Airline Tweet Sentiment Analysis: Use of Transformer Architecture and Captum Interpretability in Text Classification

Ange Olson, Pavani Samala, Yaxin (Janet) Zhuang

I. Introduction

In this project, we fine-tune a BERT model by adding an MLP head to perform text classification on Twitter Airline Sentiment data. We analyze our model conventionally, looking at metrics like accuracy and the F1 score, and then use Integrated Gradients to discern why the model makes decisions as it does. The aim is to be able to interpret our model and provide an idea of the confidence we have in the model's ability to perform well on data it hasn't seen.

This paper is organized as follows: First, we describe our dataset and show the results of our Exploratory Data Analysis (EDA). Next, we provide a description of the model that we used. Then, we discuss methodology in terms of data pre-processing and model development, and lastly we discuss interpretation and results. Finally, we summarize our findings and discuss areas of growth. Our main contribution is to see how we can interpret BERT in the context of sentiment analysis, where current work focuses on interpreting BERT in encoder-decoder settings or interpreting classification models that do not utilize transformer architecture.

In sum, our basic model performed well; we achieved a total accuracy of 0.79 on our test set, with training and validation accuracy at 0.77 and 0.78 respectively, indicating that our model was not overfitting. Based on our post-hoc analysis, the most challenging class to predict were the neutral sentiment tweets, while the easiest class to predict was the negative tweets. From interpreting our model, we find some evidence that there may be overfitting and/or weight given to words or subwords that in context do not make sense. Overall, however, we find that from a sub-sample of our test data, we have reason to believe that our model is correctly identifying and giving weight to certain tokens over others.

II. Description of Dataset

Our data is originally from Crodflower's Data for Everyone library. The data is about how travelers in February 2015 expressed their feelings on Twitter, and it was slightly reformatted by Kaggle author Figure Eight. The data has 14640 records and 15

columns, but we will use *text* and *airline_sentiment* as our data points, choosing to focus on text analysis only. The figure to the left shows the count of tweets per class and reveals that the dataset was imbalanced, containing more negative tweets. The figure to the right shows the distribution of polarity without mapping it to classes. It is seen that there are far less very negative and very positive tweets and the majority of the tweets are neutral.

