Sentiment Analysis for Code Mixed Social Media Text

Group 9

Ayush Kumar (170195)
Harsh Agarwal (170287)
Keshav Bansal (170335)
Predict the sentiments of code-mixed tweets.

The sentiment labels are Positive, Neutral and Negative.

Code mixed languages are Hinglish and Spanglish.

Measure the efficiency of the model in terms of Precision, Recall and F-measure
TERMINOLOGY

- Code-Mixing: Mixing languages while writing text. A typical pattern observed in social media text.
  - Hinglish: Hindi-English code-mixing
  - Spanglish: Spanish-English code-mixing
- Sentiment Analysis: The problem of classifying human affection and emotion.
- Opinion Mining: Extracting subjective information using Sentiment Analysis.
MOTIVATION

- Interpreting code-mixed languages is difficult because sentences might not fit a particular language model.
- Mixed text contains tokens such as hashtags, emojis and misspelled words.
- The most popularly used languages on Twitter are code-mixed.
- Twitter serves as the most common corpora for Opinion Mining regarding political and nationalist sentiments.
More than 50% of tweets are in languages other than English. Most of the other languages are used in combination with English.

Need a way to process these code-mixed texts.
Subword Level LSTM (Joshi et al.):

- Introduced learning sub-word level representations using CNN and LSTM (Subword-LSTM) architecture instead of character-level or word-level representations.
- This is useful for noisy text containing misspellings.
PREVIOUS WORK

- CSMA(Lal et al.):
  - The first layer generates sub-word level representations for words using Convolutional Neural Networks (CNNs).
  - The second component, a dual encoder network made of two Bidirectional Long Short Term Memory (BiLSTM) Networks that captures the overall sentiment information of a sentence, and selects the more important sentiment-bearing parts of the sentence in a differential manner.
  - Third layer, feature Layer uses linguistic features to augment the neural network framework for final classification.
COMPARISON OF PREVIOUS WORK

Accuracy

FAST TEXT
CHAR-LSTM
SUBWORD-LSTM
CSMA

0  22.5  45  67.5  90
Pre Processing:

- Back-Transliteration: We use the language tags (Hin and Eng) made available to us in the data to perform back transliteration of words. Google Transliterate api was used to perform back transliteration of romanised Hindi words to Devanagari.

- Removal of @user and urls from the tweets.

- Remove noisy tweets (comprising only of hashtags or urls).

- At present we remove the emoji from the sentences. We had experimented with converting them into text (For eg. 😊 -> smiling_face ) but we obtained better results by removing them altogether. This can be attributed to the inherent vagueness and subjective nature of emojis.
#holidays @harsh chuttiyon ka kya plan hai? Let's get together http://gtu.com

Transliteration

#holidays @harsh छुट्टियों का क्या plan है? Let's get together http://gtu.com

Noise Removal

<hashtag> <username> छुट्टियों का क्या plan है? Let's get together <url>

Subwords

['छ', '##टट', '##ियो', 'का', 'क्या', 'plan', 'ह', '?', 'let', ''', 's', 'get', 'together']
PRE PROCESSING PIPELINE

Tokens to IDs

Word Embeddings

Classifier

Neutral


[553, 75864, 19911, 11263, 45731, 12289, 580, 136, 12421, 112, 161, 13168, 13627]

[0.43, .98, 0.04, 0.21, .., .., …]
[0.11, .72, 0.55, 0.20, .., .., …]
[0.32, .64, 0.08, 0.16, .., .., …]
[.., .., .., .., .., .., .., .., .., .., ……..]
[.., .., .., .., .., .., .., .., .., .., ……..]
In all our models, we use BERT Multilingual (M-Bert) for encoding the sentences into vectors.

M-BERT injects audio-visual information into the pre-trained BERT model.

We also experimented with other variants of Bert such as Roberta and DistilBert. We found that M-Bert performed better than other variants for our data.
**APPROACH - 1**

- **BERT - GRU**
  - M-Bert embeddings are passed to a bidirectional GRU. The outputs from the forward and backward layer is concatenated and passed to a dense feed forward layer, which is followed by a softmax layer.
  - GRU helps to model long term dependencies in the text.
  - About 5 times slower than CNNs.
BERT-GRU ARCHITECTURE

Combination of Dense and Dropout Layers

RNN OUTPUT

BACKWARD LAYER

FORWARD LAYER

WORD EMBEDDINGS

INPUTS

THE

QUICK

BROWN

FOX
PERFORMANCE OF GRU

Accuracy

Epochs

Train Acc

Validation Acc

Accuracy

Epochs
This model works relatively better for neutral tweets, but misclassifies many of the positive and negative tweets as neutral.
The M-Bert Embeddings are fed to 3 parallel CNN layers of different filter sizes (2, 3 and 5), followed by a MaxPool layer. The output from the different layers are concatenated and then passed through a dense feed forward layer, followed by a softmax layer.

By varying the size of the filters and concatenating their outputs, we’re allowing the model to detect patterns of multiples sizes.

