Robust Deep Learning Based Sentiment Classification of Code-Mixed Text

Siddhartha Mukherjee, Vinuthkumar Prasan, Anish Nediyanchath, Manan Shah, Nikhil Kumar
Samsung R&D Institute India, Bangalore
{siddhartha.m, vinuth, anish.n, mp.shah, nik.kumar} @samsung.com

Abstract

India is one of unique countries in the world that has the legacy of diversity of languages. English influence most of these languages. This causes a large presence of code-mixed text in social media. Enormous presence of this code-mixed text provides an important research area for Natural Language Processing (NLP). This paper proposes a novel **A**ttention based deep learning technique for Sentiment **C**lassification on **C**ode-**M**ixed **T**ext (ACCMT) of Hindi-English. The proposed architecture uses fusion of character and word features. Non-availability of suitable word embedding to represent these Code-Mixed texts is another important hurdle for this league of NLP tasks. This paper also proposes a novel technique for preparing word embedding of Code-Mixed text. This embedding is prepared with two separately trained word embeddings on romanized Hindi and English respectively. This embedding is further used in the proposed deep learning based architecture for robust classification. The Proposed technique achieves 71.97% accuracy, which exceeds the baseline accuracy.

1 Introduction

Languages used in India belong to several language families. Historical presence of British on Indian soil has led to a very high influence of English language on many of these Indian languages. People belonging in a multi-lingual society of India, gives rise of a large amount of text in various social media (Patra, 2018). Inclusion of English is very common in these texts. Essentially, an utterance in which a user makes use of grammar, lexicon or other linguistic units of more than one language is said to have undergone code-mixing (Chanda, 2016). Hindi is the widely spoken language of India and used in various media. The number of native Hindi speakers is about 25% of the total Indian population; however, including dialects of Hindi termed as Hindi languages, the total is around 44% of Indians, mostly accounted from the states falling under the Hindi belt[1]. This community contributes a large amount of text on social media. The form of Hindi language used in Social Media is mixed with English and are available in roman scripts. According to the study (Dey, 2014) most common reason for this kind of code mixing in a single text is 'Ease of Use'. The code-mixed Hindi and English language poses various types of challenges (Barman, 2014), which makes the text classification task on code-mixed text, an exciting problem in NLP Community. Despite a wide research on classification of code mixed texts, there remains open opportunities with two major aspects; first technique of preparing word embedding on Code-Mixed texts and second utilization of character and word features together to improve the accuracy. This research targets these two open points for exploration.

2 Related Work

Various research works have tried to tackle these challenges. Recent work of Prabhu (2016) utilizes character level LSTMs to learn sub word level information of social media text. Then this information is used to classify the sentences using an annotated corpus. The work is very interesting and achieves good accuracy. However the work does not intend to capture the information related to word level semantics. This provides a further scope of research to study the impact of word

[1]https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 124–129
Hyderabad, India, December 2019.

embedding based approach on classification of code-mixed text. Sharma (2015) used an approach of lexicon lookup for text normalization and sentiment analysis on Code-Mixed text. Pravalika (2017) used lexicon lookup approach for domain specific sentiment analysis. These lexicon lookup based approaches lack capability to handle misspelled words and wide variety of these code mixed texts. Recent work (Lal, 2019) have used BiLSTM based dual encoder networks to represent the character based input and additional feature network to achieve good accuracy on code-mixed texts. Recent work (Yenigalla, 2018) has explored the opportunity of using both character and word embedding based feature to handle unknown words for text classification on monolingual English only text corpora. However, this approach is not common for Code-Mixed text, primarily because of the non-availability of word embedding for the Code-Mixed texts.

3 Dataset

We have considered Hi-En Code-Mixed dataset[2], shared by Prabhu (2016) as a baseline for this research.

3.1 Description

The dataset was collected from public Facebook pages of famous Indian personalities i.e. Salman Khan and Narendra Modi. The data is present in Roman script. The dataset contains 3879 comments. Each data is annotated with a 3-level of polarity scale i.e. Positive, Neutral and Negative. The dataset contains 15% negative, 50% neutral and 35% positive. Table 1 shows some example of code-mixed texts dataset.

Example	Approx. meaning in English	Polarity
Sir yeh tho sirf aap hi kar sakte hai. Great sir	Sir only you can do it. Great Sir	Positive
Kuch nahi karoge tum india ke liye	You won't do anything for India	Negative
Humari sabhayata humari pehchaan ...	Our civilization is our identity	Neutral

Table 1: Example from Hi-En Code-Mixed dataset.

