Sentiment Analysis of Airline Tweets During COVID-19 pandemic

Amy Wang, Anastasios Avramidis, Chin-Chia Leong, Jiacuo Danzeng,

Jorge Lopez Gama, Qianwen Zheng, Yubo Zhang

[*] *CIS 9665, Applied Natural Language Processing*

[+] *Zicklin School of Business, Baruch College*

December 13th

Introduction

A lot has changed since the start of the COVID-19 pandemic, and the lives of all people worldwide have transformed to adapt to the circumstances of this unique event. Restrictions got put into place which heavily impacted the social and business life of everyone and in the process, some businesses got hurt while others thrived on the new landscape created to combat this global event. One of the business sectors which got hurt the most was travel and by extension the airline industry.

Motivation

In this project, we aim to analyze what impact, if any, this new global event had on consumer sentiment for the airline industry. We strive to find answers on that front because the airline industry is struggling right now. To survive these difficult times, any airline must be very well informed on how people feel and what they value most in their air travel experience given the new circumstances. However, we are not limiting our scope to answering just if COVID-19 has an impact on people regarding air travel, since we have the opportunity to further investigate the sentiment of the people towards specific airline brands and the airline industry as a whole, and create tools that can be used for further analysis or reconfigured to similar purposes in the future.

To achieve the aforementioned goal we collected and analyzed tweets during a certain period from Twitter.com which are referring to airline brands. Since the amount of tweets collected is in the thousands and we couldn’t realistically review all of them to label them manually we developed and evaluated several machine learning algorithms to help us predict the sentiment of each tweet and categorize them as positive, neutral or negative.

Text Dataset Description

Two different datasets were used in the project. A dataset of 14640 airline tweets that had been labeled was obtained from Kaggle and been used to train our various models as well as test their performance. The second dataset, API dataset, consists of all 95614 tweets that mentioned one of several airlines during the period of November 08, 2020 until December 02, 2020. It was gathered using the python tweepy library and the twitter search api, outputting as json format.

Text Processing Description

As all the tweets gathered using the Twitter API came in json format we first had to extract the tweet text and other relevant features using the python json library. Before processing the text, feature engineering was performed using RegEx and other NLTK functions in order to gather the ratio of punctuation to word count, the number of expletives used, any hashtags used, and the number of uppercase words for each tweet. These features contained information that would be lost during the text processing but would prove useful in classification.

Additionally, the tweet text was processed using NLTK functions, this process involved word tokenization, removing stopwords, setting all works to lowercase, and lemmatization. The Sklearn count vectorization function was then used on this processed text to create a bag of words matrix that contained the 1000 most frequent words across all tweets that would later be used for some machine learning models. A similar bag of words matrix was created for the 250 most frequent hashtags.

Exploratory Data Analysis (EDA)

After text processing, we applied EDA to better understand our datasets. We considered EDA as a necessary step before building our model in order to select important features.

Through the process, we found that the API dataset was composed of four main airlines and the number of tweets had a positive relationship with the number of passengers by observing peaks appeared during weekends and the thanksgiving vacation (shown in Figure 1). This finding proved the business value of our project. We also discovered that the sentiment composition of Kaggle dataset had over 60% negative reviews(shown in Figure 2). This finding provided us a baseline to judge our models on further evaluation. Moreover, we made a chart (shown in Figure 3) that showed the percentage of sentiment for each airline. We found out that US Airways, American Air, and United Air were considered as poor performance by seeing the number of their negative tweets are a lot higher than other airlines. This provided us a direction for our further research and possible topics we would want to explore. In the real world, it is important to know who will be interested in our research, and potential sponsors of our research.

Figure 1:

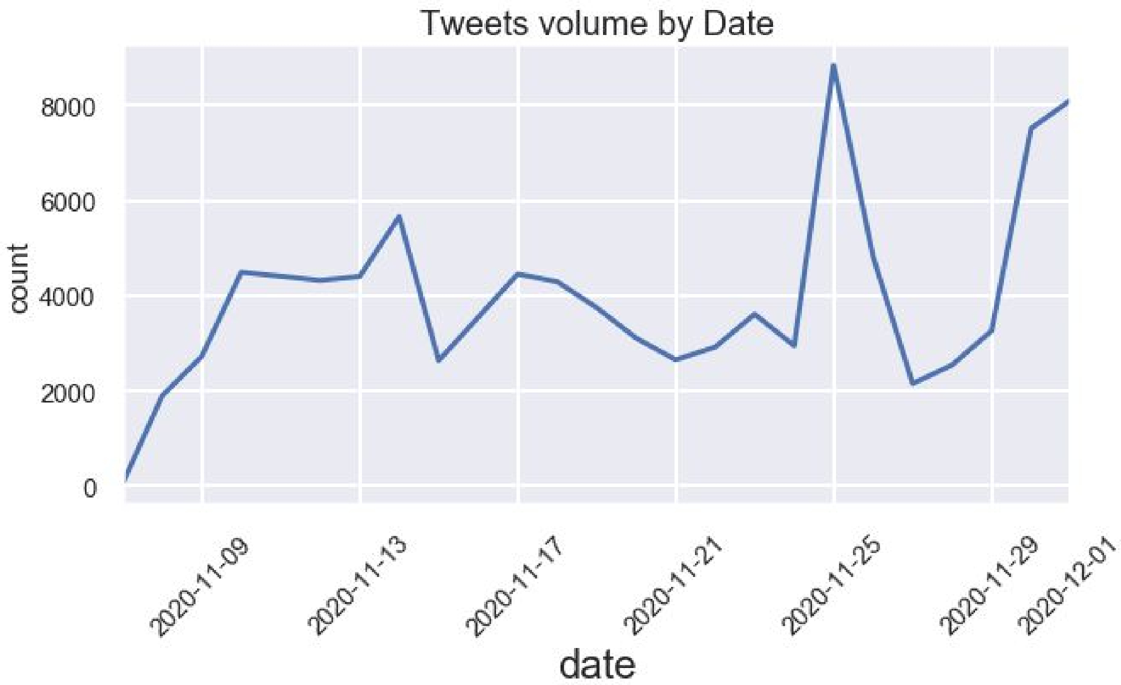


Figure 2:

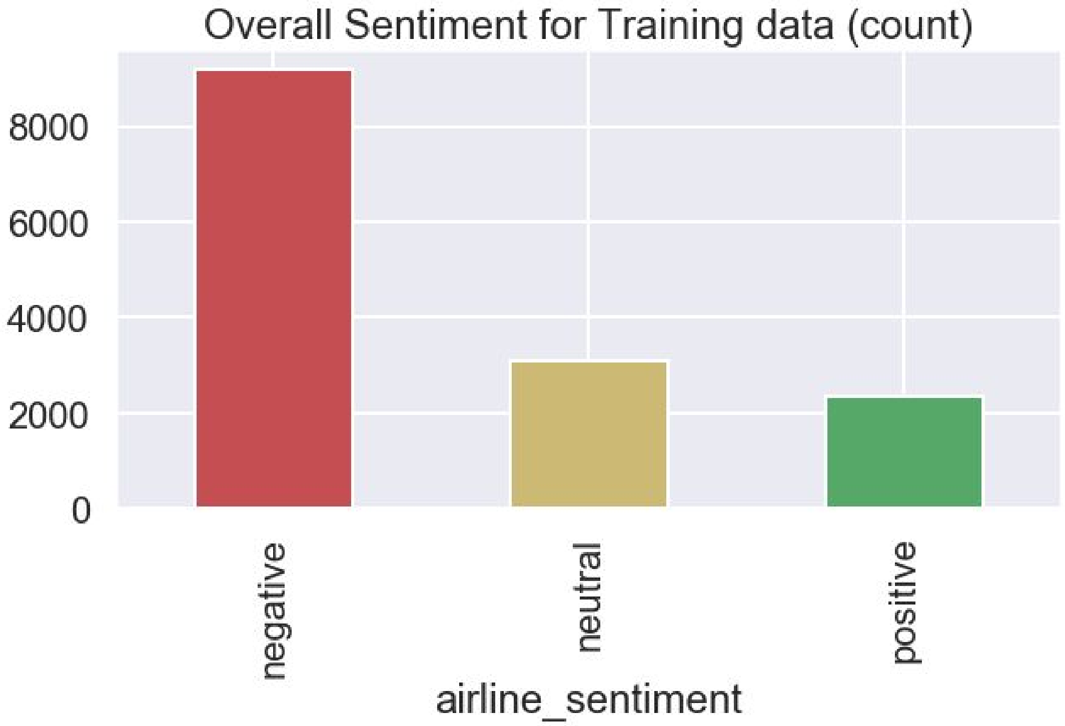
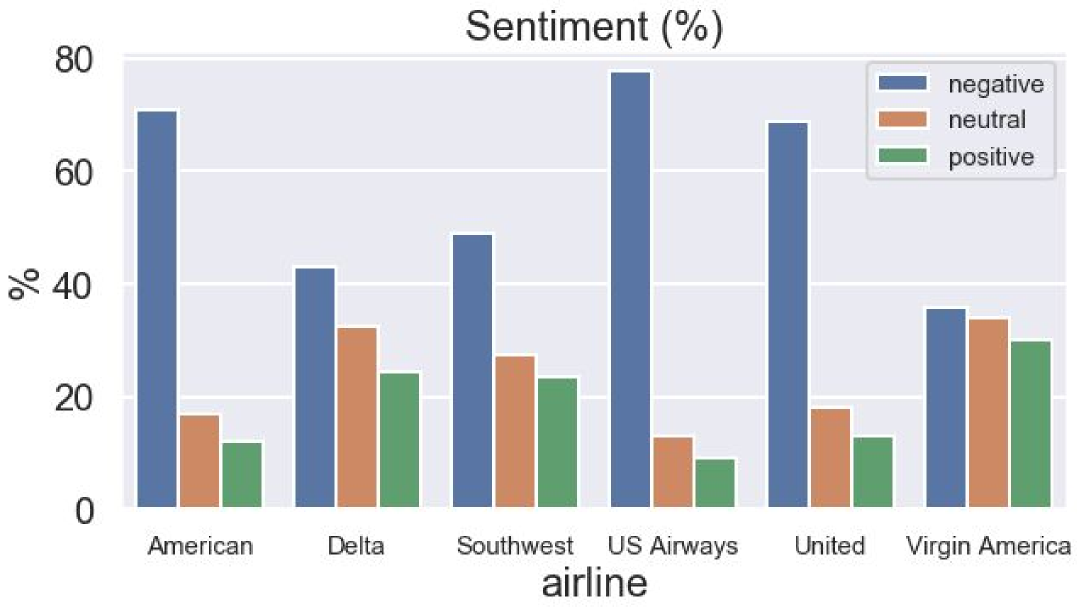


