Airline Review analysis based on text classification using BERT and LSTM

Akshat Savaliya

April 2021

1 Introduction

Text Classification and sentiment analysis is a very common and popular machine learning problem. It is used in a lot of activities like product predictions, movie recommendations, and several others. Text classification Is an important task and its application span a wide range of activities such as topic classification, spam detections and sentiment classification. Recent studies showed that models based on neural networks can outperform conventional models (e.g. Bayes model) on text classification tasks. Typical neural network-based text classification models are based on words. They typically use words in the target documents as inputs, map words into continuous vectors (embeddings), and capture the semantics in documents by using compositional functions over word embeddings such as averaging or summation of word embeddings, recurrent neural networks (RNN).

The purpose of this project is to use two Neural Network (NN) based models on Text classification problem using Airline Dataset found on Kaggle. The final aim is to provide some analysis on airlines like overall problems faced by customers in particular airline by word cloud so it can provide a good summary about airline reviews given by past customers.

With the rise of deep learning-based models and more and more public available datasets, we have seen significant progress in Text Classification task. In this project we will look at two different approaches to solve text classification problem. One approach is of using pre-trained BERT model and other one is LSTM (Long Sort Term Memory) which is based on RNN model. The structure of this report is as follows. We will first begin by introducing the dataset and do some exploratory analysis on Dataset. We then introduce the model and show some show the experiments by loss graph and different analysis matrices. Finally, we can look at the learning and future work.

2 Dataset

a) Two datasets are used for this project. One is Twitter US Airline Sentiment which is found from Kaggle. The data is scraped from twitter for Feb 2015. It consists of tweets given by users about different flights. The data has different columns like tweet$_id, airline_sentiment, airlinename, username, retweetcountetc.$

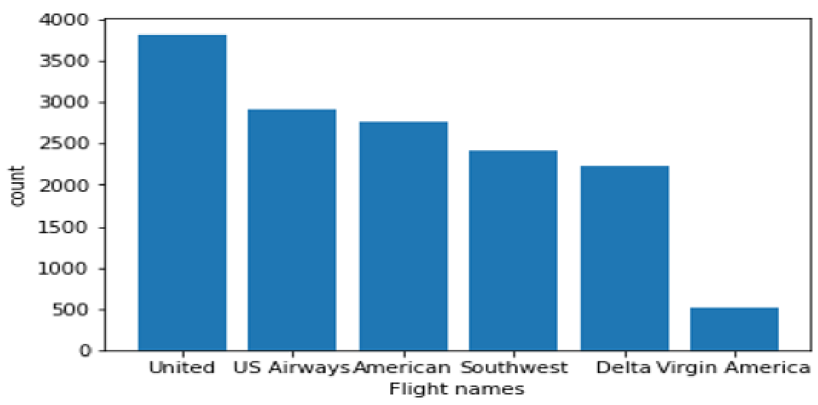


Figure 1: distribution of reviews

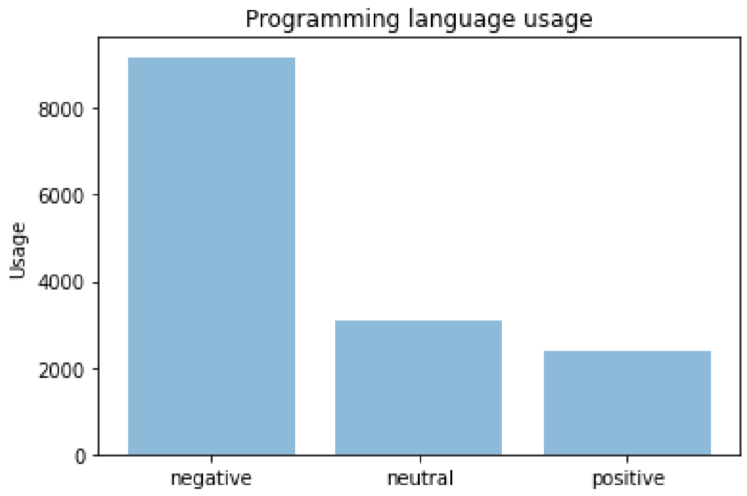


Figure 2: distribution of reviews according to classes

The dataset consists of 9178 negative reviews, 3099 neutral reviews and 2363 positive reviews. In total 14640 reviews. The major issue with this dataset is the classes are not divided equally they are lenient toward negative reviews and this affects the models while training. We will discuss some ways we can overcome this problem to better train the models in upcoming sections. Both BERT and LSTM models are trained on trained and validated on this dataset. We will compare these two models and their outputs also. The other problem with this dataset is, it is small when it comes to Neural Network NLP problems.