3.2 Challenges

Transliteration of phonetic languages, like Hindi, into roman script creates several variations of the same word. For example, "बहुत" in Hindi which means "more" in English can be transliterated as "bahut", "bohoot" or "bohut" etc.

The Romanized Code-Mixed text, available on social media imposes additional challenges of contraction of phrases. For example, 'awsm' is shortened form of 'awesome'; 'a6a' is contracted from 'accha' etc. Romanized code-mixed text also contain sentences with non-grammatical constructs like 'Bhai jaan bolu naa.. yar' as well as non-standard spelling such as 'youuuu', 'jaaaaan' etc.

The phonetic similarity of various words across participant languages in the Code-Mixed text increases the challenge by introducing disambiguation for meaning of a word. For example, "man" in English means 'an adult human male' where as in Hindi it means 'mind'.

Large availability of clean corpora has given a rise in various kinds of research for Mono-lingual texts like English. On the other hand, the limited availability of clean & standard Code-Mixed corpus restricts wide spectrum of experiments, which depends on word-embedding based input.

3.3 Character Set

The dataset is cleaned of any special characters for this research. Final character set is of 36 characters including 26 English letters and 10 numbers. Final character set is:
abcdefghijklmnopqrstuvwxyz0123456789

4 Proposed Method

The proposed method consists of two major parts. First one is preparing a suitable word-embedding of code-mixed text and later one is a robust deep learning architecture for classification on code-mixed text.

4.1 Word-Embedding

There are three main aspects for preparing word embedding for Hindi-English Code-Mixed Texts. First is preparation of a corpus of Hindi Romanized text. Second one is preparing word embedding by choosing a right algorithm of word embedding.

[2] https://github.com/DrImpossible/Sub-word-LSTM

Third, is to ensure that words from both participant languages which are similar has nearby representation. To address the first aspect, we use Indic transliteration[3] on large Hindi-English corpus[4] where the Hindi text is present in Devanagari[5] script also contains English content. In this way, we achieve the Hindi-English Code-Mixed corpus in Roman Scripts. Figure 1 depicts the process of generating the desired corpus.

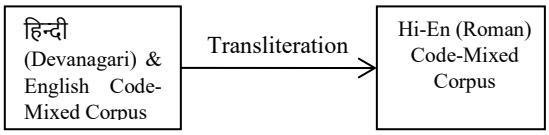

Figure 1: Corpus Preparation for Hi-En Code-mixed Text in Roman Script.

We hypothesize that the transliterated corpus represents a new language of Romanized Hindi. As discussed earlier there are various challenges of Romanized representation of Code-Mixed text such as presence multiple homo-phonic representations of a single word etc., so we have chosen fastText (Bojanowski, 2017) word representation as best method to train word embedding. This addresses the second aspect of previously discussed task of preparing word embedding. Once the corpus is generated, we have trained word embedding with fastText[6]. This trained embedding is capable of providing the vectorized representation of a Romanized Hindi word. On the other side, an utterance in the Code-Mixed corpus also contains English words as well. For example, the 1st utterance in the Table 1 contains two phrases, where 1st phrase contains the Romanized Hindi words and the 2nd phrase contains English words. This is the third and final aspect, discussed as a part of task of word embedding. Now to represent such an utterance using word embedding, we need the bi-lingual word embedding which include Romanized Hindi and English words as well. To cater to this requirement, we have used the proposed method (Smith, 2017) to represent bi-lingual representation of word from two monolingual representations. SVD is used to learn a linear transformation (a matrix), which aligns monolingual vectors from two languages in a single vector space[7]. In this experiment, we considered two monolingual word embedding(s). First is the trained word embedding of Romanized Hindi. Second one is the pre-trained & published[8] English word-embedding (Mikolov, 2018), which is trained on Wikipedia corpus.

4.2 Model Architecture

We prepare **A**ttention based deep learning architecture for **C**lassification of **C**ode-**M**ixed **T**ext (ACCMT) which uses learning from both character and word based representation. The proposed architecture consists of two major parts. The first part learns the sub-word level features from input character sequences. The other parts uses prepared word embedding as input and learn the word level features.