Figure 3:



Text Analytical Method Description

Machine learning Model comparisons

There are several machine learning models available to do sentiment analysis, and we used several that we are familiar with in order to compare how they differed in performance in a natural language processing setting. The classification models compared were Naive Bayes, Random Forest, Logistic Regression, Elastic Net Logistic Regression models, as well as 3 neural networks.

Naive Bayes is a relatively simple classification model that calculates the prior probability using the frequency of each variable in the training set. The probabilities are calculated using Bayes theorem and are assumed to be independent of each other. Once the associated probabilities are all calculated they are all used in combination to classify a new tweet based on the label with the highest estimated probability.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. This way the outcome is protected by stray errors created by the random state of individual trees since with this method a large number of uncorrelated models operate as a committee, outperforming their individual counterparts by complementing each other.

Logistic Regression is a classification model that uses a logistic function of the form:

$$\pi = \frac{e^{\beta X}}{1-e^{\beta X}}$$

where β is a vector of coefficients and X is a vector of features, to model the probability of a label based on the value of the features. An alternative way to frame the Logistic Regression model which gives us the log-odds instead and is much easier to interpret is:

$$\frac{\pi}{1-\pi} = \beta X$$

In this form a one unit increase in a feature increases the log-odds. In our specific application it would mean that the frequency of a word would increase the log-odds of being classified as a negative,positive or neutral by the respective coefficient depending on the model. While this model is binary in nature, it can easily be extended to the multinomial classification case.

Regularization versions of the logistic regression model such as Elastic Net, Lasso, and Ridge Logistic Regression work by limiting the values that the coefficient vector takes on. This has many benefits as it greatly reduces the variation of the coefficient's values if we were to train it on a different dataset, and in a sparse dataset arising from the use of a bag of words matrix it performs variable selection by shrinking coefficients that are not important to zero. This allows us to see which variables most influence the probabilities assigned. Ridge regularization typically does not set any coefficients to zero, and Lasso tends to choose the bare minimum amount of coefficients and if there is any correlation between features it will choose one and shrink the rest to zero, Elastic Net does not suffer from this problem which is why it was selected to evaluate the contribution of each feature to the probabilities assigned. 10 fold cross validation is used during the model training in order to choose λ and α, which control the coefficient shrinkage and the degree of l2 or l1 penalty applied to the model respectively.

