De-Mixing Sentiment from Code-Mixed Text

Yash Kumar Lal† Vaibhav Kumar‡ Mrinal Dhar★ Manish Shrivastava○ Philipp Koehn†
†Johns Hopkins University, ‡Carnegie Mellon University, ★Bloomberg LP,
○International Institute of Information Technology
{yash, phi}@jhu.edu, vaibhav2@andrew.cmu.edu, mrinaldhar@gmail.com,
manish.shrivastava@iiit.ac.in

Abstract

Code-mixing is the phenomenon of mixing the vocabulary and syntax of multiple languages in the same sentence. It is an increasingly common occurrence in today’s multilingual society and poses a big challenge when encountered in different downstream tasks. In this paper, we present a hybrid architecture for the task of Sentiment Analysis of English-Hindi code-mixed data. Our method consists of three components, each seeking to alleviate different issues. We first generate subword level representations for the sentences using a CNN architecture. The generated representations are used as inputs to a Dual Encoder Network which consists of two different BiLSTMs - the Collective and Specific Encoder. The Collective Encoder captures the overall sentiment of the sentence, while the Specific Encoder utilizes an attention mechanism in order to focus on individual sentiment-bearing sub-words. This, combined with a Feature Network consisting of orthographic features and specially trained word embeddings, achieves state-of-the-art results - 83.54% accuracy and 0.827 F1 score - on a benchmark dataset.

1 Introduction

Sentiment Analysis (SA) is crucial in tasks like user modeling, curating online trends, running political campaigns and opinion mining. The majority of this information comes from social media websites such as Facebook and Twitter. A large number of Indian users on such websites can speak both English and Hindi with bilingual proficiency. Consequently, English-Hindi code-mixed content has become ubiquitous on the Internet, creating the need to process this form of natural language.

Code-mixing is defined as "the embedding of linguistic units such as phrases, words and mor-

*Equal Contribution
training examples as compared to established approaches.

We perform extensive experiments to evaluate the effectiveness of this system in comparison to previously proposed approaches and find that our method outperforms all of them for English-Hindi code-mixed sentiment analysis, reporting an accuracy of 83.54% and F1 score of 0.827. An ablation of the model also shows the effectiveness of the individual components.

2 Related Work

Mohammad et al. (2013) employ surface-form and semantic features to detect sentiments in tweets and text messages using SVMs. Keeping in mind a lack of computational resources, Giatseglo et al. (2017) came up with a hybrid framework to exploit lexicons (polarized and emotional words) as well as different word embedding approaches in a polarity classification model. Ensembling simple word embedding based models with surface-form classifiers has also yielded slight improvements (Araque et al., 2017).

Extending standard NLP tasks to code-mixed data has presented peculiar challenges. Methods for POS tagging of code-mixed data obtained from online social media such as Facebook and Twitter has been attempted (Vyas et al., 2014). Shallow parsing of code-mixed data curated from social media has also been tried (Sharma et al., 2016). Work has also been done to support word level identification of languages in code-mixed text (Chittaranjan et al., 2014).

Sharma et al. (2015) tried an approach based on lexicon lookup for text normalization and sentiment analysis of code-mixed data. Pravalika et al. (2017) used a lexicon lookup approach to perform domain specific sentiment analysis. Other attempts include using sub-word level compositions with LSTMs to capture sentiment at morpheme level (Joshi et al., 2016), or using contrastive learning with Siamese networks to map code-mixed and standard language text to a common sentiment space (Choudhary et al., 2018).

3 Model Architecture

An overview of this architecture can be found in Figure 3. Our approach is built on three components. The first generates sub-word level representations for words using Convolutional Neural Networks (CNNs). This produces units that are more atomic than words, which serve as inputs for a sequential model.

In Section 3.2, we describe our second component, a dual encoder network made of two Bi-directional Long Short Term Memory (BiLSTM) Networks that: (1) captures the overall sentiment information of a sentence, and (2) selects the more important sentiment-bearing parts of the sentence in a differential manner.

Finally, we introduce a Feature Network, in Section 3.3, built on surface features and a monolingual vector representation of the sentence. It augments our base neural network to boost classification accuracy for the task.

3.1 Generating Subword Representations

Word embeddings are now commonplace but are generally trained for one language. They are not ideal for code-mixed data given the transcription of one script into another, and spelling variations in social media data. As a single character does not inherently provide any semantic information that can be used for our purpose, we dismiss character-level feature representations as a possible choice of embeddings.