b) The other dataset is used for testing purpose. It is also found from Kaggle. It consists of airline data like flight name, passenger name, flight take off destination, flight landing destination, customer review, date etc. This dataset is not labeled like the dataset we are going to use to train both of our models. So, this is the best dataset to measure our models and to see models have learnt so far in training. It consists of total 6919 reviews on 4 airlines: United Airlines, Southwest airlines, American Airlines and virgin airways. Example of review: First they changed my flight time in London so my layover was 5 hours, then I get to Chicago and they wait until the last minute to change the gate which was a 15 minutes walt from original gate. They changed my flight time there to another 5 hour layover then the flight was delayed again to leave at 9 pm. Because of AA my travel time was 31 hours in length with no apologies. No one wants to help or answer any questions they don't care about the passenger I will never fly AA again.

3 BERT (Bidirectional Encoder Representation from Transformers)

The field of NLP has been transformed with the recent advent of pre-trained models. These language models are trained on large corpus data and then find tuned for different tasks. BERT (Bidirectional Encoder Representation from Transformers) is one such model introduced by google in 2018 that uses stack transformer-based architecture and bidirectional training.

BERT's model architecture is a multi-layer bidirectional Transformer with only the encoder part. To make BERT handle a variety of tasks, the input representation can unambiguously represent both a single sentence and a pair of sentences in one token sequence. To represent a word/token it uses Word Piece embeddings (sub-word segmentation algorithm used in NLP). The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. Sentence differentiation is done in two ways firstly by separating them with a special token ([SEP]). Additionally, a learned embedding is added to every token indicating whether it belongs to sentence A or sentence B For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings It is trained on a large corpus with two objectives Masked language modeling (MSM) and Next Sentence Prediction (NSP). Although BERT can give state-of-art results for different NLP tasks still due to large number of parameters it takes a long time to train.

This pre-trained BERT model is trained on large corpus (google massive) that it can be used for many NLP problems. One of the old and popular one is Text Classification. The difference between other base non-NN models for text classification and BERT is, BERT is pre trained on large text data so that it can pick up the semantic and syntactic meaning of the words more accurately than other NN networks.

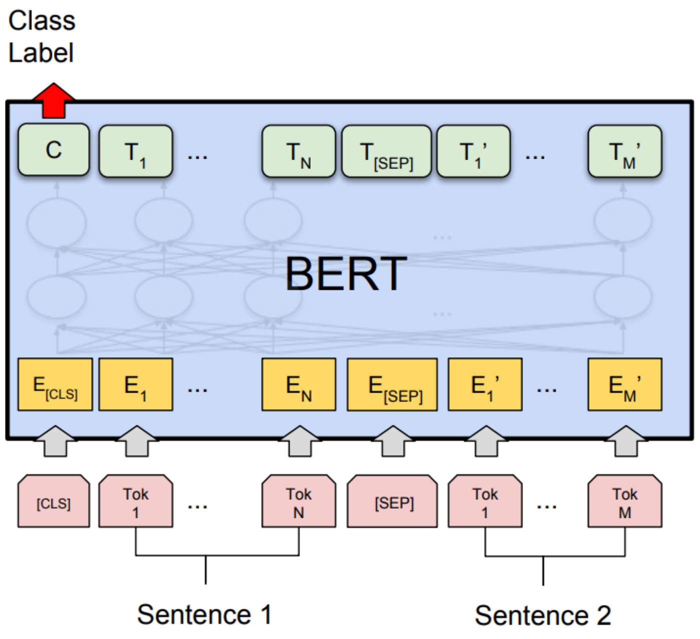


Figure 3: BERT Architecture

4 Airline Review Classification using BERT

a) Airlines data has reviews on different flights in a text format. To perform any NLP task the basic start is to clean the data to get rid of unnecessary text symbols and words stop words which are not going to help train the model best. So, the first step is to process the data and create a dataframe that can be feed into our model. Here is the link to the code for BERT model (to refer the code).https://github.com/AkshatSavaliya/NLP_Final_Project. The data is then divided into 70: 30 ratio for training and validation purpose. Important thing to consider here is when we separate data into training and validation, we have to maintain the ratio of different classes. Here we have 3 classes: Negative, Neutral and Positive. The count ratio is maintained while splitting the data.