Elastic Net Logistic Regression Coefficients

As mentioned previously, one of the benefits of the elastic net model is that we can examine the coefficients to see the features most strongly associated with positive & negative tweets. The top 10 features by coefficient value are shown in the tables below, an increase in the corresponding feature by a unit of one indicates that the estimated log-odds of the tweet increase by the corresponding coefficient value. In the case of words this would mean that the estimated log-odds increase by the value for each additional count of the word.

Positives		**Negatives**	
Word/Feature	**Coefficient Value**	**Word/Feature**	**Coefficient Value**
Thank	2.83	Hour	1.73
Great	2.23	Worst	1.66
Awesome	2.21	Number of Expletives	1.55
Punctuation Ratio	2.11	Delayed	1.54
Amazing	1.85	Luggage	1.48
Best	1.84	Hold	1.30
Love	1.76	Never	1.22
Kudos	1.24	Nothing	1.19
Appreciate	1.20	Cancelled	1.19
thx	1.08	Delayed	1.18

As we can see, for the negative coefficients the model assigned the highest value to words that intuitively would appear in the tweet of someone that is frustrated with an airline, and give us some insight into why people are upset. Cancellations, luggage problems, and delays are some problems identified by the model as being customer concerns. The positive coefficients on the other hand are words that would intuitively be associated with a positive sentiment but , at least the top 10, do not give any insight into the reasons customers are satisfied. The elastic net model also ignores words that appear frequently in all labeled tweets, leaving only those words that are strictly associated with positive or negative tweets.

RNN Model Architecture & Evaluation

This section will explore two RNN based models -- LSTM and the ULMFiT (Universal Language Model Fin-tuning for Text Classification) by explaining text-preprocessing, model architecture, and the model evaluation process.

Baseline LSTM Model:

The baseline model used Pytorch's natural language processing library Torchtext. The first step is to preprocess the text to be ready for the model. Specifically, it created a new dataframe using the two columns -- text and airline sentiment. It also transformed the existing airline sentiment labels from neutral, positive, and negative to 0, 1, and 2, respectively. It also removed symbols such as @ and stopwords using NLTK stopwords at this stage. This new dataframe is then saved into a new csv file, ready to be further processed.

The second step is to use Field () function in Torchtext to further preprocessing text. In our model, it applied the SpaCy tokenizer, used lower-cases and set batch-first to true to have the first dimension of input and output to the bath size. In Torchtext, it then used TabularDataset to load our newly defined csv file as the training data. Another essential step before model training is to build vocabularies. We used GloVe as a pre-trained word embeddings and set the minimum word frequency to three to build the vocabularies for TEXT, which has the vocabulary size of 5037 and 3 for the Label vocabulary size. The model is trained on a Colab notebook using its GPU. To load our transformed data, we used the BucketIterator, which reduces the amount of padding needed.

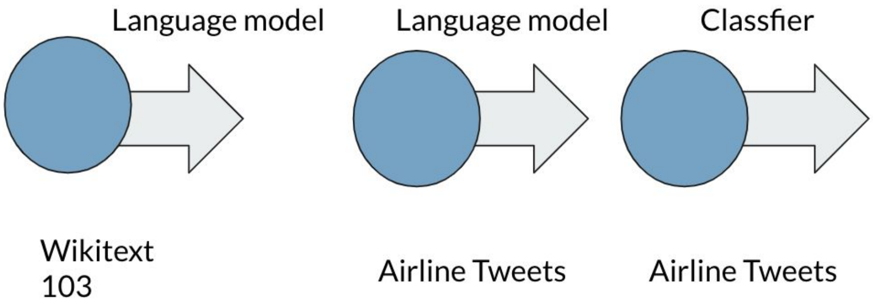
The inputs of the baseline LSTM classifier will be passed to the embedding layer, which uses the GloVe pre-trained embeddings and the embedding size of 100, representing the number of dimensions for each word. The next layer is the LSTM layer with a hidden dimension of 32 and bidirectional, followed by a dropout rate of 0.2, which essentially improved the model accuracy after adding the dropouts during the fine-tuning stage. It then follows two fully connected dense layers with a Sigmoid activation function applied to the output layer. It used the Adam weight decay regularization for optimization and the CrossEnthrophy for loss function since it is a multi-classes text classification issue.

ULMFiT Model:

For the second model, we applied the ULMFiT architecture to our training dataset, a state-of-the-art model in natural language processing first published in 2018 by Jeremy Howard. In our case, this model is built on top of the Pytorch API -- simplifying the process of building the entire model architect from scratch compared to the baseline model mentioned above. Three key advantages of this model include 1) it does not require a custom feature engineering or pre-processing, 2) ULMFiT provides a universal approach in different NLP tasks, 3) it utilizes transfer learning in NLP - an approach that benefited computer vision research for many years.