Figure [1]: Count of Tweets per Sentiment

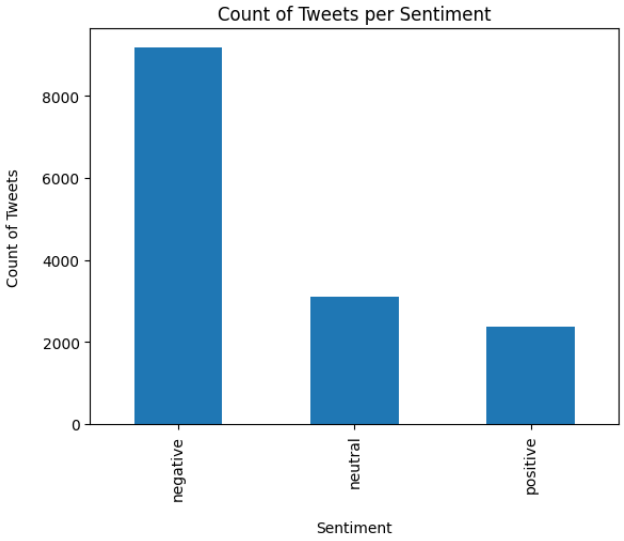
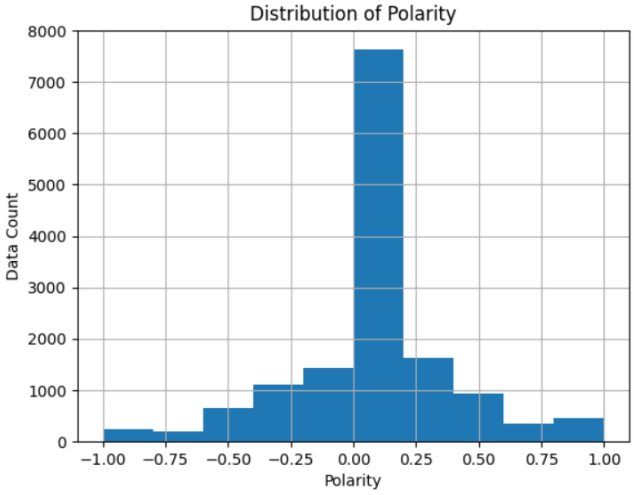
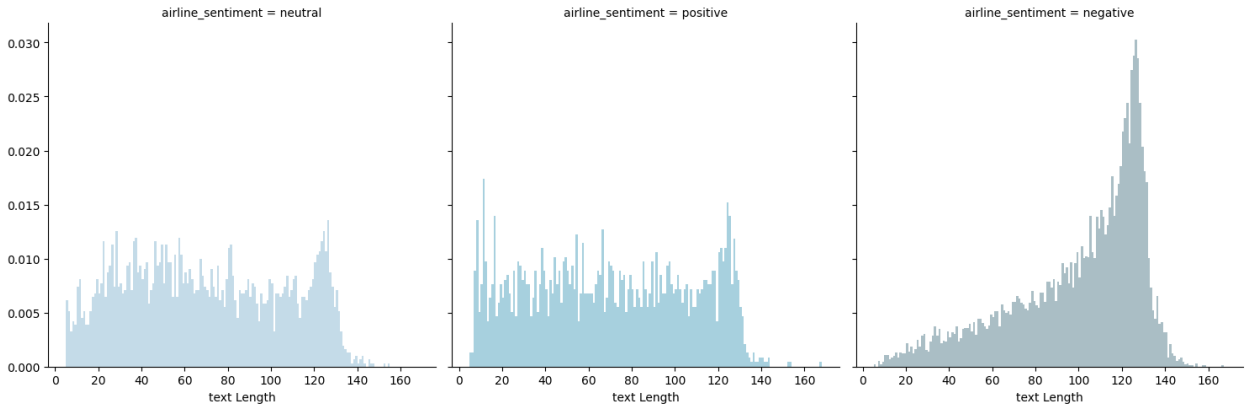


Figure [2]: Count of Tweets per Sentiment



Another variable we compared to the tweet's sentiment was tweet length. The figure below shows that the negative connotation tweets had more variation in tweet length, while the positive and negative tweets had approximately the same number of tweets for each tweet length.

Figure [3]: Length of Tweets per Sentiment



Dataset Description

Table[1]: Description of Dataset

Variable	Type	Description	Values
text	String	Twitter contents	Strings
airline_sentiment	Categorical	Travelers' feelings for six US airlines in February 2015	Positive, neutral, negative

III. Description of Model

A Multilayer Perceptron Network (MLP) is an expansion of the Perception model and is structured with an input layer, multiple hidden layers, and an output layer. The input layer receives the data and sends it to the hidden layers, which perform nonlinear transformations that are then classified by the output layer.

As the field of deep learning evolved, a neural network architecture known as Transformer was developed to solve sequence-to-sequence problems and has reached state-of-the-art performance. A transformer operates with a mechanism called self-attention and may be used in conjecture with different base models.

The transformer implemented in this project is a Bidirectional Encoder Representations from Transformers, also known as BERT, from a Google paper titled "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". It follows the typical encoder-decoder transformer architecture, but in the case of building a language model, only the encoder component is used.

Directional models read a text from left to right, using only one direct of context to provide information about a word or subword. Bi-directional models like BERT read a text from both directions, useful in a language like English where the same word can have very different meanings or convey different themes depending on the surrounding words. An important part of the BERT model is self-attention. This mechanism is used to denote which parts of a sequence of text should be used to build up that context.