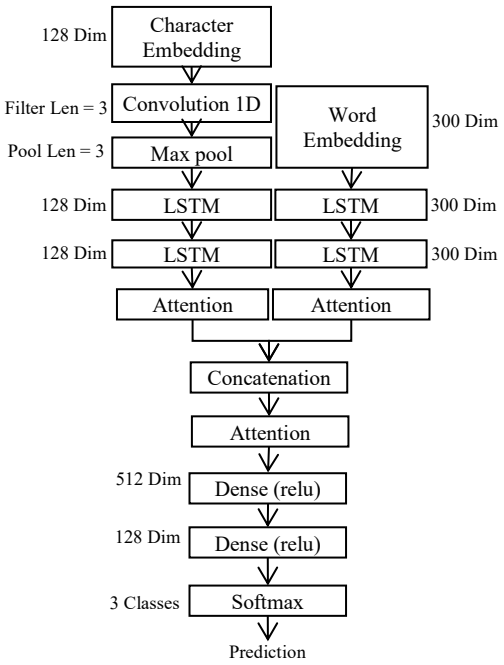

Figure 2: **A**ttention based deep learning architecture for **C**lassification of **C**ode-**M**ixed **T**ext (ACCMT)

The first part is similar as the baseline implementation Prabhu (2016), which is inspired by research work of Kim (2016). This part is independent of word vocabulary, which helps to resolve important issues in code mixed text like non-standard spelling, phrasal contraction etc.

[3] https://github.com/sanskrit-coders/indic_transliteration
[4] https://www.kaggle.com/pk13055/code-mixed-hindienglish-dataset
[5] https://en.wikipedia.org/wiki/Devanagari
[6] https://fasttext.cc/docs/en/python-module.html
[7] https://github.com/Babylonpartners/fastText_multilingual
[8] https://fasttext.cc/docs/en/pretrained-vectors.html

Even though this representation lack word level semantic interpretability, the assumption is that character n-gram serve semantic functions e.g. 'cat+s=cats'.

Formally a Sentence S is made of sequence of characters $[c_1, \dots, c_l]$where l is sentence length. $Q \in \mathbb{R}^{d \times l}$ is the representation of sentence where d being the dimension of character embedding. We perform the convolution of Q with filter $H \in \mathbb{R}^{d \times m}$ of length m. This operation provides a feature map $f \in \mathbb{R}^{l-m+1}$. Convolution is shown with '$*$' Operator in equation 1.

$$f = Q * H \tag{1}$$

Next max-pool operation of p features from f brings sub-word representation y.

$$\begin{aligned} a^t &= \tau_0 \times \tanh\left(\tau_u \widetilde{C_t} + \tau_f \widetilde{C_{t-1}}\right) \\ Where, \widetilde{C_t} &= \tanh(W_c[a^{t-1}, y^t] + b_c) \\ \tau_o &= \sigma(W_o[a^{t-1}, y^t] + b_o) \\ \tau_u &= \sigma(W_u[a^{t-1}, y^t] + b_u) \\ \tau_f &= \sigma(W_f[a^{t-1}, y^t] + b_f) \end{aligned} \tag{2}$$

Here y^t represents the input at current timestamp. Output from LSTM is a^tat time t. τ_o, τ_u, τ_f are respectively the output, input and forget gates of LSTM cell. $\widetilde{C_t}$ is the cell state at time t.

The second part is designed with intention to capture features for the word level semantic representation to counter the limitation of previous part of the architecture. For this purpose LSTM is used as well, because LSTM has performed very well (Bhasin, 2019; Tang, 2015) in various sentiment analysis and other text processing tasks. Formally a Sentence S is made of sequence of words $[p_1, \dots, p_l]$ where l is word length of S. $Q \in \mathbb{R}^{d \times l}$ is the representation of sentence where d being the dimension of word embedding. Now p_t, word at time t is passed to memory cell of LSTM and the output follows similar of equation (2).

We have introduced two separate attention layers over the LSTM output of Character based side and Word based side respectively. The intention of applying the attention is to infer the dominating features from character representation as well as word representation respectively. We have used self attention (Vaswani, 2017) for our implementation[9]. Formally, the attention can be depicted as equation (3).

$$Attention(Q, K, V) = softmax(QK^T/d_k) \tag{3}$$

The Q, K & V is same and that is the output of the previous layer. The final output after attention of sub-word level representation through character embedding part and learnt features from the word embedding part are concatenated as late fusion to feature represent of the input sentence. The joint feature is passed through another attention layer. This layer is intend to figure out the dominating learnt feature among word and character based learnt features. Following this layer, we add two consecutive fully connected layers with ReLU non-linearity. The final output of the last dense layer is passed through a Softmax layer to predict the sentiment.

