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

Ange Olson Individual Report

I. Introduction:

In this project, my group aimed to classify tweets at major US airlines as negative, position, or neutral sentiment and interpret our model using more advanced methods, including integrated gradient. We used PyTorch as our framework to develop MLP and RNN model heads on top of a BERT transformer base to take advantage of the benefits of transfer learning. We utilized the pre-trained weights of the model, focusing on fine-tuning to achieve a strong enough model performance to be able to work on interpretation.

We chose to focus on interpretation for our model with the MLP head. While in the same number of epochs (20) the RNN-head model achieved higher training and validation accuracy, the model showed signs of overfitting to the training data (see Figure A). Given the trajectory of validation accuracy and loss from our MLP-head model (see Figure A), this model appears to be better-suited for applications to data outside of the training sample and may improve given more training time. Given that in interpretation we would like to see some instances of the model “getting it wrong” to make inferences on what the model is looking at when it underperforms, we cap training at 20 epochs.

My primary work on the project was as follows:

- Developing the pre-processing pipeline
- Constructing the model classes and training/testing loops
- Using Captum *LayerIntegratedGradient* to look at token attribution scores for several predictions to get an understanding of whether or not our model was assigning classes to tweets in a way that made sense given the context of the token.

In addition to the code above, the sections of the paper and presentation I primarily worked on were related to our preprocessing, model architecture, and interpretation. My group members focused on modifying the model hyperparameters, adding to preprocessing (focusing on how tweets differ from regular text–use of contractions, slang), and looking into using other interpretation methods such as LIME to see if we could apply those to our project.

II. Individual Work:

I developed a pipeline to process both the labels and the text of the tweets. For labels, I opted to concatenate labels into a vector of one-hot encodings. This way, the output of the model (softmax applied to a similar vector of three elements) could be interpreted as the probability of the classes in the encoded labels, which we needed for interpretation.

For the text, I implemented a function using regex to remove certain non-alphanumeric characters, punctuation, and to add the word ‘not’ given ‘n’t’ as that would have an impact on sentiment analysis. The tokenizer used was the *AutoTokenizer* from the "bert-base-uncased" model. A collator function with padding based on BERT tokenization was used, with max_length equal to 300 and special characters, an attention mask, and padding/truncation true. I created a custom dataloader to be able to run mini-batches to train more quickly and developed training, validation, and testing loops for both MLP and RNN based models.

Following a Captum tutorial on interpreting BERT models, I altered the code to work for classification rather than question answering. My primary contributions were turning the code into a function that could return attribution scores visualized as an HTML object (Figure C) that would be opened in a web browser, adjusting the helper functions, and processing the labels to be able to use as target features, since for classification that field is required.

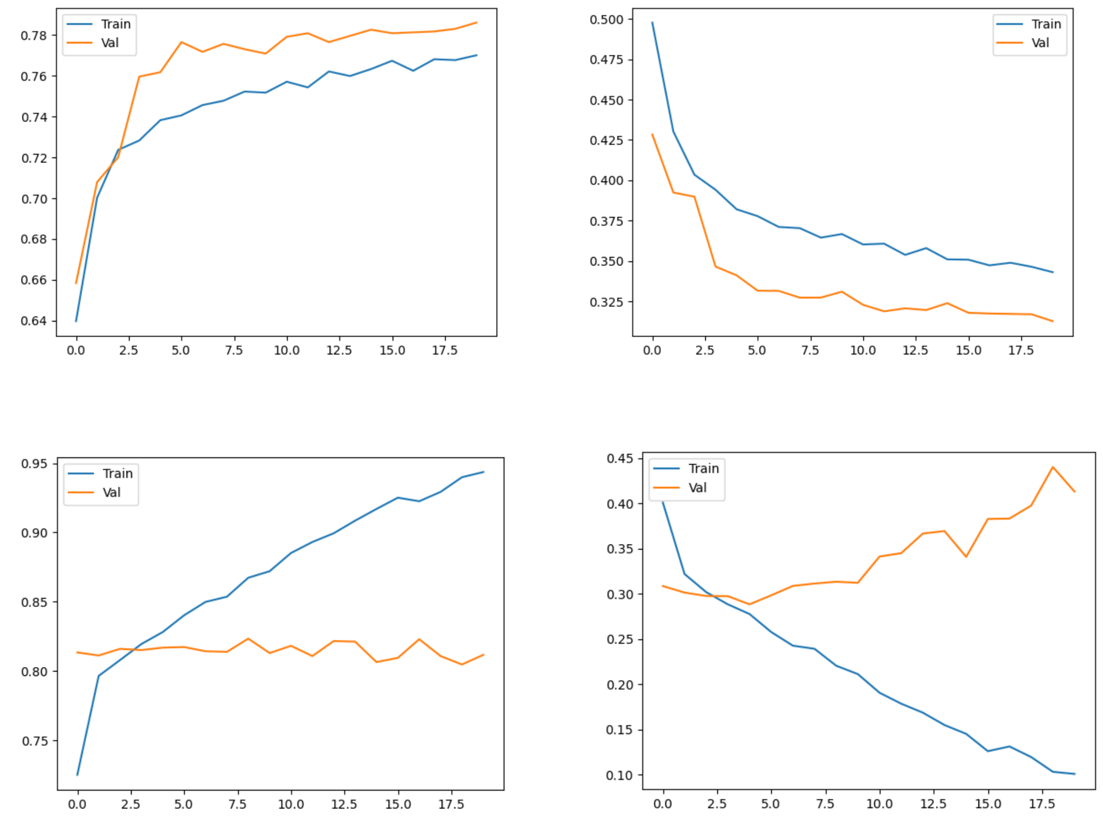
III. Results:

As discussed, our MLP-head model appears better for broader applications and shows less evidence of overfitting. Table 1 provides the end results after running the training loop for 20 epochs on both variations of our classification models.

