Table of Contents

Abstract

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. This project of analysing sentiments of tweets comes under the domain of "Pattern Classification" and "Data Mining". Both of these terms are very closely related and intertwined, and they can be formally defined as the process of discovering "useful" patterns in large set of data, either automatic (unsupervised) or semiautomatic (supervised). The project would heavily rely on techniques of "Natural Language Processing" in extracting significant patterns and features from the large data set of tweets and on "Machine Learning" techniques for accurately classifying individual unlabelled data samples (tweets) according to whichever pattern model best describes them.

Introduction

Domain Introduction

So the topic was to find the sentiment label from a sentence in order to analyze the emotions of customers towards a brand/company. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users - out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an airline tweet stream.

The features that can be used for modelling patterns and classification can be divided into two main groups: formal language based and informal blogging based. Language based features are those that deal with formal linguistics and include prior sentiment polarity of individual words and phrases, and parts of speech tagging of the sentence. Prior sentiment polarity means that some words and phrases have a natural innate tendency for expressing particular and specific sentiments in general. For example the word "excellent" has a strong positive connotation while the word "evil" possesses a strong negative connotation. So whenever a word with positive connotation is used in a sentence, chances are that the entire sentence would be expressing a positive sentiment. Parts of Speech tagging, on the other hand, is a syntactical approach to the problem. It means to automatically identify which part of speech each individual word of a sentence belongs to: noun, pronoun, adverb, adjective, verb, interjection, etc. Patterns can be extracted from analysing the frequency distribution of these parts of speech (either individually or collectively with some other part of speech) in a particular class of labelled tweets (Positive, Negative, Neutral). Twitter based features are more informal and relate with how people express themselves on online social platforms and compress their sentiments in the limited space of 140 characters offered by twitter. They include twitter hashtags, retweets, word capitalization, word lengthening, question marks, and presence of URL in tweets, exclamation marks, emoticons and internet shorthand/slangs. Classification techniques can also be divided into a two categories: Supervised vs. unsupervised and non-adaptive vs. adaptive/reinforcement techniques. Supervised approach is when we have pre-labelled data samples available and we use them to train our classifier. Training the classifier means to use the pre-labelled to extract features that best model the patterns and differences between each of the individual classes, and then classifying an unlabelled data sample according to whichever pattern best describes it. Unsupervised classification is when we do not have any labelled data for training. There are several metrics proposed for computing and comparing the results of our experiments. Some of the most popular metrics include: Precision, Recall, Accuracy, F1 Score, A typical confusion table for our problem is given below along with illustration of how to compute our required metric.

Related Work

The bag-of-words model is one of the most widely used feature model for almost all text classification tasks due to its simplicity coupled with good performance. The model represents the text to be classified as a bag or collection of individual words with no link or dependence of one word with the other, i.e. it completely disregards grammar and order of words within the text. The simplest way to incorporate this model in our classifier is by using unigrams as features. Generally speaking n-grams is a contiguous sequence of "n" words in our text, which is completely independent of any other words or grams in the text. So unigrams is just a collection of individual words in the text to be classified, and we assume that the probability of occurrence of one word will not be affected by the presence or absence of any other word in the text.

The process of designing a functional classifier for sentiment analysis can be broken down into five basic categories. They are as follows:

I. Data Acquisition
II. Data Cleaning
III. Data Visualisation
IV. Classification
V. Web Interface

Features Description

There were 8 Columns and 14640 Rows in the provided dataset.
The following columns had the following text:

- *tweet_id* – This was the primary key for our dataset as it was unique was every data entry
- *airline_sentiment* – This was out target label class column containing 3 classes (Positive, Negative, Neutral)
- *name* – This column contains the name of the tweeters and it has 7701 unique values.
- *text* – This was out main data column which will be used to train out classifier and extract features from. The target label class is dependent on this column
- *tweet_coord* – This column was ought to contain the origin co-ordinates of the tweeters however it has 93% data missing

- *tweet_created* – This column contained the time and date of the tweet post
- *tweet_location* – This column contained country or approx. location of a tweet origin
- *user_timezone* – This column contained the timezone of the user who did the tweet.

