Sentiment Analysis: Measuring the Predictive Ability of Models for U.S. Domestic Airline Tweets

Niraj Bangari, Jared Choy, Sandeep Nair, Gianni Spiga

August 3, 2025

Contents

1 Introduction

As the number of travelers exceeds pre-pandemic levels and airlines struggling to break even in profit [Asq], it is more important now than ever for airlines to understand what aspects of the customer experience need improvement. A great way to get such feedback is to measure the sentiment of tweets of travelers when mentioning their recently flown airline.

Our data comes from *Crowdflower's Data for Everyone Library*, a database with publicly available datasets. The data is described by the site: "A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as 'late flight' or 'rude service')" [kag19]. This data was made available directly to us by way of Kaggle.

Our original motivation for this analysis was to perform sentiment analysis for tweets involving the Israel-Palestine conflict. However, due to limitations in data scraping as well as the recently implemented paywall for the use of API by X, we decided this would not be feasible and continued our analysis with US airline tweets.

2 Problem Definition/Data Description/ Literature Review

Our dataset covers tweets for February 2015 regarding six domestic airlines: American Airlines, Delta, Southwest, United, U.S. Airways, and Virgin America. Table 1 below shows the following variables in our data:

Column	Description
tweet id	Unique ID for Tweet
airline sentiment	Labeled Sentiment of the Tweet
airline sentiment confidence	Confidence Measurement of Assigned Label
negative reason	Reason for Tweet (if negative sentiment)
negative reason confidence	Confidence Measurement of Negative Reason
airline	Which Airline Tweet was Referencing
name	Name of User
retweet count	Number of Retweets
text	Raw String of Entire Tweet
tweet created	Timestamp of Tweet Creation
user timezone	Timezone of User at Tweet Creation.

Table 1: Data Description Table.
There are four variables not included above: "airline sentiment gold", "negative reason gold", "tweet coord", and "tweet location." These variables contained all null values and were thus excluded from any analysis.

In our exploratory data analysis, we will discuss many of the variables in the table. However, for our main analysis, we will primarily focus on two variables, the airline sentiment and the raw text of the tweet. We will next talk about the steps for how we pre-processed our text variable so that we could set the stage for our predictive analysis.

2.1 Pre-Processing

Text pre-processing is a crucial step in natural language processing (NLP) tasks, as it helps clean and transform raw text data into a format that is suitable for analysis. Common pre-processing techniques include:

1. Lowercasing: Convert all text to lowercase to ensure uniformity and prevent duplication of words with different cases.

2. Tokenization: Break the text into words or n-grams of words. This step is essential for further analysis as it enables the model to understand the structure of the text.

3. Removing Special Characters: Eliminate non-alphanumeric characters, punctuation, and other special symbols that may not contribute much to the meaning of the text.

4. Lemmatization: Reduce words to their base or root form to handle variations in tense, number, or other grammatical forms. Stemming usually involves removing prefixes or suffixes.

After pre-processing the words, we need to numerically represent our textual data as a design matrix. TF-IDF is a numerical statistic used to reflect the importance of a term in a document relative to a collection of documents. It is commonly used for information retrieval and text mining.

The TF-IDF score for a term t in a document d is calculated as follows:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Where:

- $\text{TF}(t, d)$ is the term frequency of term t in document d, i.e., the number of times t appears in d.
- $\text{IDF}(t)$ is the inverse document frequency of term t across the entire document collection. It is calculated as:

$$\text{IDF}(t) = \log\left(\frac{N}{\text{DF}(t)}\right)$$

  Where N is the total number of documents in the collection, and $\text{DF}(t)$ is the number of documents containing the term t.

The TF-IDF score increases with the number of times a term appears in a document (TF) but is offset by the frequency of the term across all documents (IDF). Terms that are common across all documents receive lower weights, while terms that are specific to a document receive higher weights.

This matrix can be represented as a sparse matrix, where rows correspond to documents, columns correspond to terms, and each entry represents the TF-IDF score of a term in a document. The resulting TF-IDF matrix is often used as input features for machine learning models in NLP tasks.

2.2 Literature Review

Before beginning our work, we researched the topic and stumbled across two academic articles. The first one we researched was from two researchers from the University of Pune [KS16], Vishal Karde and S.S. Sonawane. The two approached the use of sentiment analysis with a business mindset, proposing that sentiment analysis would be useful for businesses to connect with customers. The researchers used two primary approaches for sentiment analysis: Machine Learning and lexicon-based approaches. The techniques they found most effective in their goal were Support Vector Machines, Naive Bayes Classification, Maximum Entropy, and Latent Semantic Analysis from a dictionary domain.