   The major issue with pre trained models is if training data is small then it can easily convert into overfitting problem. I had to face the same issue because I have nearly 15k reviews which we can consider it as small dataset when we are training BERT model. I am using BERT (base uncased) model which has 12-layer, 768-hidden, 12-heads, 110M parameters. Initially while training the validation loss was increasing after 4-5 epochs. The concept that I have used to overcome this issue is called transfer learning.

b) **Transfer Learning:** We can understand transfer leaning by simple analogy of teacher and student. A teacher has years of experience in the topic. With all this accumulative information, the lectures students get is a concise and brief overview of the topic. So it can be seen as “transfer” of information from learnt to novice. Keeping this analogy in mind we can think that a pre trained model (Neural Network) is trained on large corpus. This model gain knowledge from data which can be seen as the network "weights". These weights can be extracted and transferred to any other network. Instead of creating network from scratch we can transfer these weight features.

There are multiple ways to find tune the pre-trained model. Like

(a) **Feature extraction** – We can use a pre-trained model as a feature extraction mechanism. What we can do is that we can remove the output layer (the one which gives the probabilities for being in each of the 1000 classes) and then use the entire network as a fixed feature extractor for the new data set.

(b) **Use the Architecture of the pre-trained model** - What we can do is that we use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.

(c) **Train some layers while freeze others** - Another way to use a pre-trained model is to train is partially. What we can do is we keep the weights of initial layers of the model frozen while we retrain only the higher layers. We can try and test as to how many layers to be frozen and how many to be trained.

I used the last one which was the best-case scenario in my case. I froze some layers. The ratio of frozen and unfrozen layers is 80:20. Using this technique I was able to overcome the problem of overfitting. You can access the code link here. https://github.com/AkshatSavaliya/NLP_Final_Project The resulting graph for loss to epochs is given as below. After 20 epochs we can see that both training and validation loss curve is getting flat. One thing we can notice here is, we are getting the validation loss lower than training loss. The reason behind this is, we are using this transfer learning technique only while training the model and also other reason is the size of the validation data which in our case is small.