Works well in detecting local and position invariant features such as negation, positive remarks, etc.
BERT-CNN ARCHITECTURE

Sentence matrix $7 \times 5$

3 region sizes: (2,3,4)
2 filters for each region size
totally 6 filters

2 feature maps for each region size

6 univariate vectors concatenated together to form a single feature vector

1-max pooling

softmax function regularization in this layer

3 classes

I like this movie very much!

d=5

activation function

convolution

3 classes
PERFORMANCE OF CNN

<table>
<thead>
<tr>
<th>Epochs</th>
<th>Train Acc</th>
<th>Validation Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The model performs similar to the GRU model on neutral tweets, but is slightly better for positive and negative tweets.
APPROACH – 3

- **BERT SELF ATTENTION (Inspired from CSMA):**
  - The M-Bert embeddings are passed through the specific encoder. The output from this is passed through a dense feed-forward layer followed by softmax.
  - The Specific Encoder utilises a self-attention mechanism that focuses on individual sentiment bearing units. This helps in choosing the correct sentiment when presented with a mixture of sentiments.
BERT SELF ATTENTION ARCHITECTURE

\[ cmr^s = \sum_{i=1}^{t} a_i k_i \]

- Representation of the Specific Sentiment of the Sentence
- Attention Weights
- BiLSTM Layer
- Sub-word Representations
PERFORMANCE OF SELF ATTENTION

Accuracy

Epochs

Train Acc

Validation Acc

Accuracies over epochs:
- Train Accuracy:
  - Epoch 1: 40
  - Epoch 2: 50
  - Epoch 3: 60
  - Epoch 4: 70
  - Epoch 5: 80

- Validation Accuracy:
  - Epoch 1: 45
  - Epoch 2: 55
  - Epoch 3: 65
  - Epoch 4: 75
  - Epoch 5: 85
We can see that many of the neutrals get wrongly classified as positive and negative. Performance on positive and negative is better.
MOTIVATION FOR USING ENSEMBLE MODELS

- From the data of confusion matrix it is evident that CNN performs better than Self-Attention for neutral tweets while Self Attention performs better for positive and negative tweets.

- These results motivated us to create an ensemble of CNN and Self Attention.
**APPRAOCH - 4**

- **CNN-SELF ATTENTION ensemble**
  - Ensemble models created using 3 different aggregation techniques:
    1. **Sum**: Prediction probabilities of SA and CNN are summed and then fed to a cross-entropy loss.
    2. **Product**: Prediction probabilities of SA and CNN are multiplied and then fed to a cross-entropy loss.
    3. **Max**: Max of the prediction probabilities of each class are fed to the loss functions.
  - In practice, Sum and Max aggregation techniques seemed to work better than Product.
ENSEMBLE MODEL

Word Embeddings

Self-Attention

Predictor 1

Aggregate

Predictor 2

CNN
The model performs better than CNN for positive and neutral tweets. It also performs better than Self Attention for neutral tweets.
We visualised the sentence representations using T-distributed Stochastic Neighbour Embedding (t-SNE) algorithm which reveals that our model is able to differentiate positive and negative sentiments effectively. There are two distinct clusters corresponding to positive and negative tweets. The neutral tweets are distributed evenly among the positive and negative ones.

SENTENCE EMBEDDINGS OF LEARNED MODEL

Sentences in train data
(〜14000 sentences)

Sentences in test data
(〜3000 sentences)
DATA ANALYSIS

- Negative
- Positive
- Neutral

<table>
<thead>
<tr>
<th>Language</th>
<th>Negative</th>
<th>Positive</th>
<th>Neutral</th>
</tr>
</thead>
<tbody>
<tr>
<td>HINGLISH</td>
<td>3500</td>
<td>5250</td>
<td>7000</td>
</tr>
<tr>
<td>SPANGLISH</td>
<td>3500</td>
<td>5250</td>
<td>7000</td>
</tr>
</tbody>
</table>
DATA ANALYSIS

- **Skew in Data**
  - The dataset for Spanglish was skewed with much less data points for negative and positive tweets. To balance the data, we tried the following techniques
    - **Over-sampling**: Data for negative and positive tweets was oversampled in the inverse ratio of their proportion.
    - **Weighted cross entropy loss**: where weights were in inverse ratio of proportion.
    - **Under-sampling**: Data for neutral tweets was under sampled.
    - **New data points**: obtained from previous research corpora (*David Vilares et al.*).
Neutral data points are in general hard to classify and are most often neglected by researchers.

Since neutral tweets can be ambiguous and dependent on people's perceptions, many models performed poorly on neutral data points in comparison to positive and negative datapoints.

Some examples of datapoints annotated as neutral

- @avnishjaiswals1 @aimim_national @asadowaisi @BJP4India kutte to bhonkte rehate hain ...
- @apurba_official bhaiya u r looking khub sundoor 😊 love from kolkata bhaiya 😄
HEURISTICS

- Heuristics for classifying neutral data points.
- Train the model only on positive and negative data points.
- During predictions, use the following heuristics:
  - Feed forward and get the prediction scores for positive and negative classes.
  - If the model seems confused between the two (difference in scores ~ 0), then classify the tweet as neutral.
  - Otherwise, classify them as predicted by model.
FUTURE WORK

- Learning with Noisy Labels
  - Annotators often have many disagreements. This is especially so for crowd-workers who are not well trained. That is why one always feels that there are many errors in an annotated dataset.
  - Previous approach uses AB networks, consisting of two convolutional neural networks, one for learning sentiment scores to predict ‘clean’ labels and the other for learning a noise transition matrix to handle input noisy labels.

Ref: Learning with Noisy Labels for Sentence-level Sentiment Classification

Architecture of AB networks
FUTURE WORK

- Harnessing emoji information
  - We plan to use emojis present in the sentences as features to our model. We plan to use emoji2vec to encode emojis into vectors which can then be passed to the model along with word vectors.

Ref: [emoji2vec: Learning Emoji Representations from their Description](#)
This project is a part of Semeval 2020 SentiMix shared task hosted by Dr. Amitava Das.

We have a score of 0.659 against the top score of 0.705 for Hinglish Sentimix.

We have a score of 0.703 against the top score of 0.806 for Spanglish Sentimix.

De-Mixing Sentiment from Code-Mixed Text - [https://www.aclweb.org/anthology/P19-2052/](https://www.aclweb.org/anthology/P19-2052/)


QUESTIONS ?
THANK YOU !