In the paper titled "Emotion Detection using Natural Language Processing" [SKS22], researchers D. SakethNath, H. Kaur, and A. Singh utilized a diverse set of machine learning algorithms for categorizing reviews as positive or negative. Furthermore, in addition to employing SVM and Naive Bayes methods as outlined in the previous study, the researchers also incorporated a Recurrent Neural Network (RNN) for sentiment classification. They noted that the RNN models demonstrated the highest precision, recall, and accuracy scores among all the classification methods. From these two articles, we found their methods to be suitable for our practice, so we decided to employ the following.

3 Proposed Method

We propose four models to predict the intended sentiment of a given tweet about a domestic U.S. airline: Support Vector Machine, Multinomial Regression, Convolutional/Recurrent Neural Networks, and the Bidirectional Encoder Representations from Transformers (BERT) model.

3.1 Support Vector Machine

The first proposed method is that of Support Vector Machines (SVM). SVM specializes in classification problems, more specifically, finding the optimal hyperplane between a set of entries in a high dimensional space that allows for the most accurate classifier. The boundary line is found using a kernel function, which can be tuned to find the best possible way to divide our data. This kernel function

is the key to mapping features in a higher dimension——and is what enables us to distinguish even non-linear classes.

In addition to the kernel function, we also must note the regularization parameter C, the gamma parameter γ, and the kernel coefficient. The regularization parameter examines the trade-off between the margin and classification error. A smaller value of C will lead to a wider margin but also a higher error rate. Conversely, a larger value of C will lead to a smaller margin but runs the risk of overfitting the data. The gamma parameter focuses on the influence of training samples and influences the flexibility of the decision boundary. The kernel coefficient is specific to the type of kernel function employed, whether that be linear, polynomial, or RBF. The kernel coefficient is instrumental in determining the shape and complexity of the kernel boundary. Additionally to these parameters, we also considered the Nu (ν) parameter and penalty term in external models.

We chose SVM as a technique to apply as it is extraordinarily flexible and can handle high dimensionality that may be exhibited by our vectorized text data. Moreover, sentiment analysis requires the ability to capture complex relationships within the text, which SVM's adaptability is perfect for. Furthermore, as we have multiple classes to classify we need a model that can be versatile in the face of noise and ambiguity. While SVM is a simpler method compared to neural networks or transformers, we still found it important to observe the potency of this more fundamental model. The versatility of SVM in adapting to a wide range of parameters, harmonizing with the content of our tweets, makes it a suitable selection method.

3.2 Multinomial Regression

Multinomial regression, often referred to as multinomial logistic regression, is a statistical modeling technique used for predicting outcomes when there are multiple categorical outcomes. This approach is an extension of logistic regression, which is typically used for binary classification tasks. In multinomial regression, the dependent variable is nominal and can represent three or more categories without any intrinsic ordering. The model estimates the probability of each category of the dependent variable, given a set of independent variables, which can be nominal, ordinal, interval, or ratio-level variables.

In the context of sentiment analysis, especially with multiclass text data like tweets, multinomial regression proves to be particularly useful. Sentiment analysis aims to categorize text data based on the expressed sentiment, typically into categories like positive, negative, and neutral. Tweets, being concise and varied in content, pose a unique challenge for sentiment classification. Multinomial regression can handle this effectively by modeling the probabilities of a tweet belonging to each sentiment category. It takes into account the various features extracted from the text, such as word frequencies or the presence of specific keywords or phrases, and uses these to predict the sentiment category. This approach is advantageous because it can handle multiple classes simultaneously and provides a probabilistic understanding of class membership, which is crucial in tasks like sentiment analysis where the context and subtlety of language can lead to varying degrees of sentiment expression.

Incorporating techniques like LASSO (Least Absolute Shrinkage and Selection Operator) and Elastic Net into multinomial regression can significantly enhance the model's performance, particularly in handling high-dimensional data, which is common in text analysis. LASSO is a regularization method that not only helps in avoiding overfitting but also performs feature selection by shrinking some coefficients to zero, thus eliminating less important variables. This is highly beneficial in sentiment analysis of tweets, where the dataset might contain a vast number of features (words), many of which may not contribute significantly to the model's predictive power.