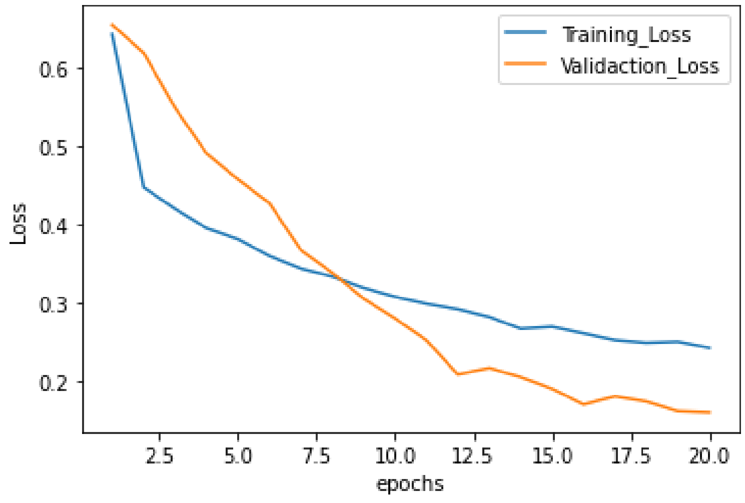


Figure 4: BERT Loss vs epoch

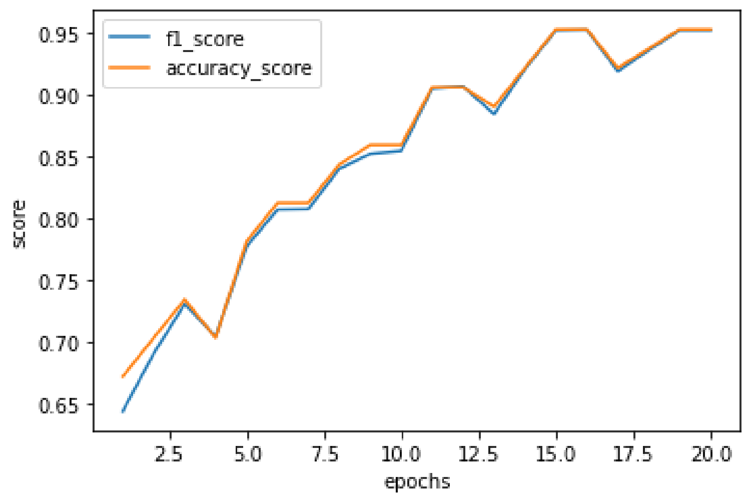


Figure 5: BERT Loss vs epoch

5 LSTM (Long Sort Term Memory) Model

LSTM is base on deep neural network RNN (Recurrent Neural Network). We know that neural network uses an algorithm called Backpropagation to update the weights of the network. So what Backpropagation does is it first calculates the gradients from the error using the chain rule, then in updates the weights (Gradient Descent). When we build an architecture with large number of hidden layer (Deep Neural Network) the model is likely to encounter update weight problem called vanishing gradient descent.

The LSTM contains special units called memory blocks in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of information. Each memory block in the original architecture contained an input gate and an output gate. The input gate controls the flow of input activations into the memory cell. The output gate controls the output flow of cell activations into the rest of the network. Later, the forget gate was added to the memory block. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cell's memory.

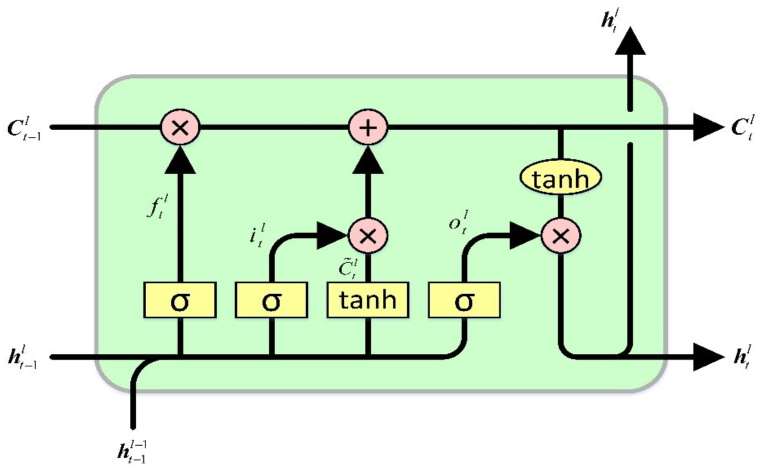


Figure 6: LSTM Cell

1. **Preprocessing and Embedding:** First, all the reviews are collected and processed to remove unnecessary symbols and stop words. In bag-of-words model, it's very high dimensional, and there's a lack of contextual relations between words. To better represent the limited content in short texts, we use Word2Vec word embedding model to learn the contextual relations among words in training data. Also, the fixed number of dimensions in word embedding model can facilitate more efficient computations. There are two general models in Word2Vec, Continuous Bag-of-Words (CBOW) and Skip-gram. Since much better performance for skip-gram model in semantic analysis can be obtained, we use word vectors trained via Word2Vec Skip-gram model as the inputs to the following stage of classification.

2. **Challenges:** • While training the LSTM model after some epochs (7-8) validation loss was increasing. There could be many reasons like overfitting. But to overcome this problem I used drop out technique in LSTM layers. At each layer of LSTM I have put dropouts. The dropouts are increased near the output layer. Dropout is a regularization method where we exclude input and recurrent connections to LSTM units from activation and weight updates while training a network. This has the effect of reducing overfitting and improving model performance. The output after using the dropouts is given below.