Keeping in mind the fact that code-mixed data has innumerable inconsistencies, we use characters to generate subword embeddings (Joshi et al., 2016). This increases the robustness of the model, which is important for noisy social media data. Intermediate sub-word feature representations are learned by filters during the convolution operation.

Let $S$ be the representation of the sentence. We generate the required embedding by passing the characters of a sentence individually into 3 layer 1-D CNN. We perform a convolution of $S$ with a filter $F$ of length $m$, before adding a bias $b$ and applying a non-linear activation function $g$. Each such operation generates a sub-word level feature
Here, $S[i:i+m-1]$ is a matrix of $(i)^{th}$ to $(i+m-1)^{th}$ character embeddings and $g$ is the ReLU activation function. Thereafter, a final max-pooling operation over $p$ feature maps generates a representation for individual subwords:

$$sw_i = \max(f_{sw}[p \cdot (i : i+p-1)])$$

A graphical representation of the architecture can be seen in Figure 1

### 3.2 Dual Encoder

We utilize a combination of two different encoders in our model.

#### 3.2.1 Collective Encoder

The collective encoder, built over a BiLSTM, aims to learn a representation for the overall sentiment of the sentence. A graphical representation of this encoder is in Figure 2(A). It takes as input the subword representation of the sentence. The last hidden state of the BiLSTM i.e. $h_t$, encapsulates the overall summary of the sentence’s sentiment, which is denoted by $cmr_c$.

#### 3.2.2 Specific Encoder

The specific encoder is similar to the collective encoder, in that it takes as input a subword representation of the sentence and is built over LSTMs, with one caveat - an affixed attention mechanism. This allows for selection of subwords which contribute the most towards the sentiment of the input text. It can be seen in Figure 2(B).

Identifying which subwords play a significant role in deciding the sentiment is crucial. The specific encoder generates a context vector $cmr^s$ to this end. We first concatenate the forward and backward states to obtain a combined annotation $(k_1, k_2, \ldots, k_t)$. Taking inspiration from (Yang et al., 2016), we quantify the significance of a subword by measuring the similarity of an additional hidden representation $u_i$ of each sub-word against a sub-word level context vector $X$. Then, a normalized significance weight $\alpha_i$ is obtained.

$$u_i = \tanh(W_i k_i + b_i)$$

$$\alpha_i = \frac{\exp(u_i^T X)}{\sum \exp(u_i^T X)}$$

The context vector $X$ can be looked at as a high-level representation of the question "is it a significant sentiment-bearing unit" evaluated across the sub-words. It is initialized randomly and learned jointly during training. Finally, we calculate a vector $cmr^s$ as a weighted sum of the sub-word annotations.

$$cmr^s = \sum \alpha_i k_i$$

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

#### 3.2.3 Fusion of the Encoders

We concatenate the outputs obtained from both these encoders and use it as inputs to further fully connected layers. Information obtained from both the encoders is utilized to come up with a unified representation of sentiment present in a sentence,

$$cmr^s_{sent} = [cmr^c; cmr^s]$$

where $cmr^s_{sent}$ represents the unified representation of the sentiment. A representation of the same can be found in Figure 3.

### 3.3 Feature Network

We also use linguistic features to augment the neural network framework of our model.

- Capital words: Number of words that have only capital letters
- Extended words: Number of words with multiple contiguous repeating characters.
- Aggregate positive and negative sentiments: Using SentiWordNet (Esuli and Sebastiani, 2006) for every word bar 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 punctuation.
- Exclamation at end of sentence: Boolean value.
- Monolingual Sentence Vectors: Bojanowski et al. (2017)’s method is used to train word vectors for Hindi words in the code-mixed sentences.
3.4 CMSA

The entire model is shown in Figure 3. Sub-word representations are fed into both the specific and the collective encoder. The outputs of the encoders are concatenated with each other, and further with the result of the Feature Network. Subsequently, these are passed through multiple fully connected layers to make the prediction. This architecture allows us to capture sentiment on the morphemic, syntactic and semantic levels simultaneously and learn which parts of a sentence provide the most value to its sentiment.

This system enables us to combine the best of neural networks involving attention mechanisms with surface and semantic features that are traditionally used for sentiment analysis.

4 Experiments And Results

4.1 Dataset

We use the dataset released by (Joshi et al., 2016). It is made up of 3879 code-mixed English-Hindi sentences which were collected from public Facebook pages popular in India.