Elastic Net, on the other hand, combines the properties of both LASSO and Ridge regression (another regularization technique). It addresses some of the limitations of LASSO, such as its tendency to select only a single variable among a group of correlated variables. Elastic Net includes both the L1 (penalty term used in LASSO) and L2 (penalty term used in Ridge) regularization terms, making it more robust, especially in situations where there are highly correlated predictors. In the context of multinomial regression for sentiment analysis, Elastic Net can be particularly useful in managing multicollinearity (a common issue in text data due to the presence of synonym words or phrases) while also performing feature selection and regularization. This leads to more accurate and interpretable models, capable of handling the complexity and nuances of natural language in social media text like tweets.

3.3 RNNs and CNNs

For deep learning, we use Recurrent Neural Networks (RNNs) which are known for efficacy in processing sequential data. The primary advantage of RNNs lies in their ability to retain information from prior inputs through their hidden layers. This characteristic is crucial for interpreting human language, where meaning is often influenced by preceding text. In our implementation, the LSTM model sequentially processes the tweet text.

We also employ Convolutional Neural Networks (CNNs), which are recognized for image processing but also utilized in natural language processing tasks. CNNs operate by learning spatial hierarchies of features from input data using convolutional layers. For the context of NLP, CNNs are adept at capturing local contextual patterns, such as specific word combinations or phrases, by applying filters across the word embeddings of the input text. This capability enables the extraction of pivotal textual features necessary for accurate sentiment analysis.

Both models undergo training on the airline dataset, with textual data preprocessed and transformed into a numerical format via word embeddings. We assess the performance of these models based on various metrics to gauge their effectiveness in sentiment classification.

3.4 Transfer Learning

Transfer learning is a machine learning concept that utilizes the insights gained from solving a particular problem and applies that acquired knowledge to address a different yet related problem. In the context of natural language processing (NLP) and sentiment analysis, transfer learning has proven to be particularly effective. The basic idea is to pre-train a model on a large dataset and then fine-tune it on a smaller dataset specific to the target task, such as sentiment analysis.

Bidirectional Encoder Representation from Transformers, or BERT, is a pre-trained language model that was developed by Google researchers [[DCLT18]] in 2018. BERT has achieved remarkable success in sentiment analysis and other NLP tasks like question answering and text prediction.

We assumed that model accuracy was limited by the small size and limited word diversity of our training data. BERT was trained on Wikipedia (2.5 Billion Words) and Google's BookCorpus dataset (800 M words) [[Mul22]]. Transfer learning will help us overcome this hurdle because pre-trained models have already learned to generalize well on diverse text data. BERT's training data has a significantly higher overall word count and text diversity compared to the Twitter sentimental analysis dataset.

3.5 Maintained Assumptions and Methodological Innovations

3.5.1 SVM

With the flexible and moldable nature of SVM, we tested multiple models to find the best classifier for our dataset. Firstly, we considered three support vector classifier functions; a linear model, a degree three polynomial model, and a radial basis function (RBF) model. The idea behind these was to find a general baseline for our model, which we could use for further testing.

Moreover, we subjected our data to a hyperparameter tune-up after we first fitted with the three kernel functions so that we could observe what function it deems to be the best, and confirm if this aligns with our prior tests. Hyperparameterization was done with GridSearchCV with three-fold cross-validation. The C parameter was constrained to values of $[0.1, 1, 10, 100]$. Additionally, the kernel function allowed for all three aforementioned methods, with polynomials reaching up to three degrees. Finally, the gamma parameter was also set to 'scaled', meaning we take the inverse of the number of features we have. We also considered float values of [0.1, 1, 10, 100]. Overall, scaled worked much better for our modeling, and was more efficient.

3.5.2 RNN and CNNs

Our RNN model was designed to capture the sequential nature of textual data in a concise Neural Network. We initialized our model with an embedding layer of dimension 100 to convert word indices into dense vectors of fixed size, capturing semantic information. Following this, we employed two LSTM layers, each comprising 50 units, with L2 regularization at a strength of 0.001 to mitigate

overfitting. Dropout was applied at a rate of 0.7 after each LSTM layer to encourage generalization further.

The model was compiled using the Adam optimizer with a learning rate of 0.0005 and a sparse categorical cross-entropy loss function, suitable for multi-class classification tasks. To optimize the training process, we implemented callbacks such as Early Stopping, with a patience of 3, to prevent overtraining and a Learning Rate Scheduler that adjusted the learning rate according to a decay rate of 0.05. Additionally, ReduceLROnPlateau was employed to reduce the learning rate by a factor of 0.2 if no improvement was observed, with a minimum learning rate set to 0.0001.

