ULD@NUIG at SemEval-2020 Task 9: Generative Morphemes with an Attention Model for Sentiment Analysis in Code-Mixed Text

Koustava Goswami, Priya Rani, Bharathi Raja Chakravarthi,
Theodorus Fransen, and John P. McCrae
Insight SFI Research Centre for Data Analytics,
Data Science Institute, National University of Ireland Galway
{koustava.goswami, priya.rani, bharathi.raja,
theodorus.fransen, john.mccrae}@insight-centre.org

Abstract

Code mixing is a common phenomena in multilingual societies where people switch from one language to another for various reasons. Recent advances in public communication over different social media sites have led to an increase in the frequency of code-mixed usage in written language. In this paper, we present the Generative Morphemes with Attention (GenMA) Model sentiment analysis system contributed to SemEval 2020 Task 9 SentiMix. The system aims to predict the sentiments of the given English-Hindi code-mixed tweets without using word-level language tags instead inferring this automatically using a morphological model. The system is based on a novel deep neural network (DNN) architecture, which has outperformed the baseline F1-score on the test data-set as well as the validation data-set. Our results can be found under the user name "koustava" on the "Sentimix Hindi English"[1] page.

1 Introduction

Sentiment analysis refers to a process of predicting the emotion content from a given text. Sentiment analysis is usually seen as a categorization problem over a variable with three values: positive, negative, neutral (Phani et al., 2016). With the increase in the popularity of social media such as Twitter, a new area of study to the field of natural language processing and thus, sentiment analysis has been explored. Most of the data extracted from social media are code-mixed (Ranjan et al., 2016; Priyadharshini et al., 2020), which have become a common approach in most cases but also pose unique challenges.

Analysis of short texts from micro-blogging platforms such as Twitter is in high demand as the analysis of these text distill and evaluate the moods and the sentiment of the users and are very useful for different organisations, be it government or business or NGO. Sentiment analysis for Indian code-mixed languages is relatively new (Jose et al., 2020; Chakravarthi et al., 2020a; Chakravarthi et al., 2020b; Rani et al., 2020). The significant difference in style of language, orthography (Chakravarthi et al., 2019) and grammar used in tweets presents specific challenges for English-Hindi code-mixed data. In this paper we aim to introduce a novel deep neural network system which was submitted for SemEval 2020 Task 9, Sub Task A for English-Hindi data (Patwa et al., 2020). We will also compare the system with other state-of-the-art systems and describe how the system has outperformed others. The systems were trained using only the Twitter data provided by the organisers excluding the word-level language tags provided in the data.

2 Related Work

Although the field of sentiment analysis is growing and several systems have advanced the state-of-the-art, the overall performance of systems to predict sentiment in code-mixed data is low. Sharma et al. (2015) predicted overall sentiment score for Hindi-English code-mixed data using a lexicon based approach. Go et al. (2009) were the first to look at the task as a query-driven classification problem. A Hindi-English data-set was introduce by Joshi et al. (2016) for sentiment analysis and they performed empirical analysis comparing the performance of various state of the art models in sentiment analysis. They also introduced

[1] `https://competitions.codalab.org/competitions/20654#learn_the_details-results`

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 968–974
Barcelona, Spain (Online), December 12, 2020.

a sub-word level representation in an LSTM model instead of character or word level representation. Dos Santos and Gatti (2014) proposed a deep convolutional neural network that exploits character level and sentence level information to predict the sentiments in short texts. All these previous experiments were dependent on the word-level language tags, and this is a disadvantage as it is time-consuming to annotate at the word level. In our approach, we create a model without the need for word-level annotation.

3 Dataset

The dataset used for the current task is provided by SentiMix English-Hindi Task 9 in SemEval-2020 (Patwa et al., 2020). It consists of English-Hindi code-mixed tweets annotated with sentiment labels: positive, negative, or neutral. Besides the sentiment labels the data-set also includes word-level language tags, which are *en* (English), *hi* (Hindi), *mixed*, and *univ* (symbols, @ mentions, hashtags). As it is very common for Twitter data to have other forms of text such as URLs and emoticons, this data-set too contains emojis such as ☺ ☹ and URLs.