Formally late fusion of learnt character features f_c & word features f_w is $f_s = (f_c, f_w)$ to represent jointly learnt features of sentence S. Then $_s$ is input to dense layers with g as ReLU non-linearity. Output a_1 is passed through second dense layer to get output a_2.

$$\begin{aligned} a_1 &= g(W_1 \times f_s + b_1) \\ a_2 &= g(W_2 \times a_1 + b_2) \end{aligned} \tag{4}$$

Further, final layer is formalized as equation 5.

$$\sigma = e^{a_2} / \sum e^{a_2}{}_i \tag{5}$$

5 Experimental Setup

This research used Keras on python for all required implementations. The baseline dataset is divided into 3 splits i.e. training, validation and testing. Initially the dataset is randomly divided into 80-20 train-test split. Further train is randomly divided into 90-10 train-validation unlike the baseline implementation which splits 80-20 as train-validation. The results are reported over the test split here.

We have experimented with various possible values of hyper parameters and the best set of hyper parameters is shown in the Fig 2. As discussed earlier first part of the architecture is meant for character based input. Here a single

[9] https://pypi.org/project/keras-self-attention/

sentence is considered to be of sequence of 200 characters. Characters beyond 200 are ignored for sentence having more than 200 characters. A sentence with less than 200 characters is zero padded. Point need to mention is that we have considered space also as valid character input. For the second part of the network we have use word embedding of different dimensions for example 100, 200 and 300. However it achieved best accuracy with 300 dimensional word-embedding. While training the fastText Word-Embedding, 'minn' & 'maxn' parameters were set to 2 and 10 respectively. For word based input, a sentence of length 40 words is considered. A sentence with lesser than 40 words is zero embedding padded whereas words beyond 40 are ignored if sentence is having more than 40 words. Also we empirically found that having two stacked LSTM layers similar to Prabhu (2016) gave optimal performance.

We have used default Keras implementations of Categorical Cross Entropy for loss functions in different experiments. Available implementation of Focal Loss[10] is used during few experiments. The intention of apply focal loss (Lin, 2017) is to check the robustness of the proposed ACCMT architecture with respect to different loss function. Of late Focal Loss has migrated from Object Detection to various other tasks, for example speech emotion recognition Tripathi (2019) etc. We wanted to experiment and capture the impact of Focal Loss on Classification of Code-Mixed text. Default Keras implementation for adam optimizer is used for experiments. On the other hand learning rate of 0.0008 and a decay of 0.000012 is set for RMS Prop in various set of experiments. Dropout at Character-LSTM part is set to 0.2 and Word-LSTM is set to 0.4, where as the dropout of dense layers are set to 0.4. We have used the available implementation of attention layer in our code for model architecture.

The model is trained over 50 epochs and batch size of 64 with 10-fold cross-validation. During each fold, the best model is picked based on validation accuracy. The experiments are conducted in the Anaconda environment on a machine with Intel Core i5 processor and NVIDIA processor for GPU acceleration, 16 GB of RAM and a 1 TB of HDD with Windows 10 Operating System. The 50 epochs of training of ACCMT takes 25 minutes in average.

6 Results and Analysis

We have conducted all experiments in the computing environment mentioned in above section. In the same environment, the implementation of Prabhu (2016) attained maximum accuracy of 66.29% across 5 different executions. Whereas the best performance of ACCMT is 71.97% exceeds the baseline performance by 5.68% in the same computing environment. To understand the impact of attention on the classification of code-mixed text, we have also experimented without attention. We have removed three attention layers from the ACCMT and created a deep learning architecture which uses only fusion of character and word features. This architecture showed a maximum of 69.845% accuracy on the same dataset. This implies that attention has improved accuracy with 2.125%. We also compared against Yenigalla (2018) which gave an accuracy of 64.3%. Table 2 showed the accuracy and F1 score of all experiments.