The CNN model was designed to capitalize on the model's ability to detect local patterns within the text. We mirrored the embedding dimension of the RNN for consistency and applied a convolutional layer with 50 filters and a kernel size of 3, again incorporating L2 regularization at a strength of 0.0001. A GlobalMaxPooling1D layer succeeded the convolutional layer, ensuring that the most salient features were extracted and forwarded through the network. As with the RNN, a dropout rate of 0.7 was used.

Similarly to the RNN, the CNN model was compiled with the Adam optimizer, a learning rate of 0.0005, and the same loss function. The same Early Stopping and ReduceLROnPlateau callbacks were implemented to monitor the validation loss and adjust the training process accordingly.

The training regimen for both models spanned 10 epochs and utilized the validation set to gauge performance improvements.

3.5.3 Transfer Learning with BERT

The specific BERT model we used, BERTBase, consists of 12 layers of Transformer encoders. Each encoder layer has a multi-head self-attention mechanism and a feedforward neural network. The multi-head attention mechanism allows the model to weigh different parts of the input sequence differently during processing. This method assigns larger weights to words most critical for downstream tasks.

Unlike earlier language models that processed text in a unidirectional manner (either left-to-right or right-to-left), BERT employs a bidirectional approach. It considers both the left and right context of each word in a sentence when making predictions. This bidirectional context is crucial for understanding the nuances and dependencies within the text.

Bert is pre-trained on two main objectives: Masked Language Model(MSM) and next sentence prediction (NSP) [s2]. The Masked Language Model objective tries to predict words that are hidden from the model. For each sentence in the training dataset, 15 percent of words are hidden (or masked) and fed through the model. For the second objective, Next Sentence Prediction, Bert is tasked with predicting the sentence succeding a given input sentence. The training dataset for this task was created using 50 percent correct sentence pairs and 50 percent randomly generated (or incorrect) sentence pairs. BERT is trained to classify each sentence pair as correct or incorrect.

We fine-tuned by adding a classifier layer on top of the pre-trained BERT model. The model was fine-tuned using TensorFlow and Keras, leveraging the Adam optimizer with a learning rate of 2e-6, an epsilon value of 1e-08, and a clipnorm parameter set to 1.0. To address the specifics of the sentiment classification task, we utilized Sparse Categorical Cross-entropy as the loss function, which is particularly suitable when dealing with integer-encoded target labels. The Categorical Cross-entropy loss function , $H(y, p)$, for a single sample is defined as:

$$H(y, p) = -\sum_{i} y_i \cdot \log(p_i)$$

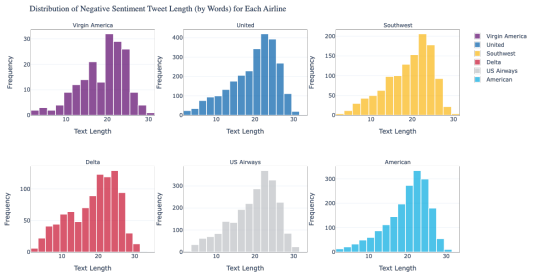
Where y represents the true class label for sample i, and p_i is the predicted probability for class i.

4 Data Analysis

4.1 Exploratory Data Analysis

A crucial step in our preliminary analysis involved visualizing the frequency of words within tweets categorized as positive, neutral, and negative. This was accomplished through the generation of word clouds, which depicted the most prominent words in each sentiment class shaped into an airplane silhouette.

The word clouds revealed distinct language patterns associated with each sentiment. In the positive category, words like "thank," "great," and the names of several airlines appeared frequently, reflecting appreciation and satisfaction. Neutral tweets often contained words such as "flight" and "now," which may be indicative of neutral statements or queries about flight status. In stark contrast, the negative sentiment word cloud was dominated by terms such as "canceled," "delayed," and "rude," highlighting the frustrations and issues faced by passengers.



(a) Tweet Length by Words For Negative Sentiment

(b) Tweet Length by Words For Positive Sentiment

Figure 1: Exploratory Data Analysis

In addition to the word frequency analysis, the class distribution was examined through a pie chart, which underscored the prevalence of negative sentiment within the dataset, constituting 63.1% of the tweets. Neutral and positive sentiments were represented by 21% and 15.9% respectively. The pie chart accentuated the class imbalance; however, the disparity was not deemed significant enough to necessitate the reparametrization of our labels. This decision was informed by the consideration that the natural distribution of sentiments in the dataset could reflect a realistic scenario where negative experiences are more frequently reported in airline-related tweets.