The pre-processing removes the word-level language tags. We normalize the data for training the Support Vector Machine (SVM) and deep neural network (DNN), by lower-casing all the tweets and removing punctuation, emojis and URLs. After converting all the tweets into lower case, extra spaces were removed from the tweets. The tweets are tokenized into characters, where each character has been mapped to an index number. The character-index mapping is created with the help of the Keras tokenizer package[2].

4 System Description

4.1 Support Vector Machine

The Support Vector Machine (SVM) is an algorithm which maximizes a particular mathematical function with respect to a given collection of data (Noble, 2006). In our experiment, we have focused on the linear SVM methodology. The objective of linear SVM optimization problem is to maximize the given equation:

$$max_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i x_j) \tag{1}$$

where α_i is the weight of the examples, *x* is the input and *y* is the label. After pre-processing the data, we experimented with the most basic input feature TF-IDF and was created with the help of TfidfVectorizer[3] of the Scikit Learn package.

4.2 Convolution Neural Network (CNN)

In this experiment, we have followed the CNN model described by Zhang et al. (2015) which has a one-character embedding layer and four convolution (CONV1D) layers. For the first three convolution layers, after each layer, one max-pooling layer has been added. In the end, one hidden layer is followed by one softmax layer. The model accepts sentences as sequence and characters as input. The character embedding is a one-hot embedding (1-to-n embedding) where the number of unique characters is *n*. The shape of the filter is 1-dimensional of size *k*. The filter slides over the input sequence matrix to create the feature map of dimension $b \times f \times s$ where *b* is the batch size, *f* is the number of filters used, and *s* is determined by the formula $m - k + 1$ where *m* is the input size. Stride 1 is used to calculate features based on each character including spaces and special characters.

4.3 Generative Morphemes with Attention (GenMA) Model

We propose an Artificial Morphemes Generative system with Self Attention (SA) layer. The model takes the input sequence as a character sequence. The model has one character embedding layer and two convolution (CONV1D) layers. Each convolution layer has one max-pooling layer each. After the

[2] `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

convolution layers, there is one bidirectional LSTM layer followed by one self-attention layer. The model has one hidden layer and one softmax layer. Liu et al. (2018) designed a network which is based on the Chinese character attention model for sentence classification. We have taken inspiration from that model. Liu et al. (2018) rely on capturing local context of sentences based on a single convolution layer whereas GenMA is capable of generating new artificial morphemes and framing a sentence as a group of new morphemes irrespective of language identification of source words. The combination of two CNN layers helps to generate new morphemes based on deep relative co-occurring characters (3 characters frame), and the LSTM layer helps to capture global information of sentences based on newly generated features. The SA layer helps to construct sentence-level information. It also captures relativity strength among different co-occurring character features. The model architecture can be found in Figure 1.