We opted to use BERT to take advantage of the benefits of transfer learning and self-attention, which we believe will lead to better results than a smaller model trained from scratch. In this project, we use BERT base, a smaller version of the model with 12 layers, 12 attention heads, and 110 million parameters. We implement a pre-trained

version of BERT and work on fine-tuning only for this project; when trained from scratch, Masked Language Modelling (MLM) and next-sentence prediction are used.

Since our project is on classification, the head we need to use on top of the pre-trained BERT model is at its simplest, an amended fully-connected layer taking the inputs as the inputs of the previous BERT fully-connected layer and the outputs in the shape of the classes we are looking to define. Several of these fully-connected layers can be added (our MLP model) or Convolution (CNN) or Recurrent (RNN) Neural Networks can be used instead to add complexity. As our aim is primarily on interpretability, we focus on what can be done given the simplest model, i.e. BERT with an MLP classification head.

IV. Methodology

Data Preprocessing:

As mentioned, the dataset used consists of tweets addressed to airline companies. The data was preprocessed by removing special characters and basic punctuation, like commas or periods. We left in exclamation points and question marks as these could be reasonably linked to different sentiments. We also split contractions into their sub-components. One of the limits of this approach is that we remove some special characters that likely are used to convey emotion–emojis. Additionally, tweets use informal language that is constantly changing–rule-based approaches for understanding some slang could have been used, but it would be unlikely for us to catch all cases. One of the reasons we chose to use a BERT base model is to take advantage of sub-word tokenization and the pre-trained embeddings–even if an entire word isn't recognized, at least some of it might be.

We also one-hot encode our labels, given that we are dealing with a multi-class problem in which we would like to assign some kind of probability of a tweet belonging to any one class.

Fine-Tuning:

Given that we are fine-tuning a BERT model, we are limited in hyperparameters we can choose; however, we look at freezing and unfreezing BERT layers, experimenting with learning rate (note: keeping a low learning rate if BERT layers are unfrozen), batch size, and optimizer.

Performance

Our model loss function is BCELoss, or Binary Cross-Entropy loss, used for classification problems. As specified, each of the individual targets falls between zero and one. We use accuracy during training in conjunction with loss to track model performance over each epoch. For post-hoc analysis on our test sample, we also look at loss and accuracy but also look to measure recall, precision, and the F1 score for each class to see where our model specifically is performing well or not.

Interpretation

One way we seek to judge the performance of our model is through interpretability. With text classification, one way to do this is to look at the effect of individual words on the predicted class of a document. In sentiment analysis, we would expect words like “bad” to lead to negative classifications, and words like “good” to lead to positive classifications. If the tokens in a document that impact its classification seem random or do not make sense within the context of a language, that might suggest that the model is overfitting or would otherwise perform poorly in production.

Integrated gradients are one way of interpreting the model. Where x is an input, and x' is a baseline, the formula for calculating integrated gradients is as follows:

$$IntegratedGrads_i(x) ::= (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i}$$

In the context of this project, an input is a numerical representation of a tweet using BERT encodings, and the baseline is a corresponding vector of null values–the [CLS] token, a vector of zeros corresponding to the tweet size, and the [SEP] token. The idea is that to understand the effect of individual words on an output, there must be a comparison made between ‘words’ that should have absolutely no effect–i.e. null words. In software and practical applications, the integrated gradient is approximated, using Reimann summation or the Gauss-Legendre quadrature rule.

We use the Captum package *LayerIntegratedGradient* class, which is a variation of integrated gradients. The algorithm assigns an attribution score to layer inputs, based on the approximation of the gradient calculation of the outputs with respect to the inputs from the baselines. In our case, the layer is the BERT embeddings, which are supposed to provide context and meaning to each token. The sum of the attribution scores of any individual input provides an idea of the overall contributions of the tokens in a document to its classification. For classification problems, a target is given corresponding to the class.