Separated by negative and positive sentiment, we charted the distribution of words in a tweet by the airline. The plots provided were cleaned to not include any hyperlinks or other outliers. From these plots, we wanted to see if there was any difference between text length and airline. Looking at both plots side by side, it appears that negative tweets are left skewed, indicating that perhaps the number of words in a negative tweet is greater than that of positive sentiment tweets. Running a Kruskal Wallis test, we determined that there was no significant difference in text length between sentiments at $\alpha = 0.05$.

H_0 : The median text length is the same across all airlines. vs. H_α : The median text length is different for some airline(s)

4.2 Details of the Main Results

4.2.1 SVM

Table 2: SVM Base Models

Method	Accuracy	$Recall_{\text{Weighted Avg}}$	$F1_{\text{Weighted Avg}}$	ROC
Linear	78.78%	78%	79%	88.00%
Polynomial	65.90%	67%	66%	74.00%
RBF	78.92%	79%	79%	88.00%

Using a base support vector classifier, we fitted our data using three kernel functions: linear, polynomial, and RBF. The linear kernel achieved an accuracy of 78.78%, the polynomial achieved an accuracy of 65.90%, and the radial basis function achieved an accuracy of 78.92%. From the preliminary results, we might favor RBF over the other kernel methods. Comparing RBF to the linear kernel, we observed

very similar results. The slight increase in accuracy makes RBF appealing to use in this first case, but this of course is just a base model. For further testing, we turned to hyperparameterization. Hyperparameterization with GridSearchCV allows us to pick the optimal regularization parameter (C), kernel function, and gamma parameter. In doing so, we allow ourselves to select the most optimal version of our model for each variation in SVM.

Table 3: SVM Hyperparameterized (Italicized = Optimal)

Method	Accuracy	$Recall_{\text{Weighted Avg}}$	$F1_{\text{Weighted Avg}}$	ROC
Base Hyperparameter — ***Linear***	78.61%	78%	79%	87.95%
Tuned 2nd Degree Polynomial	78.37%	78%	78%	88.00%
Tuned RBF	78.92%	79%	79%	88.32%
Linear Penalty	78.92%	78%	79%	89.00%
Nu Support — ***RBF***	79.06%	78%	79%	89.00%

Applying hyperparameter tuning to our data, we allow for an iterative process that seeks to evaluate multiple SVM models and return the predictions and parameters of the best-fitting model. We first applied the hyperparameterization method with three cross-validation folds to pick among the following: Kernel = (Linear, Polynomial, RBF), $C \in [0.1, 1, 10, 100]$, γ = Scale, Auto. With all of these being an option, we found that the model identified a linear kernel with the following parameters: Kernel = Linear, $C = 1$ and γ = Scale as the best-fitting configuration. This result is listed in the table as the *Base Hyperparameter — Linear* model. It performed well with a 78.62% accuracy, but this was still just in the same ballpark as the base version. For testing's sake, we also fit a tuned second-degree polynomial model and an RBF as well. The RBF surprisingly had a higher accuracy than the aforementioned base model. Next, we applied hyperparameterization to a Linear Penalty model and a Nu Support. The linear penalty model introduces L1 Regularization, intending to prevent overfitting. This model did not do much, and it predicted the same exact way as our base hyperparameter as it used the same model parameters. With Nu Support however, we saw that the accuracy improved a tad and the ROC was the highest we had. The added Nu parameter was able to take values of (.1, .5, .9). Nu Support works to find a more flexible trade-off between minimizing errors and allowing flexibility, which was helpful given our sparse matrix. From this stage, we saw that Nu Support was our best performer; however, all methods were very similar to one another.

Table 4: SVM Bigram (Italicized = Optimal)

Method	Accuracy	$Recall_{\text{Weighted Avg}}$	$F1_{\text{Weighted Avg}}$	ROC
Base Hyperparameter — ***RBF***	78.89%	78%	79%	87.96%
Linear Penalty	78.40%	78%	78%	87.96%
Nu Support — ***RBF***	78.68%	78%	79%	87.94%

We also wanted to experiment with the use of N-Grams in SVM. We focused on the usage of bigrams, which treat the tokens in clumps. For instance, the phrase "thank you" would be read together as one word. For this reason, N-Grams are suitable for SVM as we can see whether or not phrasal sentiment leads to higher accuracy. We ran the test using a hyperparameterized model, a linear penalty model, and a Nu Support model. From our results, we saw that the hyperparameterized model yielded the best accuracy. Additionally, it chose parameters $C = 100$ and an RBF kernel function. However, compared to our previous model, we can see that the N-Grams performed slightly worse compared to the original unigram versions. This may be due to the sparsity of the resulting matrix, increased dimensionality, or the noise from internet slang. Looking at the tweets from the dataset, we see some forms of speech that might not be able to have perfect sentiment analysis, which could give creedence to our results.