4.2 Baselines

We compare our approach with the following:

- **SVM** (Pang and Lee, 2008): Uses SVMs with ngrams as features.
- **NBSVM** (Wang and Manning, 2012): Uses Naive Bayes and SVM with ngrams as features.
- **MNB** (Wang and Manning, 2012): Uses Multinomial Naive Bayes with various features.
- **Lexicon Lookup** (Sharma et al., 2015): The code-mixed sentences are first transliterated from Roman to Devanagari. Thereafter, sentiment analysis is done using a lexicon based approach.
- **Char-LSTM and Sub-word-LSTM** (Joshi et al., 2016): Character and sub-word level
embeddings are passed to LSTM for classification.

- **FastText** (Joulin et al., 2017): Word Embeddings utilized for classification. It has the ability to handle OoV words as well.

- **SACMT** (Choudhary et al., 2018): Siamese networks are used to map code-mixed and standard sentences to a common sentiment space.

5 Results and Analysis

From Table 1, it is clear that CMSA outperforms the state-of-the-art techniques by 6.24% in accuracy and 0.068 in F1-score. We observe that sentence vectors (FastText) alone are not suitable for this task. A possible reason for this is the presence of two different languages.

There is a significant difference in accuracy between Subword-LSTM and Char-LSTM, as seen in Table 1, which confirms our assumption about subword-level representations being better for this task as compared to character-level embeddings. Amongst the baselines, SACMT performed the best. However, mapping two sentences into a common space does not seem to be enough for sentiment classification. With a combination of different components, our proposed method is able to overcome the shortcomings of each of these baselines and achieve significant gains.

5.1 Effect of different Encoders

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific Encoder</td>
<td>80.2%</td>
<td>0.801</td>
</tr>
<tr>
<td>Collective Encoder</td>
<td>77.3%</td>
<td>0.795</td>
</tr>
<tr>
<td>Specific + Collective (CMSA)</td>
<td>83.54%</td>
<td>0.827</td>
</tr>
</tbody>
</table>

Table 2: Different encoding mechanisms with Feature Network

We experiment with different encoders used in the dual encoder network. From Table 2, we can see that CMSA > Collective Encoder > Specific Encoder, for both accuracy and F1 score. A combination of the two encoders provides a better sentiment classification model which validates our initial hypothesis of using two separate encoders.

5.2 Effect of Different Model Components

<table>
<thead>
<tr>
<th>System</th>
<th>Accuracy</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Encoder</td>
<td>75.74%</td>
<td>0.705</td>
</tr>
<tr>
<td>Feature Network only</td>
<td>57.9%</td>
<td>0.381</td>
</tr>
<tr>
<td>CMSA</td>
<td>83.54%</td>
<td>0.827</td>
</tr>
</tbody>
</table>

Table 3: Effect of different model components

From Table 3, we see the performance trend as follows: CMSA > Dual Encoder > Feature Network. Although the feature network alone does not result in better overall performance, it is still comparable to Char-LSTM (Table 2). However, the combination of feature network with dual encoder helps in boosting the overall performance.

5.3 Effect of Training Examples

From Figure 4, it is clear that CMSA outperforms the state-of-the-art techniques by 6.24% in accuracy and 0.068 in F1-score. We observe that sentence vectors (FastText) alone are not suitable for this task. A possible reason for this is the presence of two different languages.

There is a significant difference in accuracy between Subword-LSTM and Char-LSTM, as seen in Table 1, which confirms our assumption about subword-level representations being better for this task as compared to character-level embeddings. Amongst the baselines, SACMT performed the best. However, mapping two sentences into a common space does not seem to be enough for sentiment classification. With a combination of different components, our proposed method is able to overcome the shortcomings of each of these baselines and achieve significant gains.

Figure 3: Complete Architecture of CMSA.
We experimented with varying numbers of training examples used for learning the model parameters. Different models were trained on 500, 1000, 1500 and 2000 examples. We observed that the performance of CMSA was greater than Subword-LSTM and SACMT at each point. This can be seen in Figure 4. One of the reasons for the same is the feature network in CMSA which helps in better performance even with lesser number of training examples.

6 Conclusion

We propose 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 conduct extensive experiments on a real world code-mixed social media dataset, and demonstrate that our system is able to achieve an accuracy of 83.54% and an F1-score of 0.827, outperforming state-of-the-art approaches for this task. In the future, we’d like to look at varying the attention mechanism in the model, and evaluating how it performs with a larger training set.

References