V. Results

MLP Head

Figure [4]: Training and Validation Accuracy

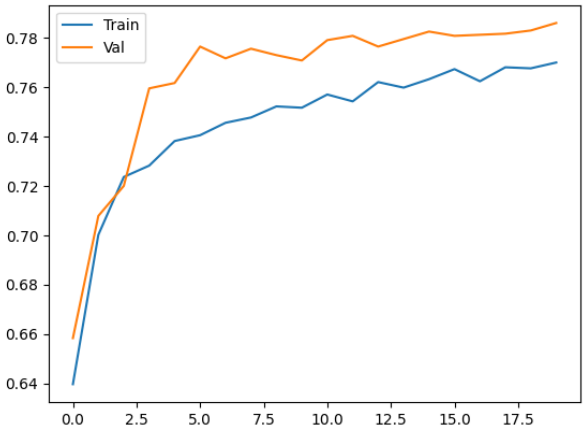
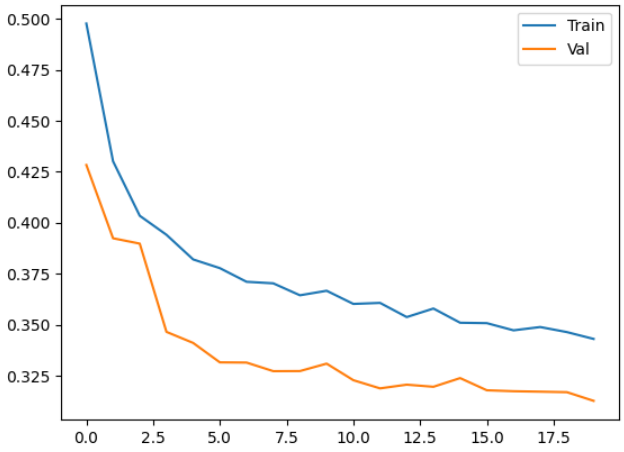


Figure [5]: Training and Validation Loss



RNN Head

Figure [6]: Training and Validation Accuracy

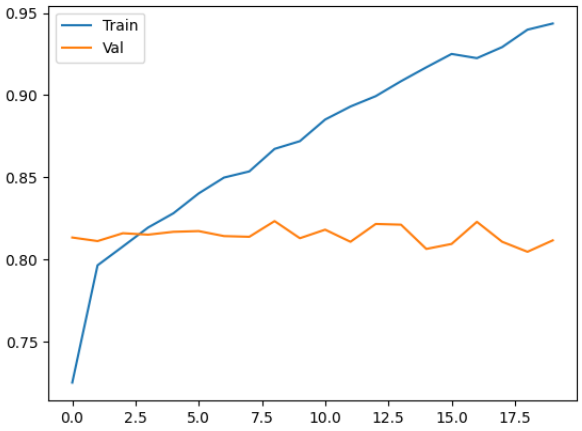
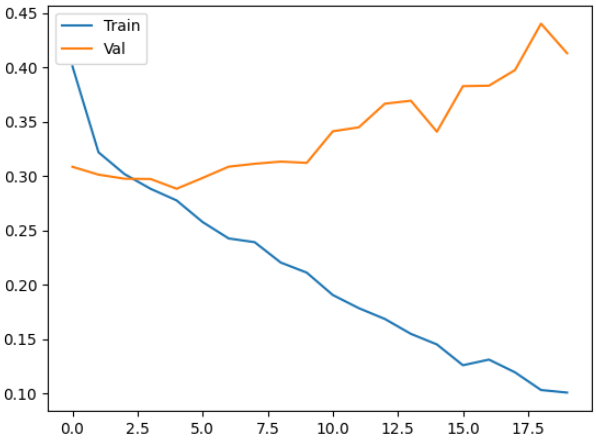


Figure [7]: Training and Validation Loss



The plots above show the training and validation accuracy and training and validation loss for running the Bert model using MLP head and RNN head on 20 epochs. It is clear that the model was overfitting when it was run using the RNN head, making MLP a better choice. The numerical values are summarized in the table below.