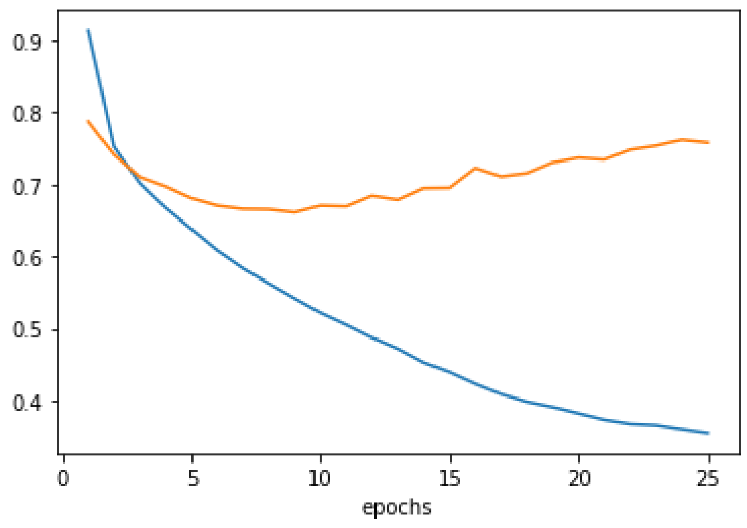


Figure 7: initial loss curves

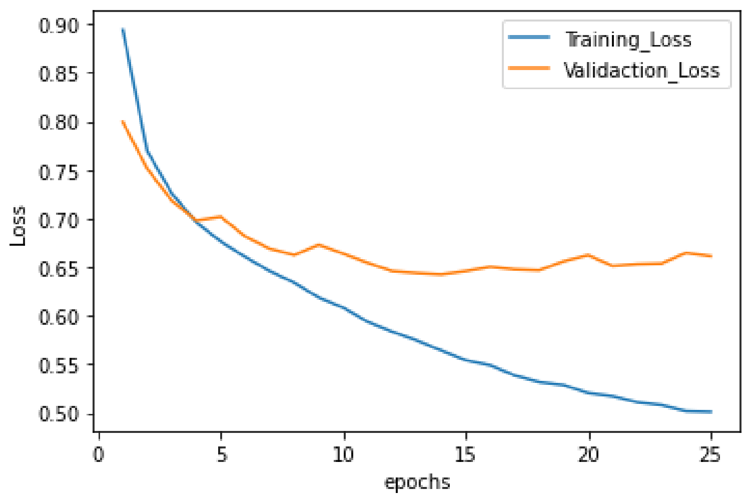


Figure 8: Loss curve after modification in Model

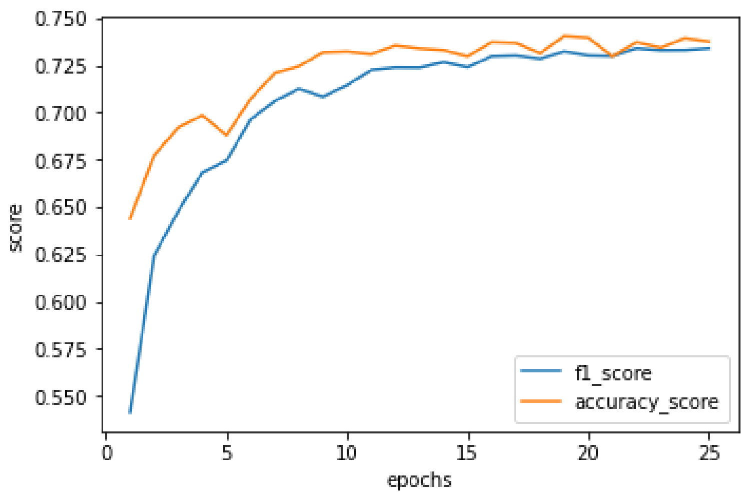
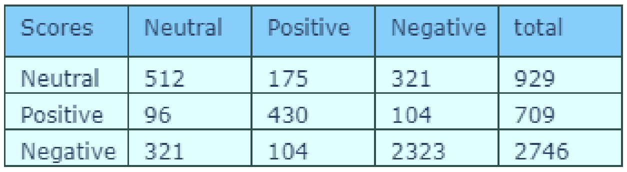


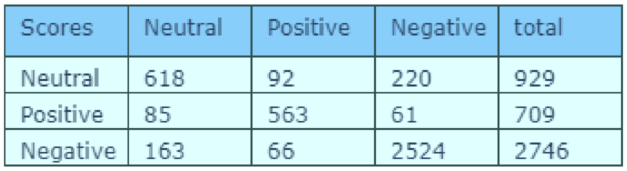
Figure 9: f1-accuracy graph

3. **Result:** LSTM is getting 75 percent accuracy and BERT is getting 84.5 percent accuracy for classification in validation data. So, we can see the difference that BERT is performing way better then LSTM. The reason we can conclude is, BERT is pre trained model which has been trained on large corpus data. Below is the confusion matrix for BERT and LSTM.



