Applying NLP to Twitter Airline Sentiment to Reveal Significant Text Features

Anudeep Golla
Stanford University
anudeepg@stanford.edu

1 Problem Description

Flying across the world has become more and more accessible within the past five to ten years. Consequently, understanding the benefits and drawbacks of various major airlines such as Southwest, Frontier, and United has become an increasingly important task before traveling. Furthermore, Twitter has emerged as a prime space for airline reviews to be shared among users. Analyzing a conglomeration of thousands of tweets thus allows us to understand general sentiment towards many major airlines. Ultimately, insights into the training and evaluation process using social media data could help reveal underlying significant effectors of natural language processing as a whole.

2 Challenges

When applying an NLP analysis approach to understanding airline sentiment, there are two main challenges that one may encounter:

(1) The first is **data preprocessing**. Extracting appropriate features from a text is critical for targeting certain language components central to the sentiment of the overall text. Even today, it is still a largely unsolved problem as to what the most important features in a piece of text are. In addition, the importance of these features may change depending on the specific type of result being sought. This project will tackle this challenge by comparing the results of employing different preprocessing techniques that emphasize different features of text.

(2) The other challenge is **improving and customizing pre-existing model architectures** that are commonly used for the specific problem of airline sentiment analysis. While many pretrained models exist for the purpose of sentiment analysis, it is important to optimize these models to the problem at hand. Specifically, determining certain augmentative architectures that combine well with the output of the already refined and efficient pretrained models can be a challenging task. In order to tackle this challenge, a pretrained model will be compared with the performance of the same model augmented by an appropriate network intended to enhance the results.

3 Methods: Dataset and Preprocessing

This project will utilize a dataset from Kaggle[1] that includes specific tweets, the airline being addressed, whether that tweet is positive, neutral, or negative toward the respective airline, and the sentiment confidence. Specifically, the dataset includes 14641 entries with each entry including the mentioned data.While many of these features provide valuable context and insight into the text itself, this project will largely focus on only the literal text and the label (positive, negative, neutral).

In order to introduce nuance to the input derived from this dataset and to tackle the first challenge mentioned above, this project will employ and compare two types of dataset preprocessing techniques to capture different features of each tweet during training. Essentially, while one technique aims to derive value from each word regardless of its place in the text, the other technique attempts to derive meaning by putting emphasis on the relationship between neighboring words.

The first technique utilizes **Bidirectional Encoder Representations from Transformers (BERT)**. Essentially, the pretrained BERT model architecture consists of a stack of Transformer encoders, where each encoder combines a self-attention layer with a feed-forward layer. BERT is unique, however, in that is learns from the bidirectional nature of each encoder. In other words, the model learns equally viewing the words from left to right as it does viewing the words right to left. This ensures that the final tokenized vector produced from preprocessing the raw text input is primed such that each word in the text is weighted equally to pass into the encoder stack. The BERT model is trained on data sourced from BooksCorpus and Wikipedia, a vast combined dataset containing over 3.3 billion words. A depiction of how input text is tokenized using the BERT tokenizer and how this tokenized sequence is passed through the BERT model architecture is shown below:

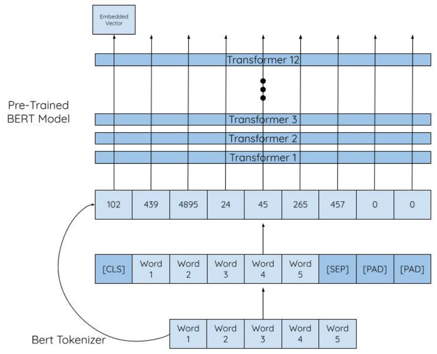

Figure 1: BERT Tokenizer

The second technique combines the use of a regular word tokenizer with a unique kernel method that groups neighboring words together. A depiction of how input text is tokenized using the regular word tokenizer and how this tokenized sequence is applied to kernels is shown below:

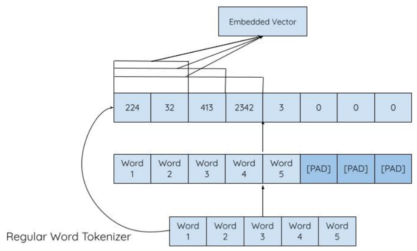

Figure 2: Regular Word Tokenizer

Through the word tokenizer, the input sequence is first padded until a desired sequence length is met. Then, the frequency that each word in the sentence occurs in the entire dataset is calculated. To promote simplicity, only words that pass a minimum frequency are considered. Once the words that will be considered are determined, they are provided with an index within a dictionary and that index is applied as their token value. Furthermore, once the input sequence is tokenized, 3 different kernel sizes of 2, 3, and 4 are applied to the tokenized vector in order to pass these grouped segments into a convolutional layer. Thus, with this data preprocessing technique, the relationship between words that are at most three units away within a sentence is emphasized during learning.