Table[2]: MLP vs RNN results

	MLP Head	RNN Head
train_loss	*0.3430536785354353*	*0.10062554912412003*
train_accuracy	*0.7700128424657534*	*0.9436001712328768*
val_loss	*0.31268693341149223*	*0.4129580193095737*
val_accuracy	*0.7860243055555556*	*0.8116319444444444*

Test Data: F1, Precision, Recall by class

We evaluated the model using precision, recall, and F1-score and summarized the result in Table 2. 84% of tweets classified as negative were truly negative and 90% of the actual negative connotated tweets were correctly classified. 71% of tweets classified as positive were truly positive and 73% of the actual positive connotated tweets were correctly classified. 65% of tweets classified as neutral were truly neutral and 49% of the actual neutral connotated tweets were correctly classified. The F1-scores follow the same trend, where it is highest for negative tweets followed by positive and neutral tweets. Overall, the model performed better while classifying negative tweets.

Table 2: Classification Metrics

Table[3]: Classification Metric Results

Classes/Metrics	Precision	Recall	F1-score
Negative	0.84	0.90	0.87
Positive	0.71	0.73	0.72
Neutral	0.65	0.49	0.56

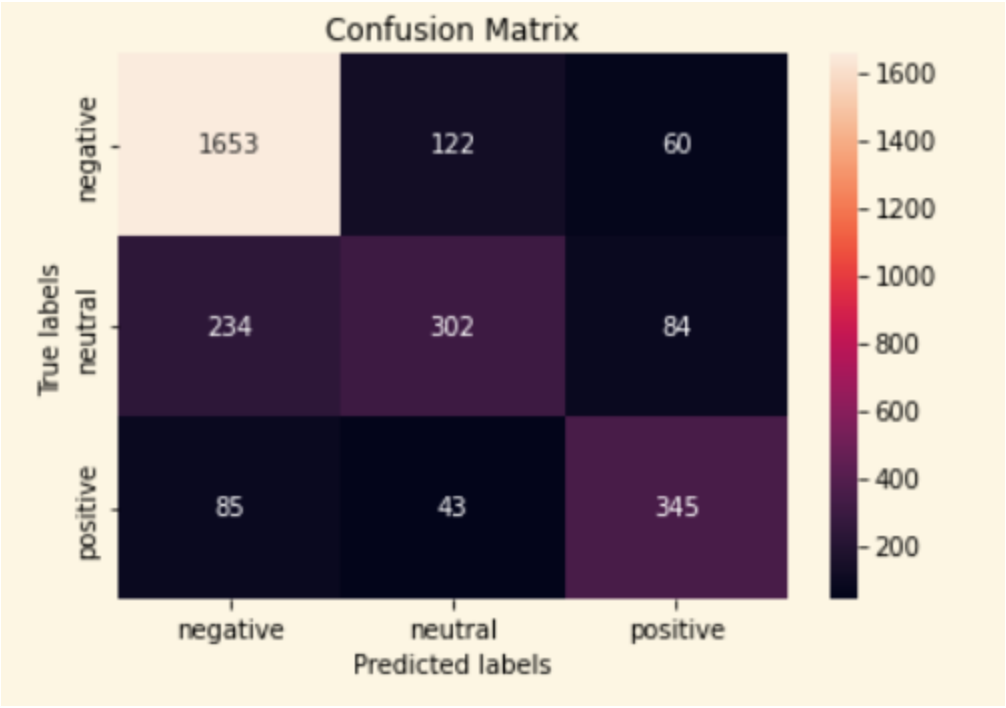


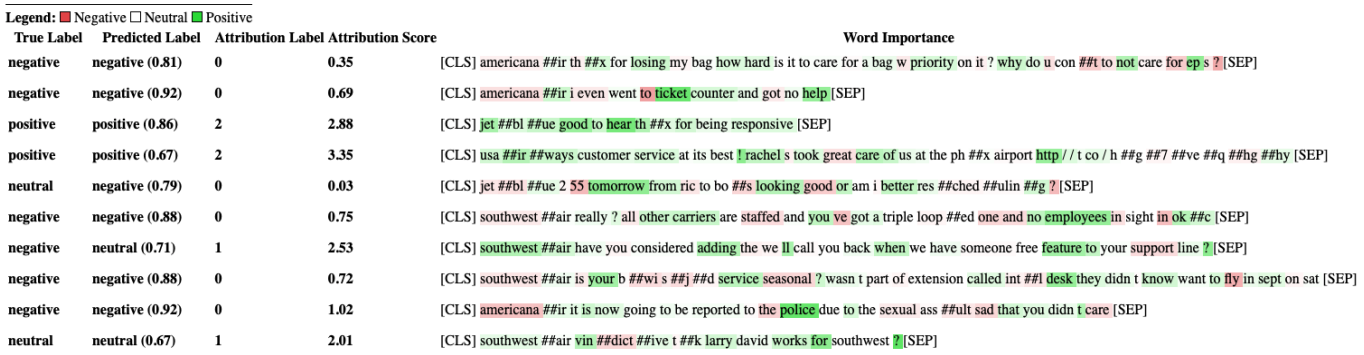
Figure 8: Confusion Matrix

Interpret Confusion Matrix: In this figure, there are 1835 actual labels negative, and the model predicted 1972 negative labels. The model correctly predicts 1653 negative labels from 1972 predicted negative labels and correctly classifies 1653 negative labels from 1835 actual labels. For neutral classification, there are 620 actual labels, and the model makes 489 neutral predictions. The model correctly classifies 302 labels from 620 actual labels and 489 predicted labels. For positive classification, the model correctly classifies 345 labels from 473 actual labels and 467 predicted labels.

Interpretation Results

Below are examples of attributions assigned to tokens in a series of tweets. Note that positive, negative, and neutral do not correspond to sentiment but to effect on the output. If a tweet is classified as positive, for example, then a token with a strong negative attribution (in dark red) had a negative impact with the output–we would expect a word like “bad” to fall into this category.