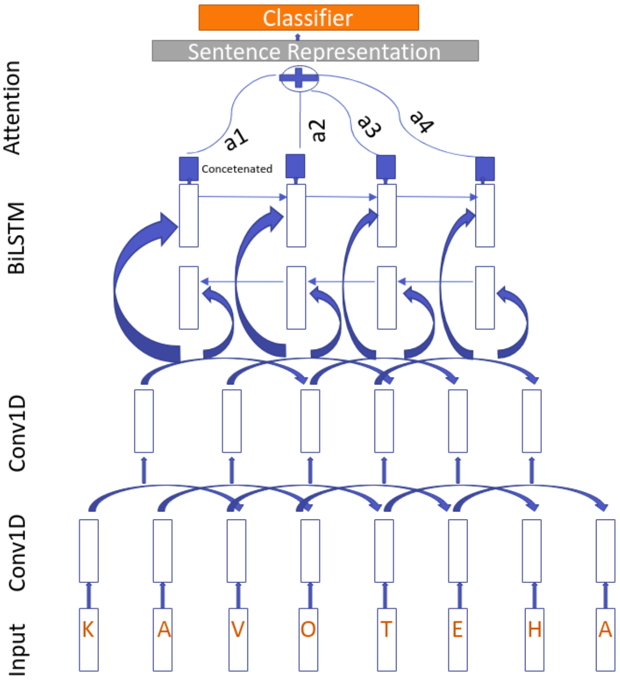

Figure 1: Relative Character Attention Model; the input is the character embeddings which is mentioned with physical characters; boxes filled with blue colour are the concatenated outputs of the BiLSTM.

A Convolution Neural Network layer is used as a feature extractor of the sentences. The one-dimensional convolution implements 1-dimensional filters which slides over the sentences as a feature extractor. Let the filters have a shape of 1 × k where k is the filter size. Let $x_i \in \{0,1\}^n$ denote the one-hot representation of the i-th character considering character vocabulary size is n. For each position j in the sentence, we have a window vector w_j with k consecutive character vectors (Zhou et al., 2015) denoted as

$$w_j = [x_j, x_{j+1}, ..., x_{j+k-1}] \tag{2}$$

The 1-dimensional k-sized filters slide over the window vector w_j to create the feature map s where $s \in \mathbb{R}^{m-k+1}$ and where m is the input size. Multiple filters are used to generate different feature maps for each window w_j. The new feature representation, W_j, will represent a new feature map vector for the j-th position of the sentence. The second convolution layer will take feature representations as input and generate a high-order feature representation of the characters. The max-pooling network after each convolution network helps to capture the most important features of size d. The new high-order representations are then feed to the LSTM (Long Short Term Memory Network) as input.

Long Short Term Memory (LSTM) Network layer takes the output of the previous CNN layer as input. It produces a new representation sequences in the form of $h_1, h_2, ...h_n$ where h_t is the hidden state of the LSTM of time step t, summarising all the information of the input features (morphemes) of the sentences. An LSTM unit is composed of one memory cell and three gates (input gate, forget gate and output gate) (Hochreiter and Schmidhuber, 1997). At each time step *t*, the hidden state takes the previous time step hidden state h_{t-1} and characters (x_t) as input. Let us denote memory cell, input gate, forget gate and output gate as c_t,i_t,f_t,o_t. The output hidden state h_t and the memory cell c_t of timestep *t* is defined by Equation 3

$$\begin{aligned} i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad &, \quad f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad &, \quad f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ c_t = f_t \odot c_{t-1} + i_t \odot q_t \quad &, \quad h_t = o_t \odot \tanh(c_t) \end{aligned} \tag{3}$$

Here $\odot$ is the element wise operation, W_i,W_f,W_o,W_q are the weights of the matrices, b_i,b_f,b_o,b_q are the biases and σ denotes the logistic sigmoid function. A Bidirectional LSTM (BiLSTM) network has been used which has helped us to summarise the information of the features from both directions. The Bidirectional LSTM consists of a forward and backward pass which gives us two annotations of the hidden state h_{for} and h_{back}. We obtained the final hidden state representation by concatenating both the hidden states $h_i = h_{i-for} \oplus h_{i-back}$, where h_i is the hidden state of the *i*-th timestep and $\oplus$ is the element-wise sum between the matrices.

The Attention layer helps us to determine the importance of one morpheme over others while building sentence embeddings for classification. A self-attention mechanism has been adopted from Baziotis et al. (2018) which will help to identify the morphemes that are important for capturing the sentiment of the sentence. The self-attention mechanism is built upon attention mechanism by Bahdanau et al. (2015). The attention mechanism assigns weight a_i to each feature's annotation based on output h_i of the LSTM's hidden states, with the help of the softmax function as illustrated in Equation 4

$$a_i = \tanh(W_h \cdot h_i + b_h) \quad , \quad a_i = \frac{exp(a_i)}{\sum_{t=1}^{T} exp(a_t))} \tag{4}$$

The new representation will give a fixed representation of the sentence by taking the weighted sum of all feature label annotations as shown in Equation 5

$$r = \sum_{i=1}^{T} a_i \cdot h_i \tag{5}$$

where W_h and b_h are the attention weights and bias respectively.