Experiments	***Results***	
	Accuracy	***F1***
Yenigalla (2018)	64.3%	62.2
ACCMT (adamax + Focal Loss)	70.10%	68.1
ACCMT (RMS prop + categorical cross entropy)	69.75%	67.5
ACCMT (adamax + categorical cross entropy)	**71.97%**	**70.93**
ACCMT (RMS Prop + Focal Loss)	70.32%	68.71

Table 2: Results of ACCMT on Hi-En Code Mixed dataset with different loss-function and initializers.

7 Conclusion

This paper shows the architecture of attention based deep learning architecture (ACCMT) which does fusion of character and word feature to develop a robust classifier for code-mixed text. The proposed ACCMT architecture performs well on the Hi-En code-mixed dataset and outperforms the baseline accuracy. A major contribution of this paper is the technique of training word embedding for code-mixed text. This technique is used for

[10] https://github.com/mkocabas/focal-loss-keras

generating word embedding for Hindi-English code mixed corpus, which is required in this research work. This proposed technique is very easy to implement for other code-mixed languages as well and will be helpful for generating word embedding for low resource code-mixed languages majorly Indian languages e.g. Bengali, Tamil and Malayalam etc. This also opens up opportunities of research on other code-mixed languages. This work also shows the impact of attention for the classification of code-mixed text. Lal (2019) showed that introduction of feature network has improved the accuracy significantly. The integration of such feature network in ACCMT is considered for future course of improvement for the on-going research.

References

Patra, B.G., Das, D. and Das, A., 2018. *Sentiment Analysis of Code-Mixed Indian Languages: An Overview of SAIL_Code-Mixed Shared Task@ ICON-2017*. arXiv preprint arXiv:1803.06745.

Chanda, Arunavha, Dipankar Das, and Chandan Mazumdar. *Unraveling the English-Bengali code-mixing phenomenon.* In Proceedings of the Second Workshop on Computational Approaches to Code Switching, pp. 80-89. 2016.

Dey, A. and Fung, P., 2014, May. *A Hindi-English Code-Switching Corpus*. In LREC (pp. 2410-2413)

Barman, U., Das, A., Wagner, J. and Foster, J., 2014. *Code mixing: A challenge for language identification in the language of social media*. In Proceedings of the first workshop on computational approaches to code switching (pp. 13-23).

Prabhu, A., Joshi, A., Shrivastava, M. and Varma, V., 2016. *Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text*. arXiv preprint arXiv:1611.00472.

Yenigalla, P., Kar, S., Singh, C., Nagar, A., & Mathur, G. (2018, June). *Addressing unseen word problem in text classification*. In International Conference on Applications of Natural Language to Information Systems (pp. 339-351). Springer, Cham.

Lal, Y.K., Kumar, V., Dhar, M., Shrivastava, M. and Koehn, P., 2019, July. *De-Mixing Sentiment from Code-Mixed Text*. In Proceedings of the 57th Conference of the Association for Computational Linguistics: Student Research Workshop (pp. 371-377).

Bhasin, A., Natarajan, B., Mathur, G., Jeon, J.H. and Kim, J.S., 2019, June. *Unified Parallel Intent and Slot Prediction with Cross Fusion and Slot Masking*. In International Conference on Applications of Natural Language to Information Systems (pp. 277-285). Springer, Cham.

Sharma, S., Srinivas, P. Y. K. L., & Balabantaray, R. C. (2015, August). *Text normalization of code mix and sentiment analysis.* In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1468-1473). IEEE.

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

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching word vectors with subword information*. Transactions of the Association for Computational Linguistics, 5, 135-146.

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016, March). *Character-aware neural language models* In Thirtieth AAAI Conference on Artificial Intelligence.

Tang, D., Qin, B., & Liu, T. (2015, September). *Document modeling with gated recurrent neural network for sentiment classification*. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1422-1432).

Smith, S.L., Turban, D.H., Hamblin, S. and Hammerla, N.Y., 2017. *Offline bilingual word vectors, orthogonal transformations and the inverted softmax*. arXiv preprint arXiv:1702.03859.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. and Joulin, A., 2018, May. *Advances in Pre-Training Distributed Word Representations*. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. *Attention is all you need*. In Advances in neural information processing systems (pp. 5998-6008).

Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

Tripathi, S., Kumar, A., Ramesh, A., Singh, C. and Yenigalla, P., 2019. *Focal Loss based Residual Convolutional Neural Network for Speech Emotion Recognition*. arXiv preprint arXiv:1906.05682.