The best results from all of our testing saw Nu Support be the most suitable for our data; however, we still only saw an accuracy hovering around 80% across all of our modelling. So while our results show that SVM is suitable for sentiment analysis, it might also need to be more fine-tuned and diagnosed to work perfectly.

4.2.2 Multinomial Regression

Model	Accuracy	F1 Score	ROC-AUC	LOG-LOSS
Standard	79.90%	0.785	0.915	0.511
LASSO	79.06%	0.782	0.905	0.510
Elastic Net	80.31%	0.791	0.925	0.500

Table 5: Multinomial Models' Results without N-Grams

Without N-Grams The comparative analysis of different multinomial models applied to a text classification task, without incorporating N-grams, reveals distinct performance characteristics based on various evaluation metrics. The Standard Multinomial Model exhibits a commendable accuracy of 79.90%, indicating its proficiency in correctly predicting the text data's class nearly 80% of the time. Its F1 score of 0.785 suggests a good balance between precision and recall, while the ROC-AUC score of 0.915 reflects a strong ability to differentiate between classes. The Log-Loss of 0.511, although not minimal, points towards a reasonable level of confidence in its predictions.

In contrast, the LASSO-regularized Model shows a slight dip in accuracy (79.06%) compared to the standard model, hinting at a marginal reduction in overall predictive capability. The F1 score is almost parallel to the standard model, maintaining a similar balance in precision and recall. The ROC-AUC score, at 0.905, though slightly lower, still indicates good class separation capability. The Log-Loss value, marginally better at 0.510, suggests that the LASSO regularization has not significantly affected the model's confidence in its predictions.

The Elastic Net-regularized Model, however, emerges as the most effective among the three. It not only surpasses the standard and LASSO models with an accuracy of 80.31%, but also achieves the highest F1 score of 0.791, indicating an improved balance of precision and recall. Its ROC-AUC score of 0.925 is the highest, denoting an exceptional ability to distinguish between different classes. Furthermore, the lowest Log-Loss score of 0.500 among the models signifies that this model is the most confident and accurate in its predictions. This superior performance underscores the advantages of combining L1 and L2 regularization, as seen in the Elastic Net approach, particularly for text classification tasks like this one.

Model	Accuracy	F1 Score	ROC-AUC	LOG-LOSS
Standard	79.90%	0.788	0.915	0.511
LASSO	78.78%	0.778	0.904	0.512
Elastic Net	80.17%	0.794	0.911	0.492

Table 6: Multinomial Models' Results with N-Grams

With N-Grams The inclusion of N-grams in multinomial models for text classification reveals nuanced changes in their performance, as reflected in various metrics. The Standard Multinomial Model with N-Grams maintains its accuracy at 79.90%, identical to its performance without N-grams, indicating a consistent predictive ability. There's a slight enhancement in its F1 score to 0.788, suggesting improved precision and recall balance, while the ROC-AUC remains steady at 0.915, affirming its strong class differentiation capability. The Log-Loss metric also remains stable at 0.511, indicating no significant change in the model's confidence in its predictions.

The LASSO-regularized Model, however, experiences a slight decline with the integration of N-Grams. Its accuracy drops to 78.78%, and the F1 Score decreases to 0.778, hinting at a marginal reduction in both overall accuracy and the balance of precision and recall. The ROC-AUC score dips slightly to 0.904, suggesting a reduced but still competent ability to differentiate between classes. The Log-Loss increases marginally to 0.512, indicating a slight increase in prediction uncertainty.

Contrastingly, the Elastic Net-regularized Model with N-Grams demonstrates a notable improvement. It achieves an accuracy of 80.17%, surpassing both the standard and LASSO models, and its F1 Score increases to 0.794, indicating the best balance between precision and recall among the three models. The ROC-AUC, at 0.911, although slightly lower than its non-N-gram counterpart, still denotes a robust class differentiation ability. Most impressively, the model achieves the lowest Log-Loss score of 0.492, underscoring the most confident and accurate predictions among the compared models.

In summary, the integration of N-grams into these multinomial models shows diverse impacts. While the Standard Model demonstrates stability, the LASSO Model slightly underperforms with N-grams. The Elastic Net Model, in contrast, shines in this setup, illustrating the effectiveness of its regularization approach in managing the complexity added by N-grams in text data.