Specifically, the model used the Wikitext pre-trained model and fine-tuned with our airline tweets dataset (shown in Exhibit 1). Despite having a much smaller dataset, our classifier takes advantage of the pertain weights trained from Wikitext's over 100M tokens. Another key differentiator of the ULMFiT model is training one layer at a time instead of all layers for each epoch. Our model was unfreezed three times and trained on four epochs -- achieving an 85.7% validation accuracy.

Exhibit 1:



CNN Model Architecture

CNN, Convolutional Neural Network, is also an implementation of neural networks as RNN in the previous paragraph. A major reason why RNNs are comparatively slow is that each word in their input strings must be processed sequentially. For instance, in the sentence "Parallelization is fun", the words 'parallelization' and 'is' must be processed before the word 'fun'. In contrast, all the elements in a CNN are processed simultaneously, which can speed up processing immensely. One way to speed up the training time is to improve the network by adding a "Convolutional" layer. CNN originally comes from image processing. It passes a "filter" over the data and calculates a higher-level representation. It has been shown to work surprisingly well for text, even though they have none of the sequence processing ability of LSTMs.

For our CNN model, the first layer is the embedded layer that uses 100 length vectors to represent each word, vectorize consumer complaints text, by turning each text into either a sequence of integers or into a vector and set the max number of words in each complaint at 280. Next with Conv1D from Tensorflow, creating a 1D CNN model which has 96 filters and 5 kernels. The next layer is the LSTM layer with 100 memory units. The output layer must create 3 output values, one for each sentiment. Additionally, because it's a multi-class classification problem, softmax is used as activation function and categorical_creossentrop is used as the loss function.

Model Evaluation

The naive bayes, random forest, and both logistic regression models were trained with 1255 features, the first 5 features were the retweet count, the number of uppercase words, the uppercase to total words ratio in a tweet, and the number of expletives in each tweet. Features 5-1005 was the count of the top 1000 words per tweet, and the last 250 features were the count of the top 250 hashtags per tweet.

The results for each model are presented below, the fit time is included as well due to the major differences and the trade off between some of the more accurate models and fit time. The Kaggle dataset was split into 10248 rows for training and 4392 rows for testing (70/30 split).

Model	Fit Time	Test Accuracy
Naive Bayes	0.1 seconds	76.5%
Random Forest	53.12 seconds	76.2%

Logistic Regression	0.58 seconds	75.7%
Elastic Net	4 hours	78.8%
CNN	3 mins	81.9%
RNN-LSTM	5mins on GPU	71.4%
ULMFiT	25 mins	85.7%

All the models had relatively similar performance. The ULMFiT model has the best performance. It should be noted that the dataset used to train the models only consisted of 10248 rows and features, classification with more than the 1000 words and 250 hashtags that appeared more frequently in the tweets could result in improved accuracy but was not possible for this project due to computational limits.

For RNN model fine-tuning, we explored the following approaches (shown in Exhibit 2). Two approaches helped to improve the accuracy 1) Increased training epochs from 10 to 20. 2) Adding dropouts after the embedding layer.

Exhibit 2:

Epoches	**10**	**20**	**20**	**20**
Fine-tuning	baseline	increased 10 epoches	used ReLu()	Added a dropout layer
Validation Acc	67%	67%	68.12%	71.47%

Results and Conclusions

After extensive testing, seeking the most consistent classification model to help us classify the tweets we previously collected, we decided to use two of the best performing models to predict the sentiment of the api tweets. Those are the Elastic Net logistic regression model and the Convolutional Neural Network (CNN). Despite the ULMFiT model achieving better results, its current library only supports batch prediction instead of providing a solution to return all predicted labels at once. Thus for efficiency, we decided not to proceed with this model.

The reason we chose two models instead of one was to see if the outcome would be substantially different given the fact that the predictive accuracies of most of our models are somewhat similar. As it turned out after labeling the unseen data there was a substantial difference in the classification of our data. The Elastic Net produced a distribution of approximately 83% negative 12% neutral and 5% positive tweets while CNN produced a distribution of approximately 64% negative 26% neutral and 10% positive tweets. To further investigate and understand which of these two results is the most reliable one, we run frequency distributions of individual words and bigrams. The distribution results helped us identify a few things about both our models and our study.

At first, we noticed the surprising lack of COVID-19 related mentions in the tweets. It seems like the pandemic itself may have had a severe impact on the industry, but the changes forced into it didn't change the core experience that determines the sentiment of people towards airlines at least not at this time.