The Output layer consists of one fully-connected layer with one softmax layer. The sentence representation after the attention layer is the input for the dense layer. The output of the dense layer is the input of the softmax which gives the probability distribution of all the classes with the help of the softmax function as shown in Equation 6

$$p_i = \frac{exp(a_i)}{\sum_{t=1}^{T} exp(a_t)} \tag{6}$$

where a_i is the output of the dense layer.

4.4 Parameters

A Linear SVM kernel is used for the first SVM model. Other parameters are kept as mentioned in the implementation (Noble, 2006). For the next two models, the convolution network setup is the same. We have used 32 filters and the kernel size is 3. The maxpooling size is 3. The hidden size h_i of LSTM units is kept to 100. The dense layer has 32 neurons and it has 50 percent dropout. The Adam optimizer (Kingma and Ba, 2015) is used to train our model with the default learning set to 0.0001. The batch size is set to 10. For the convolution layer in both the experiments we have used the relu activation function (Nair and Hinton, 2010) and for the dense layer we have used tanh activation function (Kalman and Kwasny, 1992). Categorical cross entropy loss is used for the multi-class classification. We have used Keras[4] to train and test our model.

[4] `https://keras.io`

5 Results

Overall we see varying performance across the classifier, with some performing much better out-of-sample than others. Table 1 shows the class-wise macro F1-score of the models on the test set for different models.

Model	Pos Class	Neg Class	Neut Class	Score
SVM	0.64	0.62	0.57	0.61
Char-CNN	0.68	0.65	0.56	0.63
GenMA	0.73	0.67	0.63	**0.68**

Table 1: F1-scores of three algorithms on dataset

The state-of-the-art character CNN model has performed better than the SVM model. One of the main reasons for a CNN to perform better than SVM is that a CNN is capable of identifying the features of the sentence with the help of neural model weight distribution. It also takes special characters into account which make the sentence embedding more robust to work on. On the other hand, the hyper-tuning settings of the tf-idf vectors could be the cause of lower performance of the SVM.

Our GenMA model has outperformed all classical models as well as the state-of-the-art character CNN model as it considers a sentence composed of a different set of morphemes. The individual results on three different sentiment classes show that the model outperforms the other two models while recognizing individual classes whereas the SVM model recognizes neutral classes better than the CNN model. Our model has achieved 0.68 F1-score in the test set which is 7 percent better than the SVM and 5 percent better than the character CNN model.

6 Discussion

Our proposed GenMA model outperforms other models as it is capable of generating new morphemes out of neighbor characters and it identifies the essential morphemes to classify a sentence. The two main advantages of our model are:

- The model can construct sentence embeddings based on the new generative morphemes which are created artificially in combination of both the languages Hindi and English. These morphemes carry the features of both the languages. As illustrated in figure 2, the new morpheme **avo** generated by the model, where the character "a" is taken from the Hindi word "ka" and the character "vo" belongs to the English word "vote", shows that these new artificial generative morphemes have features of both Hindi and English. Thus, the multilingual word-level language identification annotations are not required.

- The model is able to correctly identify the co-occurring character sets with highest importance in sentiment analysis. The attention mechanism is visualized in Figure 2. The red characters are the important characters followed by blue characters. The black characters are contributing least significantly to the sentence classification. In the generated artificial morpheme, some morphemes put more emphasis on sentence polarity (An example is morpheme "ote", which weights 5 times (0.061) than the normal morpheme "arn" (0.011)). The softmax attention weights are able to rank character importance from high to low.