4.2.3 RNNs and CNNs

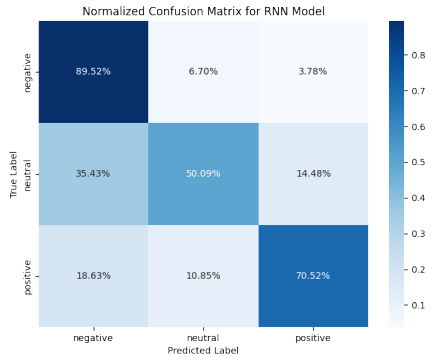
Model	Accuracy	F1 Score	ROC-AUC	LOG-LOSS
RNN	78.8%	0.78	0.89	0.56
CNN	80.5%	0.79	0.89	0.53

Table 7: RNN and CNN Model Results

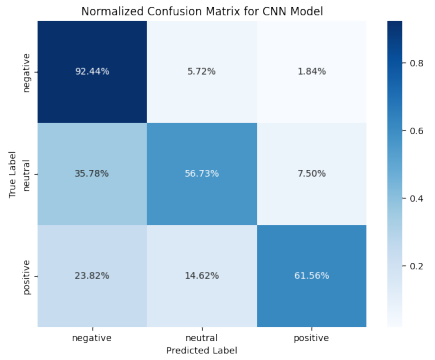
The RNN model, utilizing LSTM layers known for their efficiency in capturing temporal dependencies, achieved an overall accuracy of 78.8%. This model demonstrated a robust capability to discern the negative sentiment, as evidenced by the confusion matrix, with an 89.52% accuracy in classifying negative tweets correctly. However, it exhibited moderate performance in recognizing neutral and positive sentiments, achieving 50.09% and 70.52% accuracy in these categories, respectively. This model's F1 score stood at 0.78, with an ROC-AUC of 0.89 and a Log-Loss of 0.56, indicating a strong model but with potential room for improvement in balancing the classification across sentiments. The ROC criteria was treated using the one vs all method.

In contrast, the CNN model, which applies convolutional filters to capture local contextual patterns in text data, slightly outperformed the RNN with an overall accuracy of 80.5%. The confusion matrix for the CNN model indicates a higher precision in identifying negative sentiment, with 92.44% accuracy. It also showed superior performance in classifying neutral but was slightly worse for positive sentiments, with accuracies of 56.73% and 61.56%, respectively. The F1 score for the CNN model was marginally higher at 0.79, along with an equal ROC-AUC of 0.89 and a lower Log-Loss of 0.53.

The training progression over epochs further substantiates these findings. The RNN model exhibited a steady increase in validation accuracy up to the third epoch, after which a divergence between training and validation accuracy suggested the onset of overfitting. Conversely, the CNN model maintained a closer convergence between training and validation accuracy throughout the epochs, peaking at the fifth epoch with a validation accuracy of 80.54%.



(a) RNN Confusion Matrix



(b) CNN Confusion Matrix

Discussing Results The superior performance of the Convolutional Neural Network (CNN) model in our sentiment analysis task was surprising, as we assumed that Recurrent Neural Networks (RNNs) would be the go-to model for sequential data. However, the architecture of CNNs allows them to excel in identifying local patterns, which can be particularly advantageous in text sentiment analysis. This is especially effective in datasets like ours, where emotive expressions, specific adjectives, and phrases are strong predictors of sentiment. The convolutional layers can extract these salient features from

short segments of text, regardless of their position within the tweet, providing a form of automatic feature engineering that is highly relevant for classifying sentiment.

In our dataset, which consists of tweets that are relatively short in length, CNN's ability to capture essential contextual clues without the need for understanding the entire sequence could explain its better performance. For shorter sequences of text, the local context captured by CNNs can be effective for sentiment classification, and for longer passages of texts, RNNs may prove to be a better classifier.

Our results show that CNNs can be more effective than RNNs for certain sentiment analysis tasks. It also suggests that the ability of CNNs to perform feature extraction on smaller text data is a powerful tool that can yield significant advantages in natural language processing challenges.

4.2.4 Transfer Learning

The fine-tuned BERT model had an accuracy score of 90 percent. It has the highest accuracy score among all the other models we tested.