Feature Engineering

Data Cleaning

Since this is a lot of information we only filter out the information that we need and discard the rest so we need to perform the data formatting techniques or data cleaning.

- Tokenization: It is the process of breaking a stream of text up into words, symbols and other meaningful elements called "tokens". Tokens can be separated by whitespace characters and/or punctuation characters. It is done so that we can look at tokens as individual components that make up a tweet.
- URL's and user references (identified by tokens "http" and "@") are removed if we are interested in only analysing the text of the tweet.
- Punctuation marks and digits/numerals may be removed if for example we wish to compare the tweet to a list of English words.
- Lowercase Conversion: Tweet may be normalized by converting it to lowercase which makes it's comparison with an English dictionary easier.
- Stemming: It is the text normalizing process of reducing a derived word to its root or stem. For example a stemmer would reduce the phrases "stemmer", "stemmed", "stemming" to the root word "stem". Advantage of stemming is that it makes comparison between words simpler, as we do not need to deal with complex grammatical transformations of the word. In our case we employed the algorithm of "porter stemming" on both the tweets and the dictionary, whenever there was a need of comparison.
- Stop-words removal: Stop words are class of some extremely common words which hold no additional information when used in a text and are thus claimed to be useless. Examples include "a", "an", "the", "he", "she", "by", "on", etc. It is sometimes convenient to remove these words because they hold no additional information since they are used almost equally in all classes of text, for example when computing prior-sentiment-polarity of words in a tweet according to their frequency of occurrence in different classes and using this polarity to calculate the average sentiment of the tweet over the set of words used in that tweet.
- Parts-of-Speech Tagging: POS-Tagging is the process of assigning a tag to each word in the sentence as to which grammatical part of speech that word belongs to, i.e. noun, verb, adjective, adverb, coordinating conjunction etc.

So the data cleaning was performed serially in below given order:

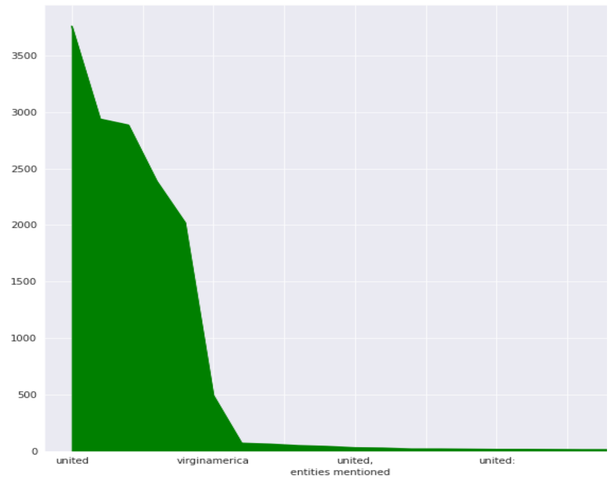
- *tweet_id* and *tweet_coord* column was dropped
- Text column was converted to lowers case
- Removal was HTML and URL's from text
- @Entities were separated from text and allocated to a new column

- #Hashtags were converted to normal text and were also allotted to another column for future analysis of trends
- Decontraction of Text
- Emoticons were converted to equivalent emoticon text as they enhanced the value of text.
- Punctuation and StopWords Removal
- Lemmatization and tokenization

Exploratory Data Analysis

Data Analysis is a crucial part and it include the visualization which is an essential part in model building as it describes the relationship between different columns of a dataset in an interactive manner. so that one can easily understand the model and extract the important aspects from the data in order to fabricate them in model even if they don't put a strong co-relation in model in current scenario but do would prove out well beneficial in future.