Figure 9: Captum LayerIntegratedGradient Results



Legend: Negative Neutral Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
negative	negative (0.81)	0	0.35	[CLS] americana ##ir th ##x for losing my bag how hard is it to care for a bag w priority on it ? why do u con ##t to not care for ep s ? [SEP]
negative	negative (0.92)	0	0.69	[CLS] americana ##ir i even went to ticket counter and got no help [SEP]
positive	positive (0.86)	2	2.88	[CLS] jet ##bl ##ue good to hear th ##x for being responsive [SEP]
positive	positive (0.67)	2	3.35	[CLS] usa ##ir ##ways customer service at its best ! rachel s took great care of us at the ph ##x airport http / / t co / h ##g ##7 ##ve ##q ##hg ##hy [SEP]
neutral	negative (0.79)	0	0.03	[CLS] jet ##bl ##ue 2 55 tomorrow from ric to bo ##s looking good or am i better res ##ched ##ulin ##g ? [SEP]
negative	negative (0.88)	0	0.75	[CLS] southwest ##air really ? all other carriers are staffed and you ve got a triple loop ##ed one and no employees in sight in ok ##c [SEP]
negative	neutral (0.71)	1	2.53	[CLS] southwest ##air have you considered adding the we ll call you back when we have someone free feature to your support line ? [SEP]
negative	negative (0.88)	0	0.72	[CLS] southwest ##air is your b ##wi s ##j ##d service seasonal ? wasn t part of extension called int ##l desk they didn t know want to fly in sept on sat [SEP]
negative	negative (0.92)	0	1.02	[CLS] americana ##ir it is now going to be reported to the police due to the sexual ass ##ult sad that you didn t care [SEP]
neutral	neutral (0.67)	1	2.01	[CLS] southwest ##air vin ##dict ##ive t ##k larry david works for southwest ? [SEP]

We use ten samples in this analysis, eight of which are predicted correctly. The first column in Figure 9 displays the true label of the Tweet in the leftmost column. The second column displays the prediction, with a confidence score (the highest score from the softmax function output of the model) between zero and one acting as the probability that the predicted class is the true class. The attribution label is the location in the model one-hot encoding of the highest prediction, used in calculating the integrated gradient, and the attribution score is the sum of all the individual attribution scores of each word or subword in the Tweet.