Secondly, while the top words or bigrams using either method were not wildly different in the negative reviews included tokens like "Sorry" or bigrams like "never asked", the positive reviews resulting from the convolutional neural network included tokens like "Thank" or bigrams like "Great news" which obviously point towards a positive sentiment, but the tokens or bigrams from the positive reviews resulting from the elastic net didn't show a clear indication towards a positive sentiment. This observation served as an explanation as to why the elastic net is more negatively biased since it seems to have problems identifying positive tweets and also possibly having problems classifying neutral ones correctly as well.

The convolutional neural network while having similar accuracy produced more reliable results after close inspection due to the aforementioned observation about the frequency distribution and due to the fact that the distribution of its results follow closer to those of our original training dataset since we have no reason to believe the distribution changed between now and then changed significantly.

Practical Implications

Throughout this project we analyzed text data and developed machine learning algorithms to identify the sentiment of consumer tweets, discovering what are determinants. Trying to figure out which factors contribute to people's happiness is essential for any business, especially in trying times like the present when a pandemic threatens not only our public health but the profitability and viability of businesses in all industries. Airlines being one of the most heavily impacted businesses by that fact can benefit from our analyses and tools we developed in many ways.

As we mentioned in the results part of our analysis the convolutional neural network we built can produce acceptable results predicting the sentiment of tweets. As a result, an airline can use the same tools as we did to gather tweets and use our model to classify them continuously in order to possess up-to-date information always about consumer sentiment and what is being discussed on Twitter based on the industry or its brand.

During our analysis, we saw that the most frequent words used in tweets regarding the airline industry are the brand names of airlines. The easiest thing an airline can understand from classifying and analyzing tweets is the people's opinion about their brand depending on whether they are mentioned mostly on positive, negative, or neutral tweets. Airlines can further expand on negative tweets and define problems in real-time, defining problems in early stages, and making reactions.

Another implication would be to search for words, bigrams, or trigrams which are associated with certain parts of the airline experience and analyze whether they appear a lot in tweets and which sentiment is usually associated with the tweets they appear on to identify key areas that are important to their consumer. By having this information, airlines can invest in those areas to improve customer's experience and enhance their brand images.

Finally, during our analysis, we did not identify a notable difference or enough mentions of COVID-19 or the pandemic to conclude that the current situation has changed what determines people sentiment based on tweets for the airline industry. Empirically it certainly had at some point when this situation first started, but at this point and with the data we collected we can conclude that there is a substantial difference.

Limitations

Concluding it has to be said that our approach has certain limitations which need to be taken into consideration evaluating our results and drawing conclusions from them. The biggest one was our datasets since the scope of this project can be expanded to be much more in-depth, analytical than it is, and dive deeper into certain aspects that stood out. More specifically, we worked with a limited amount of training data points from a specific time in the past and while our testing was substantially larger it contained data from a very small time frame.

Another important one was the lack of sufficient processing power for more extensive testing and tuning of our models. That’s especially true when it comes to neural networks. We had to work within a specific time frame and limit the amount of data processed to a manageable level according to our budget on that front.

Last but not least, the project is only analyzing Twitter data. In other words, Twitter users who post airline-related tweets cannot represent all passengers around the world. Also, not all passengers tweet their experience. The research itself can be biased. We strongly recommend that the usage of our research should be considered with other criteria and data sources, which are valuable to the industry and the business. However, even with these limitations, the great potential of text analytics and the value of the project are undoubted.

References

Himanshu. “Sentiment Analysis - TorchText.” Medium, Medium, 28 Apr. 2018, medium.com/@sonicboom8/sentiment-analysis-torchtext-55fb57b1fab8.

Howard, Jeremy, and Sebastian Ruder. “Universal Language Model Fine-Tuning for Text Classification.” ArXiv.org, 23 May 2018, arxiv.org/abs/1801.06146.

Ruder, Sebastian, and Julian Eisenschlos. “Fast.ai NLP.” Fast.ai NLP · Practical NLP, Sept. 2019, nlp.fast.ai/.

Trevett, Ben, “Multi-class Sentiment Analysis”. Pytorch Sentiment Analysis, Github 10.Apr. 2019,https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/5%20-%20Multi-class%20Sentiment%20Analysis.ipynb

Kaggle dataset originally from originally came from Crowdflower's Data for Everyone library downloaded from: https://www.kaggle.com/crowdflower/twitter-airline-sentiment