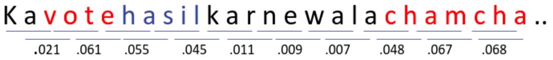

Figure 2: Character of tweets (English-Hindi) with attention

7 Conclusion

In this paper we have proposed a novel deep neural model which has outperformed the baseline scores on code-mixed data proposed in Patwa et al. (2020) and state-of-the-art models discussed in Section 5. Our model is capable of classifying the sentiment of the sentences without considering language difference between words in the sentences with an F1-score of 0.68 on the test data.

Future work may reveal how to capture sentiment based on emojis that are widely used in tweets. One of our settings is artificial morpheme generation for the Hindi and English dataset. But we have not explored this method in the context of morphologically complex code-mixed datasets. We will aim to implement the model in the complex code-mixed dataset in the future. We will also try to capture word level information of code-mixed sentences without language identity to understand what the important key words are to classify sentences.

Acknowledgments

This publication has emanated from research in part supported by the Irish Research Council under grant number IRCLA/2017/129 (CARDAMOM-Comparative Deep Models of Language for Minority and Historical Languages) and co-funded by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 (Insight), SFI/12/RC/2289_P2 (Insight_2).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Christos Baziotis, Athanasiou Nikolaos, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive RNNs. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 613–621, New Orleans, Louisiana, June. Association for Computational Linguistics.

Bharathi Raja Chakravarthi, Mihael Arcan, and John P. McCrae. 2019. Comparison of Different Orthographies for Machine Translation of Under-Resourced Dravidian Languages. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *OpenAccess Series in Informatics (OASIcs)*, pages 6:1–6:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Bharathi Raja Chakravarthi, Navya Jose, Shardul Suryawanshi, Elizabeth Sherly, and John Philip McCrae. 2020a. A sentiment analysis dataset for code-mixed Malayalam-English. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 177–184, Marseille, France, May. European Language Resources association.

Bharathi Raja Chakravarthi, Vigneshwaran Muralidaran, Ruba Priyadharshini, and John Philip McCrae. 2020b. Corpus creation for sentiment analysis in code-mixed Tamil-English text. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 202–210, Marseille, France, May. European Language Resources association.

Cícero Dos Santos and Maíra 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, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Navya Jose, Bharathi Raja Chakravarthi, Shardul Suryawanshi, Elizabeth Sherly, and John P. McCrae. 2020. A survey of current datasets for code-switching research. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 136–141.

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.

Barry L Kalman and Stan C Kwasny. 1992. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zengjian Liu, Xiaolong Wang, Qingcai Chen, and Buzhou Tang. 2018. Chinese clinical entity recognition via attention-based cnn-lstm-crf. In *2018 IEEE International Conference on Healthcare Informatics Workshop (ICHI-W)*, pages 68–69.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

William S Noble. 2006. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567.

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.

Shanta Phani, Shibamouli Lahiri, and Arindam Biswas. 2016. Sentiment analysis of tweets in three Indian languages. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 93–102, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Ruba Priyadharshini, Bharathi Raja Chakravarthi, Mani Vegupatti, and John P. McCrae. 2020. Named entity recognition for code-mixed Indian corpus using meta embedding. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 68–72.

Priya Rani, Shardul Suryawanshi, Koustava Goswami, Bharathi Raja Chakravarthi, Theodorus Fransen, and John Philip McCrae. 2020. A comparative study of different state-of-the-art hate speech detection methods in Hindi-English code-mixed data. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 42–48, Marseille, France, May. European Language Resources Association (ELRA).

Prakash Ranjan, Bharathi Raja, Ruba Priyadharshini, and Rakesh Chandra Balabantaray. 2016. A comparative study on code-mixed data of Indian social media vs formal text. In *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pages 608–611.

Shashank Sharma, PYKL Srinivas, and Rakesh Chandra 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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630*.