Scores	Neutral	Positive	Negative	total
Neutral	512	175	321	929
Positive	96	430	104	709
Negative	321	104	2323	2746

Figure 10: LSTM Confusion Matrix



Scores	Neutral	Positive	Negative	total
Neutral	618	92	220	929
Positive	85	563	61	709
Negative	163	66	2524	2746

Figure 11: BERT Confusion Matrix

6 Evaluation on 2nd Dataset:

The 2nd dataset, as described above consist of nearly 7k reviews given by users to 4 different flights. It does not consist of labels so to evaluate the result I have manually labeled 500 reviews top predict the accuracy of the classification on the dataset. The purpose of this project is to give some analysis on new flight review data by training BERT and LSTM model. I have trained both the models at their full extent and predicted the reviews in positive, negative and neutral classes. I have also created word cloud to see what king of words users are using to give feedback. Like in negative reviews words like bad flight, no apologies, long flight can be seen. That way we can understand and provide some analysis on different flights and their issues to our customers. Here are the results I have got for dataset using BERT model. I have manually given labels to 500 reviews to positive and negative. BERT is able to predict 85.7 percent of the reviews correctly. The reason we are getting higher prediction accuracy is because we have long reviews in testing compare to training part.

American Airline: Below we can see some graphs for American Airline. People faced some issues like flight delay, rude behavior, issues regarding luggage, boarding gate changed issue which can be seen in word cloud. In the date graph we can see that over the time negative reviews are decreasing.

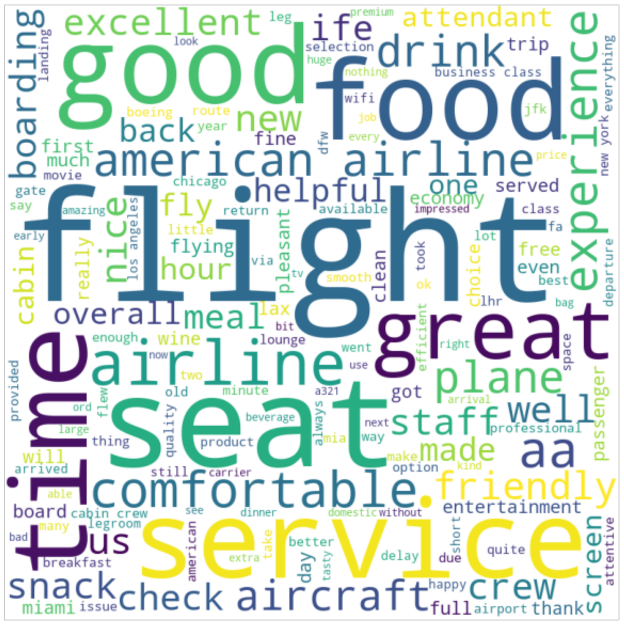


Figure 12: Positive wordcloud for American Airline (BERT)

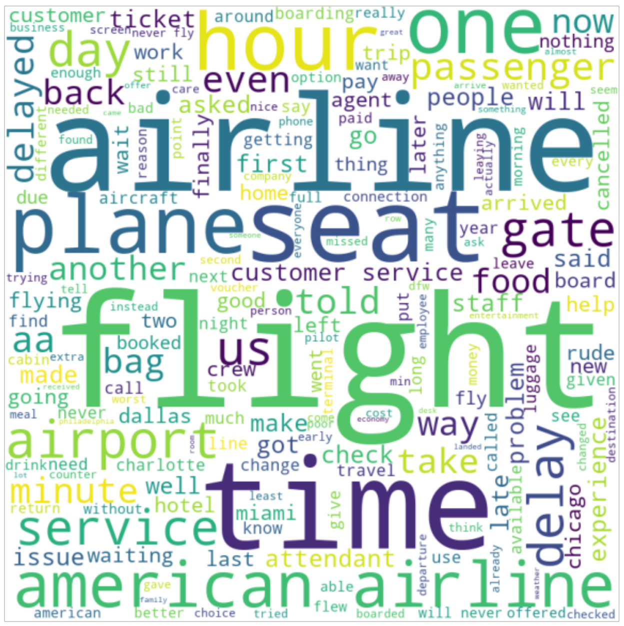


Figure 13: Negative wordcloud for American Airline (BERT)