4 Methods: Architectures and Hyperparameters

In this project, three models will be trained and evaluated. The first is a baseline pretrained BERT model that will be applied to the BERT-tokenized dataset (Figure 1). The architecture of this model is shown below:

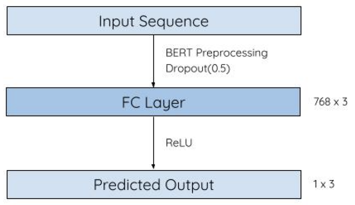

Figure 3: BERT Architecture

The BERT model and 1 dropout iteration is followed by a 768x3 fully connected linear layer. The output of this layer is activated through a ReLU to produce the final output.
The second model is the same BERT model augmented by a fully connected neural network, as shown:

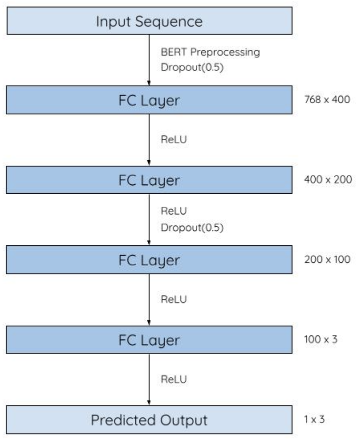

Figure 4: BERT+NN Architecture

In this augmented model, the BERT output is passed into a series of fully connected linear layers with dimensions 768x400, 400x200, 200x100, and 100x3 respectively. This fully connected series is meant to further learn on the embedded vector produced by the BERT model. This model will also be applied to the BERT-tokenized dataset.

Finally, the third model will be a three-way CNN that utilizes the separate kernels described in the above methods. A detailed diagram is shown below:

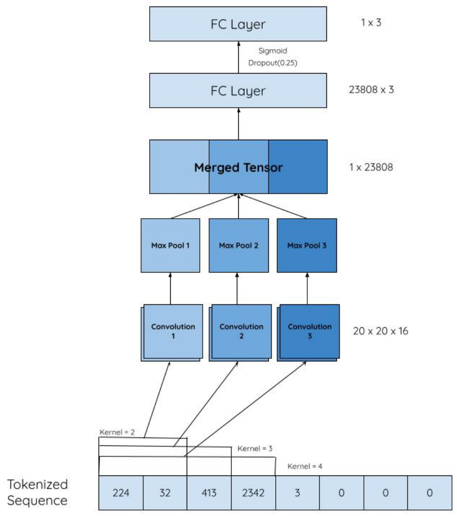

Figure 5: CNN Architecture

As shown, the output of each of the three kernel-conv1d-maxpool series will be merged into one tensor, which will then be passed into a 23808x3 fully connected liner layer to derive the output.

The training process will utilize a 90-5-5 split for training, validation, and test sets. All three models will be trained using Adam optimization and cross entropy loss as we are working with multi classification. While the pretrained model was trained for 10 epochs, each of the other models were trained for 66 epochs with a learning rate of 1e-6. Furthermore, the pretrained model was trained with a mini-batch size of 5 while the other two models were trained without batches.

5 Results

Provided below is each model's resulting accuracies for the training, validation, and test sets as well as the precision and recall for the test set. Precision and recall values were calculated by averaging the precision and recall of each of the three classes within the predictions respectively.

	Train Accuracy	Validation Accuracy	Test Accuracy	Test Precision	Test Recall
BERT	0.927	0.824	0.827	0.787	0.777
BERT + NN	0.901	0.811	0.813	0.768	0.753
CNN	0.769	0.715	0.723	0.684	0.548

Table 1: Accuracy, Precision, and Recall Values

While the BERT model achieve the highest accuracies in all data sets, it is worth noting that the difference between both validation and test accuracies and training accuracy for the BERT model is significantly higher. In addition, it is shown that the augmentation of a fully connected neural network did not improve the performance of the individual BERT model. Finally, while performance of the CNN in terms of accuracy was much lower, the difference between both validation and test accuracies and training accuracy for the CNN model was minimal. This association is depicted in the below loss curve for the CNN:

It is shown how the training and validation loss decrease together while training is stopped before a significant difference between the losses emerges. Finally, while both the BERT and augmented BERT models showed small changes in their test precision and test recall values, the CNN model exhibited a much larger increase. This difference for the CNN largely favored precision over recall.