Some words which appear to have negative connotations, based on either their positive impact on the negative predictions or their negative impact on the positive predictions are “police,” “americana” (American Airlines), “service,” “desk,” “no,” “you,” “tomorrow,” and “help.” Some of these words reasonably do have negative connotations–American Airlines may not be a popular airline, and words related to “help desk” or “service desk” can be reasonably linked to an individual who has a problem and may have negative sentiments. Conversely, the words “good,” “care,” and “responsive” are positively associated with positive sentiment Tweets, which does make sense.

However, we can also see areas where the model may be placing weight on words that may not be indicative of any sentiment (but are not contributing to the third category, neutral). For example, “http,” a url component, has a high attribution score towards the positive sentiment, which explainability-wise makes no sense, and suggests the model may be picking up on noise in the data or overfitting based on specific patterns that might not be present in different test sets.

Lastly, we look to see which words from this sample are associated with neutrality. Of the ten samples above, two were actually neutral (with one incorrectly labeled as negative) and one negative tweet was incorrectly labeled as neutral. For the one true positive, “Southwest,” “Larry,” “vin,” “works,” “for,” and “?” all positively contribute to the

prediction. Stories can be created to explain why the model looks at some of these tokens as neutral contributors–one interesting piece of information is that Southwest seems to be a contributor to neutrality, suggesting that it is not viewed negatively. “For,” a stopword, also makes sense in this case and the name “Larry” also is understandable as not being indicative of a specific sentiment. The negative tweet predicted as neutral also contains a question mark and “Southwest.”

VI. Summary and Conclusion

This project focused on combining a BERT model with an MLP head to perform text classification on Twitter Airline Sentiment data. The model was fine tuned by changing batch size, epochs, and learning rate. After calculating the classification metrics, we were able to see the model had an overall accuracy of 79% and performed better while classifying negative tweets. By implementing an Integrated Gradient technique, we were able to see which words impacted the model’s classification of positive or negative, and saw that the model was grasping the context of some words. However, the attributions were not always correct and leaves room for improvement. Recent works, led by Shanshan Yu, Jindian Su, and Da Luo at the South China University of Technology, have found that training models with auxiliary sentences have improved accuracy scores. They have also found that introducing a domain-specific dataset has improved the model’s performance.

VII. References

https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment?datasetId=17&sortBy=voteCount
https://medium.com/aiguys/bert-explainability-5b54cff01407
https://captum.ai/tutorials/Bert_SQUAD_Interpret
https://github.com/bentrevett/pytorch-sentiment-analysis/issues/87
https://www.geeksforgeeks.org/fine-tuning-bert-model-for-sentiment-analysis/
https://stackoverflow.com/questions/73911673/pytorch-dataloader-with-huggingface-transformer-getting-error-unable-to-creat
https://github.com/pytorch/captum/blob/master/captum/attr/_utils/visualization.py
https://www.theaidream.com/post/google-bert-understanding-the-architecture
https://arxiv.org/abs/1810.04805
https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html
https://www.tensorflow.org/tutorials/interpretability/integrated_gradients
https://arxiv.org/pdf/1703.01365.pdf
https://github.com/pytorch/captum/blob/master/captum/attr/_core/layer/layer_integrated_gradients.py
https://www.researchgate.net/publication/337510909_Improving_BERT-Based_Text_Classification_With_Auxiliary_Sentence_and_Domain_Knowledge