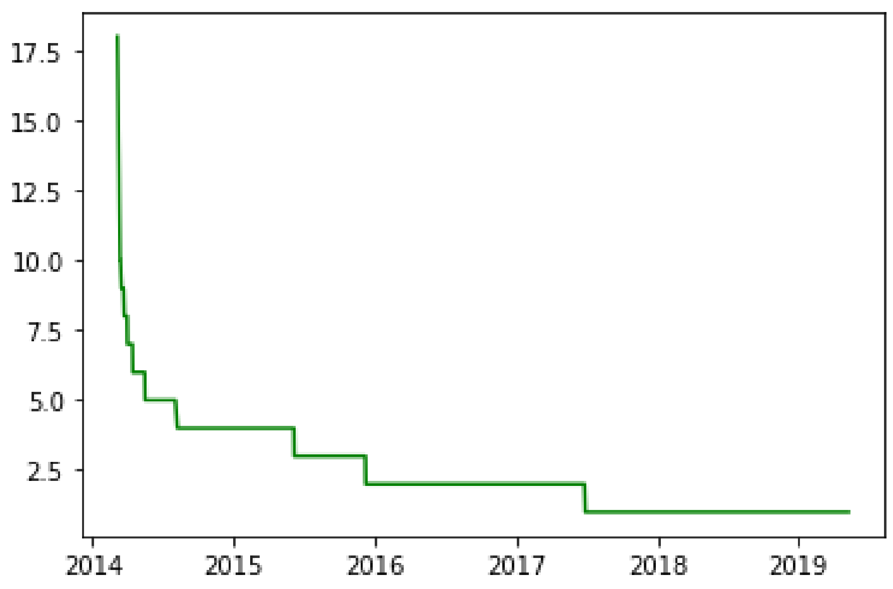


Figure 14: Negative review over time for American Airline

7 Conclusion:

In this project, we implemented BERT and LSTM on Twitter US Airline Sentiment. From our experiment we can observe that Pre trained BERT model outperforms LSTM model. From confusion matrix we can see that BERT can predict 84.5 percent correctly compare to LSTM which has 75 percent prediction accuracy. Our empirical results come in line with the expected results. The pretrained language model outperformed the end to end trained model by a huge margin, achieving better results in the first epoch itself. These results show the benefit these large models enjoy due to many parameters and the large dataset they are trained on. Another interesting observation we observe is that the larger model almost plateaued after a couple of epochs and no major improvement was observed in later epochs.

8 Learning:

This project allowed us to explore the domain of Text Classification using Neural Nets, an important task in Natural Language Processing. Some of the interesting learning I observed are as follows:

1. Data Processing and modeling the data in a shape that we can feed into model is a much more challenging task than creating model itself.
2. The major advancement in NLP in terms of Pre -trained language models. Discounting the compute and training time these models outperformed the other models. It also made us aware of the huge impact libraries like huggingface have in making these models accessible to the public.

3. The importance of hyperparameter tuning, especially when training models. Batch size and learning rate played a huge role in determining the ease of training these models.

9 Future Work:

Due to time and resource constraints, I was unable to perform some experiments and would be part of our future work.It would be an interesting exercise to see how much improvement can be achieved by ensembling the results from all the models like CNN model. Another interesting experiment would be to take the trained/finetuned models and see how well the results transfer to other datasets like IMDB reviews or Hate/Positive tweets on twitter.

10 Github Code link:

Code: https://github.com/AkshatSavaliya/NLP_Final_Project

11 Citations

1. **Github code link:** https://github.com/AkshatSavaliya/NLP_Final_Project
2. **Dataset1:** https://www.kaggle.com/crowdflower/twitter-airline-sentiment
3. **Dataset2:** https://www.kaggle.com/efehandanisman/skytrax-airline-reviews
4. **BERT Pre-training of Deep Bidirectional Transformers for Language Understanding:** https://arxiv.org/pdf/1810.04805.pdf
5. **LSTM:** https://dl.acm.org/doi/10.1162/neco.1997.9.8.1735
6. https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/
7. https://algotech.netlify.app/blog/text-lstm/
8. https://towardsdatascience.com/lstm-vs-bert-a-step-by-step-guide-for-tweet-sentiment-analysis-ced697948c47
9. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf