

in

Conference on Empirical Methods in Natural Language Processing
(*EMNLP-2018*)

Brussels, Belgium

Report No: IIIT/TR/2018/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2018

Looking Beyond the Obvious: Code-Mixed Sentiment Analysis (CMSA)

Anonymous EMNLP submission

Abstract

Sentiment analysis of online social media has many applications in e-commerce, recommendation systems, analysis of current trends, political campaigns, etc. In a multilingual society, such content is often a composition of different languages. This phenomenon of mixing the vocabulary and syntax of two or more languages (code-mixing) makes the processing of such content significantly harder.

In this paper, we present a hybrid architecture for the task of Sentiment Analysis of English-Hindi code-mixed data. We use two different Bidirectional Long Short Term Memory (BiLSTM) Networks, one that looks at the overall sentiment of the sentence while the other utilizes an attention mechanism in order to focus on the individual sentiment bearing sub-words. This, combined with traditionally used orthographic features and monolingually trained word embeddings achieves the state-of-the-art results on a benchmark dataset. Our system scores an accuracy of 83.54% and an F1-score of 0.827.

1 Introduction

Sentiment analysis (SA) has a variety of applications in analyzing movie reviews, user modeling, curating online trends and political campaigns and opinion mining. A majority of this information comes from online social media such as Facebook and Twitter. In recent times, these websites have taken over traditional forms of digital communication like e-mail. As communication became more informal and as the ease of putting one's thoughts on the Internet increased, so did the presence of code-mixing online.

A large number of Indian users on online social media websites can speak English and Hindi with bilingual proficiency. Consequently, English-Hindi code-mixed content has become ubiquitous on the Internet which has created the need to process this form of natural language. Myers-Scotton (1993) defines code-mixing as ”the embedding of linguistic units such as phrases, words and morphemes of one language into an utterance of another language”. Typically, a code-mixed sentence retains the underlying grammar and script of one of the languages it is comprised of.

However, given that there is no formally defined grammar for a code-mixed hybrid language, traditional approaches to SA do not work very well on code-mixed content. Choudhury et al. (2007) investigated how language used on these social media platforms, which they have called *texting* language, differs from the standard language that is found in more formal texts like books. Adding to the problem is the lack of available annotated code-mixed data for Sentiment Analysis, which is a severe limitation in itself for deep learning techniques.

Due to the nature of code-mixing, words from one of the two code-mixed languages may not be written in the same script as their standard form. For example, in the code-mixed sentence *bahan, where are you?*, the word *bahan* is written in a non-standard Roman spelling for the actual Hindi word “बहन”, which means “sister”. As there is no formally ”correct” way of spelling these words in a non-standard script, this results in spelling variations between different writers and makes existing sentiment analysis systems unsuitable for code-mixed data. Therefore, there is a need to develop new systems that target code-mixed content specifically.

There have been multiple attempts in the past to develop solutions for this problem, including a shared task in ICON 2017 that targets Sentiment Analysis of code-mixed Indian languages specifically (Patra et al., 2018).

Pravalika et al. (2017) make use of a lexicon lookup approach for performing domain specific sentiment analysis. Other attempts include using Sub-word level compositions with LSTMs to capture sentiment at morpheme level (C16), or using contrastive learning with siamese networks to map code-mixed and standard language text to a common sentiment space (Choudhary et al., 2018). While these systems are effective to a degree, they are primarily deep learning techniques and hence they're limited by their dependence on abundant data.

In this paper, we propose a hybrid neural network framework, called *Code-Mixed Sentiment Analysis* or CMSA, that tackles the limitations of the previous systems by combining deep learning techniques with training of surface features and sentence vector representations, which we call a *Feature Network*. By augmenting the neural network with specific linguistic features, we are able to work around the lack of abundant data.

The primary contributions of the work presented in this paper include development of a neural network architecture that utilizes Attention mechanisms with LSTMs and simultaneously learns a Feature Network, using it to augment the deep learning techniques for the task of sentiment analysis.