- **Top 20 tags in comparison to tweets**

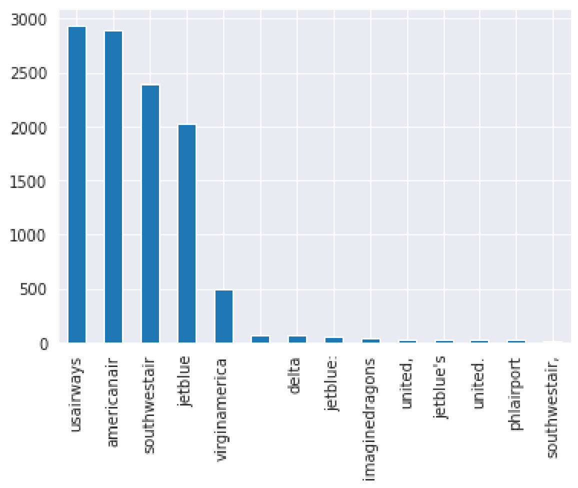


Plot 1

Inference: From this area plot, we have observed that every tag was of US based airlines so it was clear that the dataset we are given is of tweets mentioning Airlines.

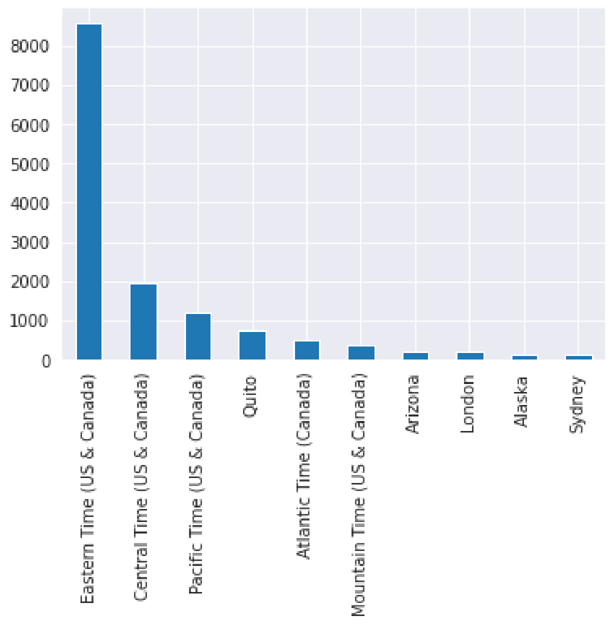
United Airways is the most mentioned tag in the tweets and the second most mentioned tag is Virgin America.