Class	Precision	Recall	F1-Score	Support
Positive	0.88	0.87	0.87	305
Negative	0.93	0.96	0.95	686
Neutral	0.86	0.82	0.84	323
Accuracy			0.90	1314
Macro Avg	0.89	0.88	0.89	1314
Weighted Avg	0.90	0.90	0.90	1314

Table 8: Classification Report for fine-tuned BERT model

From the classification report, we can observe that our model does a better job of predicting negative sentiments compared to positive and neutral sentiments. Positive and neutral classes have lower precision and recall scores in comparison to the negative classes. Tweets with negative sentiments usually are complaints about rude service, delayed flights or bad quality of food. As a result, classifying negative tweets is easier for the model. Positive and neutral tweets, however, have increased text diversity. This makes classifying between positive and neutral sentiments harder. The confusion matrix validates this idea. There aren't false positives for negative sentiments. Positive and neutral sentiments have many more false positives in comparison.

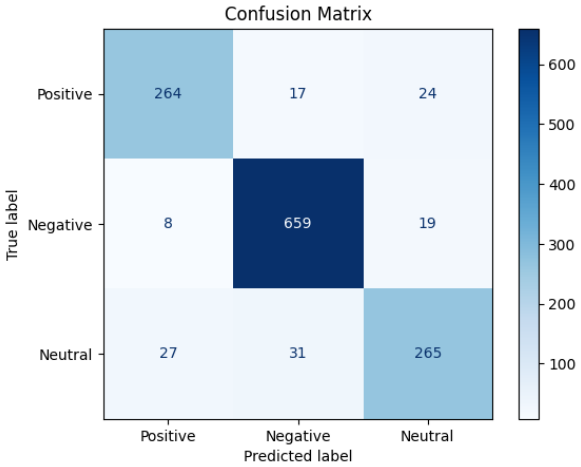


Figure 2: Confusion Matrix for the BERT Model.

The variation in sentiment classification scores can be attributed to factors beyond limited text diversity. Firstly, an overrepresentation of negative sentiment in the dataset may account for the observed differences. Secondly, the lower precision and recall scores for positive and neutral sentiments might stem from their smaller sample size. Another potential factor is the accuracy discrepancy in annotating positive and neutral sentiments compared to negative sentiments. This is evident in the

sentiment confidence column, where numerical representations indicate lower confidence for positive and neutral labels.

5 Conclusion and Discussion

In concluding our analysis of sentiment in U.S. domestic airline tweets, the models evaluated here present a clear picture of the strengths and limitations to our different approaches to sentiment analysis. Among these, the fine-tuned BERT model emerges as the top performer, with an impressive accuracy of 90%. Its advanced architecture, leveraging deep bidirectional understanding, enables it to excel in the challenging task of sentiment analysis, especially with neutral labeling. However, we would like to acknowledge the valuable insights offered by other models as well.

The SVM models, known for their flexibility in handling complex relationships within text, also played a critical role in our analysis. Though they didn't reach the heights of BERT in accuracy, they offered valuable insights into the utility of different kernel functions and parameter tuning in sentiment classification.

The Elastic Net Multinomial Regression model, with a combination of L1 and L2 regularization, showed impressive effectiveness, particularly when incorporating N-grams. It achieved commendable accuracy and balanced metrics such as F1 Score and ROC-AUC, making it a strong contender in the realm of predictive modeling for text data. The LASSO-regularized model, while slightly underperforming in comparison, provided an essential perspective on the role of feature selection in handling high-dimensional data. In future analysis, we would like to investigate the Ridge penalty standalone, as the L2 norm had a much larger penalizing coefficient compared to that of the L1 norm.

Lastly, the deep learning approaches using RNNs and CNNs demonstrated the importance of model architecture in handling sequential data. While they didn't surpass BERT, their performance highlighted the potential of deep learning in capturing contextual information in text, albeit with different emphases – RNNs on sequential data and CNNs on local patterns.

Each of these models contributes uniquely to our understanding of sentiment analysis in the context of airline tweets. While BERT stands out for its overall performance, the other models provide valuable lessons in the nuances of natural language processing and the importance of choosing the right model and approach for specific data characteristics and analytical goals. We look forward to expanding this knowledge with future projects in Natural Language Processing.

6 GitHub Repository

The code files for this report can be found by clicking **here** for reproduction and results.

References

[Asq] James Asquith, *Analysis—how much money do empty flights really cost airlines.*

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).

[kag19] *Twitter us airline sentiment*, 2019.

[KS16] Vishal A. Kharde and S.S. Sonawane, *Sentiment Analysis of Twitter Data: A Survey of Techniques.*

[Mul22] Britney Muller, *BERT 101 State Of The Art NLP Model Explained*, March 2 2022.

[SKS22] Damam SakethNath, Harjeet Kaur, and Arun Singh, *Emotion Detection using Natural Language Processing* .