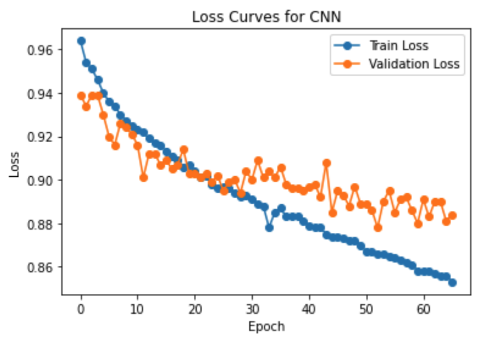

Figure 6: CNN Loss Curve

Further provided are the confusion matrices for each model when applied to the test set. It is clear that all the models performed best with classifying texts as 0 or ’negative’.

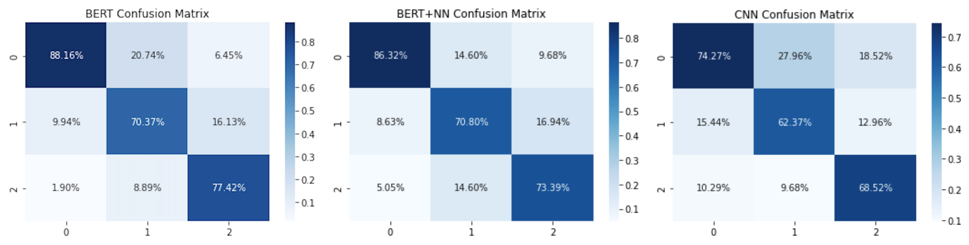

Figure 7: Confusion Matrices

6 Discussion

Ultimately, this project addressed both challenges discussed at the beginning of the report. In terms of the dataset preprocessing techniques, it was seen that the models trained on the BERT-tokenized data performed better than the models trained on the regular-word-tokenized data. However, these results inspire interesting discussion; the fact that the difference between both validation and test accuracies and training accuracy for the BERT and augmented BERT models is significantly higher suggests that there is some level of overfitting involved. Furthermore, since the CNN does not exhibit this separation between training and validation/test accuracy, it may be possible that the BERT-tokenizer strategy leads to higher likelihood of overfitting. It was also shown that all the models performed best with classifying texts as 0 or ’negative’. It is possible that this result is due to a dataset imbalance, as there are more than double the number of ’negative’ data examples as there are for ’positive’ and ’neutral’. This imbalance could have also led to poorer performance overall on ’positive’ and ’neutral’ examples due to lack of exposure to these entries. In addition, the project also tackled the challenge of improving pre-existing models used for sentiment analysis. The results displayed that augmenting a fully connected neural network did not improve the performance of the baseline BERT model. There are two potential explanations for this. First, it may be possible that the embedded vector outputted by the BERT model has already maximized feature extraction from the input text and further learning on the outputted vector in any capacity is unnecessary. On the other hand, it is possible that the architecture and hyperparameters specifically chosen for the augmented neural network were sub optimal in further learning on the BERT output. Further exploration into this area could reveal other possibilities of improving performance of pretrained NLP models.

7 Contributions

This project was done entirely individually with guidance on model design decisions inspired from the references listed below.

References

[1] Eight, Figure. “Twitter Us Airline Sentiment.” Kaggle, 16 Oct. 2019, https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment?resource=download.

[2] Gao, Zhengjie, et al. "Target-dependent sentiment classification with BERT." Ieee Access 7 (2019): 154290-154299.

[3] Angiani, Giulio, et al. "A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter." KDWeb. 2016.

[4] Winastwan, Ruben. Text Classification with Bert in Pytorch - towards Data Science. https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f.

[5] López, Fernando. Text Classification with CNNs in Pytorch - towards Data Science. https://towardsdatascience.com/text-classification-with-cnns-in-pytorch-1113df31e79f.