- **Top 15 entities with respect to their occurrence**



Plot 2

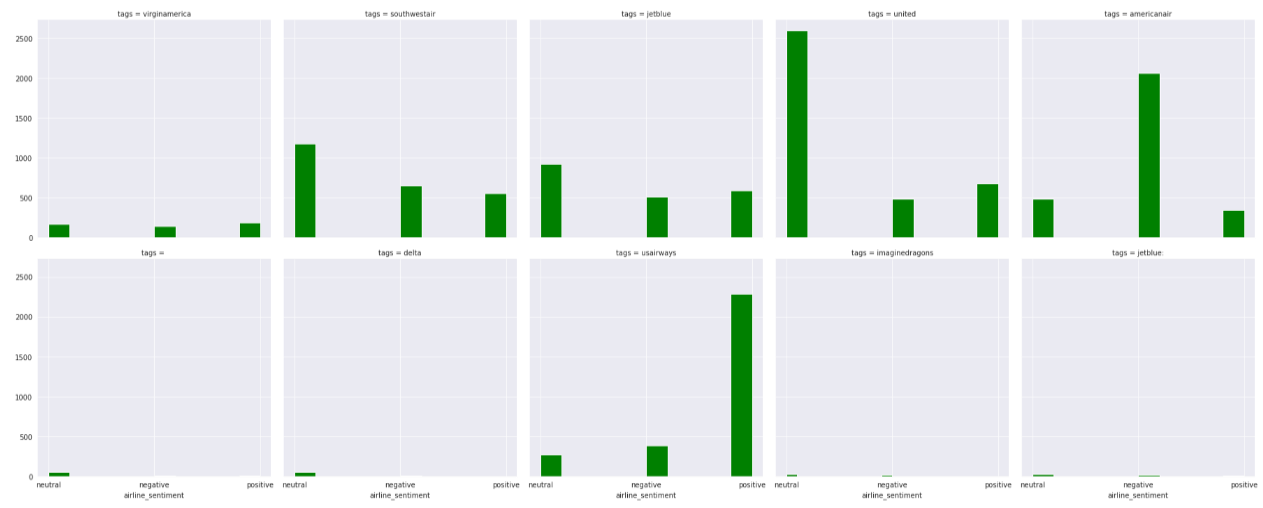
- **Timezone V/S Tweet**



Plot 3

- **Top 10 tags in comparison to *airline_sentiment***

From this visualisation, we can predict about the services of the top 10 mentioned tags. Firstly, we have created a dataframe named as raw_df and extracted the top 10 tags through splitting and then plot a graph for the top 10 extracted tags in comparison to their sentiment(target label) so that one can easily predict about their services.



Plot 4

Inference:

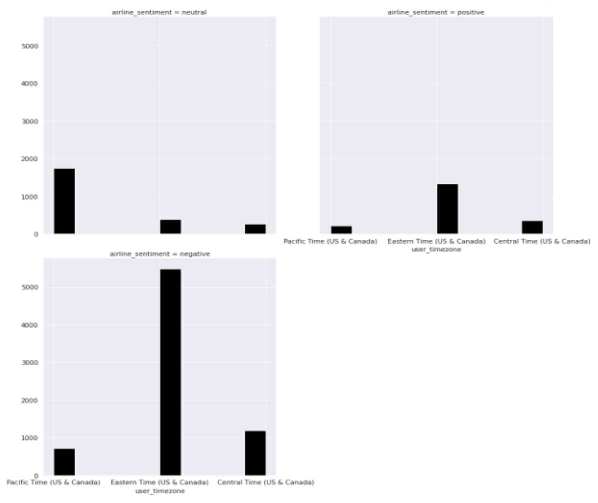
➢ US airways provide better service as it is mostly mentioned in those tweets whose sentiment is positive in nature.

➢ South west air provides satisfied services because it is mostly mentioned in those tweets whose sentiment is neutral.

➢ American air provides worst services as it is mentioned in most of the tweets which are negatively classified.

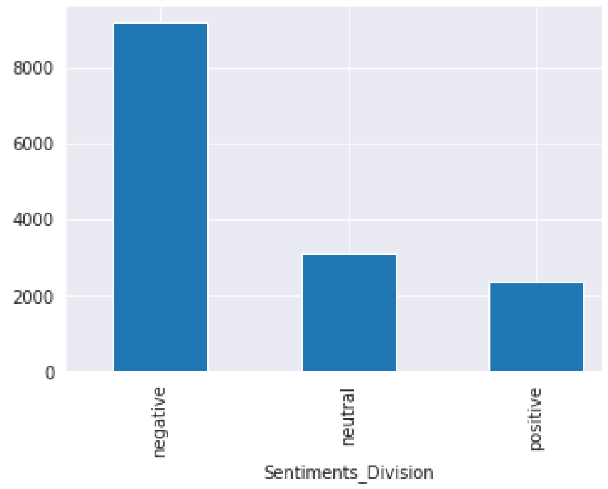
- **Timezone V/S Sentiment**

  Pacific Time - Neutral, Eastern Time - Positive, Eastern Time - Negative.