Table 1: Model Performance After 20 Epochs

Model	**Train Loss**	**Train Acc**	**Val Loss**	**Val Acc**
RNN-Head	0.10062554912412003	0.9436001712328768	0.4129580193095737	0.8116319444444444
MLP-Head	0.3430536785354353	0.7700128424657534	0.31268693341149223	0.7860243055555556

Figure A: Model Performance Over 20 Epochs



From upper left, going clockwise: MLP accuracy, MLP loss, RNN accuracy, RNN loss. The RNN-head model overfits; validation accuracy does not improve over time, and loss actually increases.

Test loss and accuracy were comparable to validation loss and accuracy for the MLP-head model; we did not test our RNN-head model on our test set.

Table 2 describes the results of post-hoc analysis on our test set performance. Here, we see that the model performs best when classifying negative Tweets; neutral-sentiment tweets are most difficult. Table 3 displays the confusion matrix, showing how often the neutral-sentiment Tweets were mistaken, particularly for negative Tweets.

Table 2: Classification Metrics

Classes/Metrics	Precision	Recall	F1-score
Negative (A)	0.84	0.90	0.87
Positive (C)	0.71	0.73	0.72

Neutral (B)	0.65	0.49	0.56

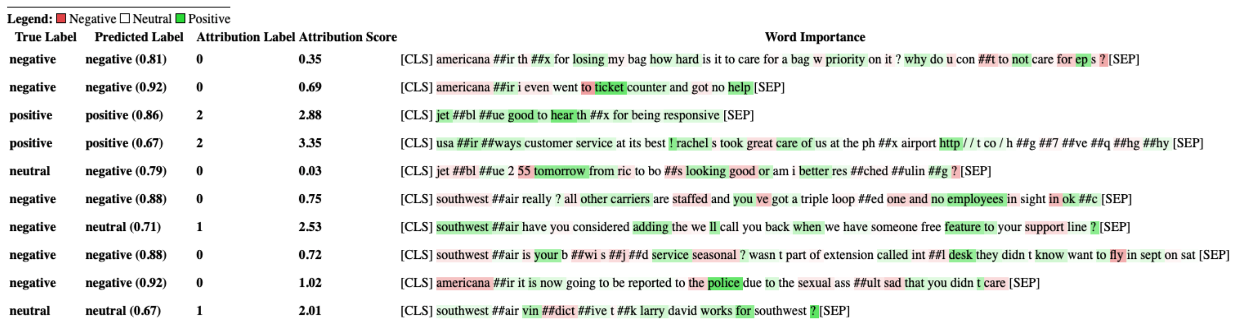
Table 3: Confusion Matrix

	Neg	**Neutral**	**Pos**
Neg	1653	122	60
Neutral	234	302	84
Pos	85	43	345

Neutral-sentiment tweets were almost as often recognized as negative tweets.

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 B: Interpretation Visual



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 [X] 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.”

IV. Summary and Conclusions:

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 left 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.

I learned a great deal in this project, particularly around integrated gradients and how to use the mathematics behind the model optimizer to find out feature importance in the context of natural language processing. Beyond learning the theory, my programming skills and ability to understand the functions in the packages I am using has increased tremendously–I felt well-prepared by this class to read through Captum source files to find out what each function was doing, what I needed to input (given that I was not exactly following the guide) and why. Additionally, I learned a great deal about the difference that pre-processing can make given the dataset. Primarily in class we have worked with more formal text, and the preprocessing done there is much different than for informal text found in Tweets. I noticed significant differences in model performance by looking at the predictions and attributions. The first go around, with basic pre-processing, each tweet still contained words or subwords that didn't make sense. While the resulting pre-processing was not perfect, reading through the data and thinking through what kinds of changes you need to make given the specific text helps.

Given more time, I think that improvements to our project could be two-fold: one, in the modeling itself, and two, in the interpretation. Modelling-wise, I think there was a good deal more we could have done in pre-processing. While it was easier to remove special characters and punctuation, given Emojis, there was certain punctuations that we likely wanted to keep in–a smiley face should indicate positive sentiment, for example. If the BERT tokenizer did not take this into account, we could have looked into augmenting the BERT tokenizer. While somewhat outside the scope of this course, in addition to the tweet data, information was originally collected on timestamp and other tabular data, which we removed for the sake of focusing on text but which could reasonably have an impact on sentiment or would be interesting to analyze–airlines could reasonably be interested in knowing which times of year or times of day are indicative of positive or negative sentiments (assuming tweets are representative of real-time emotions). Lastly, I would have been interested in seeing how other, simpler methods of interpretability may have worked. Would TF-IDF give us more information about token importance? Would looking at the attention mask values for certain tokens? Is there a way to look specifically at sensitivity to gain an understanding of token importance? There are many methods out there that we did not have the time to learn about or implement that would have been interesting to try.

Code Calculation:
Novel Code: 596 lines (NLP_CodeNovel.py) Note: includes in-class code
Edited: 104 lines (NLP_CodeEdited.py)

Code from Internet: 185 (NLP_InternetCode.py)

185-104/(185+596) * 100 = 81/781 * 100 = ~10.37

Resources:

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