We performed extensive experimental evaluation to demonstrate the effectiveness of our system in comparison to state-of-the-art and traditional techniques for the sentiment analysis task. To the best of our knowledge, our system outperforms the state-of-the-art systems for sentiment analysis of English-Hindi code-mixed data.

2 Related Work

The problem of sentiment analysis has attracted extensive studies and a myriad of publications. Mohammad et al. (2013) created two SVM based classifiers to detect sentiments in tweets and text messages respectively, involving the use of a number of surface-form and semantic features.

Bojanowski et al. (2016) proposed a skip-gram based word representation model that proved to be effective for languages with large vocabularies, and could be useful in classification tasks such as sentiment analysis. Giatsoglou et al. (2017) trained document vectors lexicon-based, word embedding-based and hybrid vectorizations with a polarity classification model for sentiment analysis. Their approach proves to be useful when there is a lack of available computational resources.

Another approach has been to use ensemble techniques for improving sentiment analysis by deep learning systems (Araque et al., 2017). They used word embeddings and a linear machine learning algorithm as their base classifier and aggregate it with two models for surface classifiers used for sentiment analysis.

Code-mixing has attracted a lot of research in recent times, and there is considerable effort in the community to support code-mixing in language systems and models.

Bali et al. (2014) studied the extent of English-Hindi code-mixing on online social media, particularly Facebook, and their work demonstrates the need for systems that can automatically process this data. Vyas et al. (2014) deal with POS tagging of code-mixed data obtained from online social media such as Facebook and Twitter.

Apart from this, Sharma et al. (2016) were the first to attempt shallow parsing of code-mixed data obtained from online social media. Work has also been done to support word level identification of languages that comprise a code-mixed language in given text (Chittaranjan et al., 2014).

There is some work for Code-mixed sentiment analysis too. Sharma et al. (2015) presented an approach based on lexicon lookup for text normalization and sentiment analysis of code-mixed data. C16 used sub-word level compositions to learn information about the sentiment value at the morpheme level. Choudhary et al. (2018) use siamese networks to maintain a common sentiment space mapping between code-mixed and standard languages. They utilize contrastive learning to classify sentiments in code-mixed content.

Although, the above models work well, they suffer from the lack of abundant data required for training a good deep model. We show that by looking at the input differently and by augmenting the model with specific features we can achieve significant improvements on the state-of-the-art.

3 Model Architecture

In this section we present the architecture of our system. We first present the overview followed by the explanations of the various components of the model.

3.1 Overview

Our system consists of three primary components. The first component uses sub-word level representations with a Convolutional Neural Network (CNN). Sub-words provide the optimal level of detail in order for the CNN to capture morpheme level sentiment information. In Section 3.3, we describe our second component, a parallel encoder network consisting of two bi-directional Long Short Term Memory (BiLSTM) Networks, that capture 1) sentiment information from specific parts of the sentence and 2) the overall sentiment information of a sentence.

Finally, the third component, which is a Feature Network, described in Section 3.4, consists of surface features and a vector representation of the sentence that can represent out-of-vocabulary words as well. It augments the neural network framework of our system to boost the classification accuracy for the task of sentiment analysis.

3.2 Sub-word level representations with CNN

We take inspiration from the work done by C16 involving sub-word level compositions for a similar task in sentiment analysis.

Traditionally, statistical approaches to sentiment analysis work on word-level feature representations. However, they make the assumption that a language has finite vocabulary which might work for English but clearly does not work in our case, given the problem of spelling variations in code-mixed language. Therefore, word-level representations would be too coarse for our task.

On the other end of the spectrum would be character-level feature representations. However, it is difficult to establish a mapping between the meaning of a word and the way it is constructed by characters. As one character inherently does not provide any semantic information that can be used for our purpose, we dismiss character-level feature representations as our choice of embeddings as well.

As a middle ground between the two discussed strategies, sub-word level embeddings prove to be an appropriate choice for representations for our task. While individual characters might not provide any semantic value on their own, groups of characters can hold meaning that can be interpreted by humans. For example: The word *disengage* can be broken down into the sub-words *dis* and *engage*, both of which project semantic information.

Therefore, we believe that sub-word level representations are correctly suited for our task.

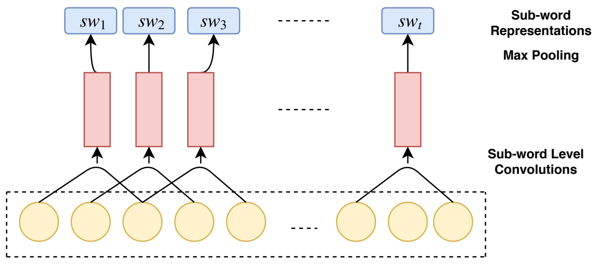

Figure 1: Convolution Neural Network for Generating Sub-Word Representation

3.3 BiLSTM network with Attention mechanisms

We utilize a combination of two different encoders in our model, an overview of which was presented in Section 3.1. Below we explain both these encoders in detail.

3.3.1 Collective Encoder

The purpose of the collective encoder is to learn a representation which encapsulates the overall sentiment of the sentence. The graphical representation of this encoder can be seen in Fig. 2(A). The inputs to the Collective Encoder are sub-word representations obtained from the CNN. In this encoder, we use RNNs with BiLSTM cells. This encoder is useful in coming up with a representation of the input sentence that captures the overall sentiment information embedded in it. The last hidden state of the BiLSTM i.e h_t, encapsulates the overall summary of the entire sentence's sentiment which we represent as cmu^c.

The forward state updates of the BiLSTM satisfy the following equations

$$\overrightarrow{f}_t = \sigma[\overrightarrow{W}_f[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t] + \overrightarrow{b}_f] \qquad (1)$$

$$\overrightarrow{i}_t = \sigma[\overrightarrow{W}_i[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t] + \overrightarrow{b}_i] \qquad (2)$$

$$\overrightarrow{o}_t = \sigma[\overrightarrow{W}_o[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t] + \overrightarrow{b}_o] \qquad (3)$$

$$\overrightarrow{l}_t = \tanh[\overrightarrow{V}[\overrightarrow{h}_{t-1}, \overrightarrow{r}_t] + \overrightarrow{d}] \qquad (4)$$

$$\overrightarrow{c}_t = \overrightarrow{f}_t \cdot \overrightarrow{c}_{t-1} + \overrightarrow{i}_t \cdot \overrightarrow{l}_t \qquad (5)$$

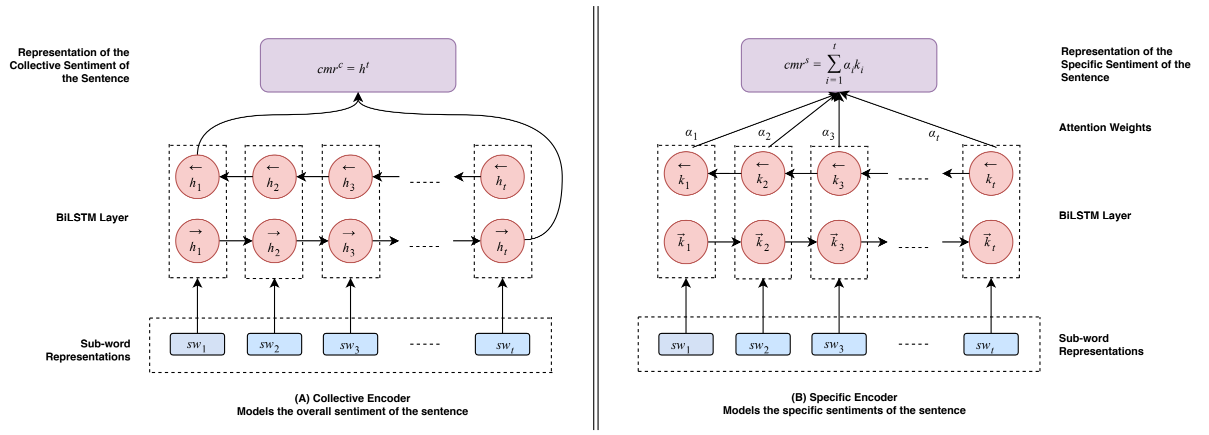
Representation of the Collective Sentiment of the Sentence
$cmr^c = h^t$
BiLSTM Layer
Sub-word Representations
sw_1 sw_2 sw_3 ..... sw_t
(A) Collective Encoder
Models the overall sentiment of the sentence
$cmr^s = \sum_{i=1}^{t} \alpha_i k_i$
Representation of the Specific Sentiment of the Sentence
Attention Weights
α_1 α_2 α_3 α_t
BiLSTM Layer
Sub-word Representations
(B) Specific Encoder
Models the specific sentiments of the sentence

Figure 2: The two different encoders used in the model.

$$\overrightarrow{h}_t = \overrightarrow{o}_t \cdot \tanh(\overrightarrow{c}_t) \tag{6}$$

here σ is the logistic sigmoid function, $\overrightarrow{f}_t$, $\overrightarrow{i}_t$, $\overrightarrow{o}_t$ represent the forget, input and output gates respectively. $\overrightarrow{r}_t$ denotes the input at time t and $\overrightarrow{h}_t$ denotes the latent state, $\overrightarrow{b}_f$, $\overrightarrow{b}_i$, $\overrightarrow{b}_o$ and $\overrightarrow{d}$ represent the bias terms. The forget, input and output gates control the flow of information throughout the sequence. The backward states $(\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_t)$ are computed in a similar manner as above.

Finally, we represent the vector cmr^c as follows:

$$cmr^c = h^t = \begin{bmatrix} \overrightarrow{h}_t \\ \overleftarrow{h}_1 \end{bmatrix} \tag{7}$$

3.3.2 Specific Encoder

The graphical representation of this encoder can be seen from Fig. 2(B). The architecture of the Specific Encoder is very similar to the Collective Encoder, with one major difference: in this encoder, we employ a sub-word level attention mechanism. Our motive behind this is to be able to choose the sub-words which contribute the most towards the input's sentiment.

Different sub-words may encapsulate different information about sentiments, however it is crucial to identify the sub-words which play an important role in defining the overall sentiment of the sentence. The Specific Encoder helps in this by providing a context vector cmr^s.

In order to generate this context vector, we first concatenate the forward and backward states to obtain the annotations $(k_1, k_2, \ldots, k_t)$, where

$$k_i = \begin{bmatrix} \overrightarrow{k}_i \\ \overleftarrow{k}_i \end{bmatrix} \tag{8}$$

We then calculate the context vector cmr^s as follows:

$$cmr^s = \sum_{j=1}^{R} \alpha_j k_j \tag{9}$$

where α_i determines the part of the input sequence which should be emphasized or ignored and k_i stands for the output of the hidden units. α_i is computed by using a global context vector u_w as the query (Yang et al., 2016). u_w is randomly initialized and learned during the training.

Using such a mechanism helps our model to adaptively select the more important sub-words from the less important ones.

3.3.3 Fusion of the Encoders

The fusion of the vectors obtained from both the encoders can be seen in Fig. 3. We concatenate the outputs obtained from both these encoders and use it as inputs to a fully connected neural network.

As shown in Fig. 2 and Fig. 3, we can see that h_t is incorporated into cmr^c to provide a summary of the sentiments present in the sentence. An important thing to notice is that, different encoding mechanisms will be evoked in both the encoders when trained jointly. The last hidden state of the collective encoder h_t plays a different role from that of k_t. The former has the responsibility to encode the sentiment present in the sequence of the sub-words which form the sentence, while the latter is used for computing attention weights. Information obtained from both the encoders is utilized

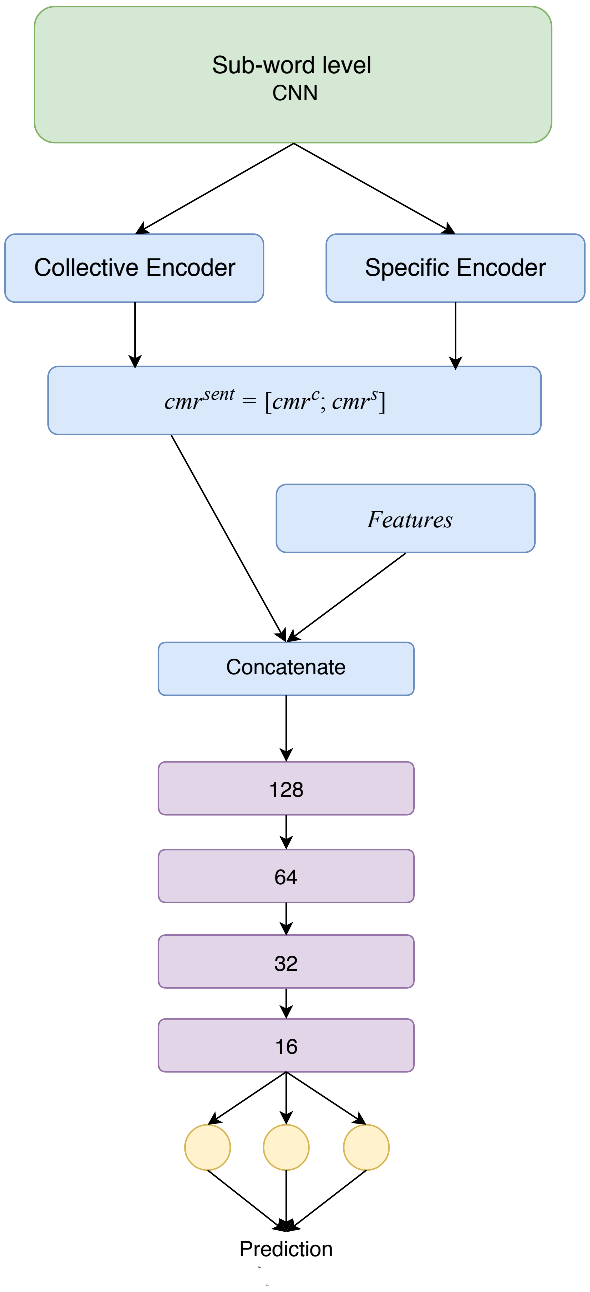

Figure 3: Complete Architecture of CMSA.

to come up with a unified representation of sentiment present in a sentence,

$$cmr^{sent} = [cmr^c; cmr^s] \tag{10}$$

where cmr^{sent} represents the unified representation of the sentiment.

3.4 Feature Network

Mohammad et al. (2013) attempted sentiment analysis of tweets using classifiers trained on surface level features. We attempt to use similar linguistic features to augment the neural network framework of our model. These features are defined as:

- Capital words: Number of words which are in all capital letters
- Extended words: Number of words which have one or more contiguous repeating characters.
- Aggregate positive and negative sentiments: Using SentiWordNet (Esuli and Sebastiani, 2006) for every word except articles and conjunctions, and combining the sentiment polarity values into net positive aggregate and net negative aggregate features.
- Repeated exclamations and other punctuation: Number of sets of two or more contiguous exclamations, question marks, full stops, etc.
- Exclamation at end of sentence: Boolean value denoting whether the given sentence ends with an exclamation or not.
- Monolingual Sentence Vectors: While we do not have any suitable data to train word vectors for the code-mixed Hindi words in our input, we can reliably use the word representations for the English words in the input. We chose to use FastText sentence vectors (Bojanowski et al., 2017) due to its support for learning vector representations of out-of-vocabulary words, which is useful for our dataset that contains Hindi words in Roman script.

3.5 Complete model - CMSA

As depicted in Fig. 2, the sub-word CNN feeds into a BiLSTM network with attention mechanism that models specific sentiments of the sentence, forming the *Specific Encoder*. The sub-word CNN is also connected to a BiLSTM in parallel that models the overall sentiment of the sentence, forming the *Collective Encoder*. The results from both these encoders are combined by concatenation, which is further combined with a Feature Network based on surface and semantic features, to augment the neural network learning.

A representation the complete model (CMSA) can be seen in Fig. 3. This architecture allows

Original Sentence	English Translation	Sentiment Label
I thought play *accha thha*	I thought that the play was good	**Positive**
Sorrrry , *aaj subah tak pata nhi tha* that I wudnt be able to come today :(	Sorry , Did not know until this morning that I wouldn't be able to come today :(	**Negative**
Trailer *dhannnsu hai bhai*	Trailer is amazing, brother	**Positive**

Table 1: Examples from the dataset

us to capture sentiment on the morpheme, syntactic and semantic levels simultaneously and learn specifically which parts of a sentence provide the most value to its sentiment classification.

With this system, we are able to combine the best of neural networks involving attention mechanisms with surface and semantic features that are used traditionally for sentiment analysis.

4 Experiments

In this section, we describe the dataset used, evaluation of our system with state-of-the-art and traditional approaches along with the parameters used for training the model.

4.1 Dataset

We make use of the dataset released by C16, consisting of 3879 code-mixed English-Hindi sentences that they collected from public Facebook pages popular in India. The posts and comments on those pages attract comments from people across the country with varied sentiment polarity. This dataset contains 15% negative, 50% neutral and 35% positive sentences.

As is evident from examples in Table 1, there are some obvious problems in processing code-mixed data. The sentences are short, lack formally defined grammatical structure, and can contain spelling variations. Therefore, traditional approaches to sentiment analysis do not tend to work for code-mixed data.

4.2 Baseline

We compare our approach with the methods proposed by the following:

- (Wang and Manning, 2012): They use Naive Bayes (NB) and Multinomial Naive Bayes (MNB) methods alongside Support Vector Machines (SVM) and identify how using unigram and bigram models or their combination affects classification accuracy.
- (Pang and Lee, 2008): The use an SVM based approach with ngram models to classify sentences based on their sentiments.
- (Sharma et al., 2015): They perform word-level language identification of the code-mixed sentence, transliterate the English words from Roman to Devanagari and then perform sentiment analysis using a lexicon based approach.
- (C16): Their system uses character embeddings of sentences as input to a Convolutional Neural Network that extracts sub-word level information from the sentence. LSTMs are then used to propagate relevant information.
- (Choudhary et al., 2018): They introduced a framework called *Sentiment Analysis of Code-Mixed Text (SACMT)*, which utilizes siamese networks to map code-mixed and standard sentences to a common sentiment space and then classifies sentiment using contrastive learning.
- (Joulin et al., 2017): Their system, FastText, performs classification tasks by incorporating sub-word information in word vectors and is able to handle out-of-vocabulary words as well. They provide an API for using their system along with pre-trained models for vector representations.

4.3 Parameter Learning

We use an Intel(R) Xeon(R) CPU E5-2658 v3 @ 2.20GHz with 128GB RAM and GeForce GTX

Method	Reported in	Accuracy	F1-score
SVM (Unigram)	(Pang and Lee, 2008)	57.6%	0.5232
SVM (Uni+Bigram)	(Pang and Lee, 2008)	52.96%	0.3773
NBSVM (Unigram)	(Wang and Manning, 2012)	59.15%	0.5335
NBSVM (Uni+Bigram)	(Wang and Manning, 2012)	62.5%	0.5375
MNB (Unigram)	(Wang and Manning, 2012)	66.75%	0.6143
MNB (Uni+Bigram)	(Wang and Manning, 2012)	66.36%	0.6046
MNB (Tf-Idf)	(Wang and Manning, 2012)	63.53%	0.4783
Lexicon Lookup	(Sharma et al., 2015)	51.15%	0.252
Char-LSTM	(C16)	59.8%	0.511
Subword-LSTM	(C16)	69.7%	0.658
FastText	(Joulin et al., 2017)	46.39%	0.505
SACMT	(Choudhary et al., 2018)	77.3%	0.759
CMSA	Proposed	**83.54%**	**0.827**

Table 2: Results show that proposed system performs significantly better over state-of-the-art methods for code-mixed Sentiment Analysis

1080 GPU. The system architecture has been implemented in Keras (Chollet et al., 2015). The dataset is randomly divided into training and validation sets in a 4:1 ratio. The hyperparameters of our model are trained using the validation set. The proposed model and all variants described in this paper are learned by optimizing the validation accuracy. We used a batch size of 128 and used AdaMax (Kingma and Ba, 2014) as the optimizer. We use categorical crossentropy for learning the parameters of the model.

5 Results and Analysis

In this section, we present the results obtained by various experiments that were run with our system.

5.1 Performance comparison with baselines

We compared our method with the baselines. From Table 2, it can be clearly observed that CMSA outperforms the state-of-the-art techniques by 6.24% in accuracy and 0.068 in F1-score.

We can observe a trend in the performances among the baselines themselves. The lexicon lookup approach performs the worst on our dataset among all the methods, likely because of the high number of spelling variations or misspellings in our data that it is unable to handle correctly. Naive Bayes methods perform fairly well, with Multinomial Naive Bayes performing better than simple SVM or SVM in combination with Naive Bayes.

The methods Char-LSTM, Subword-LSTM and SACMT all employ deep learning techniques to perform the task of sentiment analysis, and it can be observed that they perform better than traditional approaches by a fair margin. The significant difference in accuracy between Subword-LSTM and Char-LSTM, as seen in table 2, confirms our assumptions from section 3.2 about subword-level representations being better for this use case as compared to character-level embeddings.

However, deep learning techniques on their own do not suffice for the task. In order to augment these techniques, we involve the use of specific linguistic features, based on real world knowledge of code-mixing. Ultimately, this helps our system learn classification of sentiments better than the above described methods even on fewer training samples, as can be observed from Figure 4 where we compare against Subword-LSTM model which is architecturally most similar to our model but does not use linguistic features or attention.

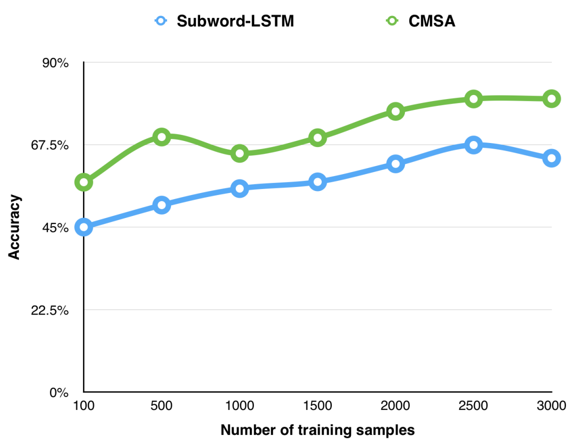

Figure 4: Performance comparison with baseline for varying number of training sentences

5.2 Effect of different Encoders

We note the effects of varying the kind of recurrent network used. We compare the results obtained by using BiLSTMs, LSTMs, GRU and Vanilla RNN. From Table 3, the observable trend in performance is: BiLSTM >LSTM >GRU >RNN, although the differences are not substantial. One of the reasons for this could be that a LSTM or a GRU can handle long term dependencies better than an RNN, and a BiLSTM works better than an LSTM particularly when sentiment information may be embedded towards the beginning of the sentence.

We also experimented with variants of our own model, by replacing the fused encoders with the Specific Encoder and the Collective Encoder separately. From Table 4, we can see that the trend in performance is as follows: CMSA >Collective Encoder >Specific Encoder. This shows that merely learning the overall sentiment embedded in a sentence is not enough, and neither is just learning the sentiments associated with the sub-words themselves. A combination of the two encoders provides a better sentiment classification model.

5.3 Effect of using Feature Network

One of the most distinguishing aspects of our work from previous approaches for this task involves the augmentation of deep learning techniques with a Feature Network. We experimented by performing sentiment analysis using just the deep learning framework, just the feature network and finally, the combination of the two. We report the results in Table 5, where it can be easily observed that the Feature Network is able to positively augment the classification by the neural network.

Encoder	Accuracy	F1-score
RNN	75.91%	0.699
GRU	78.58%	0.768
LSTM	78.5%	0.758
BiLSTM (CMSA)	**83.54%**	**0.827**

Table 3: Performance using different encoders

Method	Accuracy	F1-score
Specific Encoder	80.2%	0.801
Collective Encoder	77.3%	0.795
Specific + Collective (CMSA)	**83.54%**	**0.827**

Table 4: Performance using different encoding mechanisms with Feature Network

System	Accuracy	F1-score
Encoders only	75.74%	0.705
Feature Network only	57.9%	0.381
CMSA	**83.54%**	**0.827**

Table 5: Performance using Feature Network and Encoders separately and combined

6 Conclusion

We have proposed a neural network architecture for sentiment analysis of English-Hindi code-mixed data that improves on traditional and state-of-the-art approaches. We use a hybrid approach that combines recurrent neural networks utilizing attention mechanisms, with surface features, yielding a unified representation that can be trained to classify sentiments. We conducted extensive experiments on a real world dataset consisting of online social media text, and demonstrated that our system is able to achieve a sentiment classification accuracy of 83.54% and an F1-score of 0.827, outperforming state-of-the-art approaches for this task.

References

Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Snchez-Rada, and Carlos A. Iglesias. 2017. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246.

Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas. 2014. i am borrowing ya mixing? an analysis of english-hindi code mixing in facebook. In *In Proceedings of the First Workshop on Computational Approaches to Code Switching, EMNLP. Monica Stella Cardenas-Claros and Neny Isharyanti*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using crf: Code-switching shared task report of msr india system. page 7379, Doha, Qatar. Association for Computational Linguistics.

François Chollet et al. 2015. Keras. https://keras.io.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018. Sentiment analysis of code-mixed languages leveraging resource rich languages.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3):157–174.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Maria Giatsoglou, Manolis G. Vozalis, Konstantinos Diamantaras, Athena Vakali, George Sarigiannidis, and Konstantinos Ch. Chatzisavvas. 2017. Sentiment analysis leveraging emotions and word embeddings. *Expert Systems with Applications*, 69:214 – 224.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *CoRR*, abs/1308.6242.

Carol Myers-Scotton. 1993. Common and uncommon ground: Social and structural factors in codeswitching. *Language in society*, 22(4):475–503.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Braja Gopal Patra, Dipankar Das, and Amitava Das. 2018. Sentiment analysis of code-mixed indian languages: An overview of sail_code-mixed shared task @icon-2017. *CoRR*, abs/1803.06745.

A. Pravalika, V. Oza, N. P. Meghana, and S. S. Kamath. 2017. Domain-specific sentiment analysis approaches for code-mixed social network data. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, volume 00, pages 1–6.

A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D. M. Sharma. 2016. Shallow Parsing Pipeline for Hindi-English Code-Mixed Social Media Text. *ArXiv e-prints*.

Shashank Sharma, PYKL Srinivas, and Rakesh Chandra Balabantaray. 2015. Text normalization of code mix and sentiment analysis. *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1468–1473.

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)*, page 974979, Doha, Qatar. Association for Computational Linguistics.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.