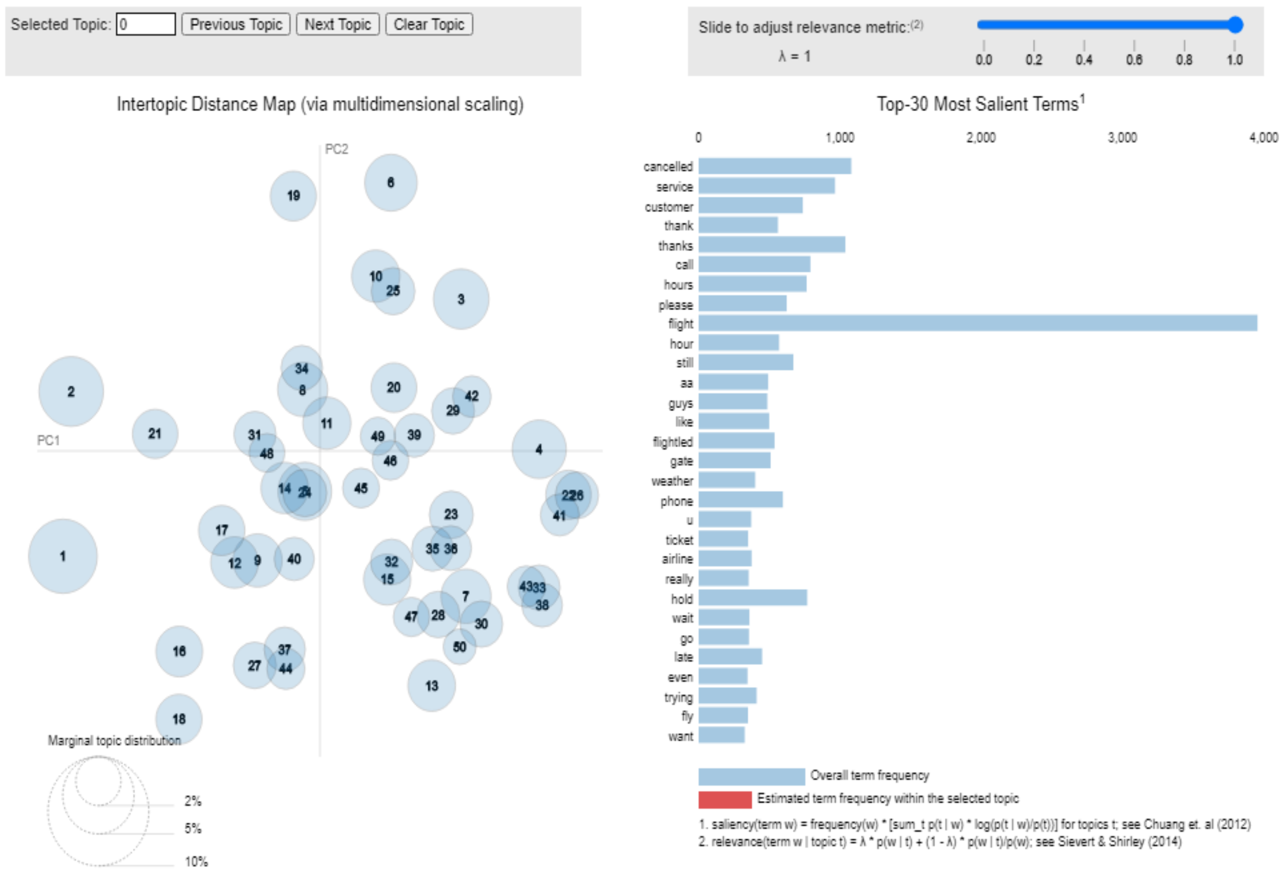
Plot 5

- **Sentiment V/S Tweets**
  70% of Tweets are negative which shows that dataset is imbalanced and it can cause a problem while classification.



Plot 6

- **Top 10 common topics**
  10 common topics with the help of **LDA** (*Latent Dirichlet Allocation*) which is an unsupervised machine learning model that takes document as input and give topics as output by calculating the reoccurrence of the words present in the text and then extract the common topics from the text. After extracting the common topics, ***pyLDAvis*** library which is an interactive way to visualise the topic models was used. What was observed is that [“Cancellation”, “Flight”] are the most common topic among all the tweets which pretty much adds up as the dataset is of flight.



Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
Marginal topic distribution
2%
5%
10%
Top-30 Most Salient Terms1
0 1,000 2,000 3,000 4,000
cancelled
service
customer
thank
thanks
call
hours
please
flight
hour
still
aa
guys
like
flightled
gate
weather
phone
u
ticket
airline
really
hold
wait
go
late
even
trying
fly
want
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Plot 7

- **Word Clouds**
  We have generated four different word clouds –
  - **Thank** and **Flight** are the most common words in the whole text column.

flight
thank
help
time
plane
bag
customer
service
hour
cancelled
flightled

Word Cloud 1

  - Most Frequent Word in Positive Labelled Text – **Thank**

thank
flight
great
awesome
customer
service
today
got
love
good
best
time

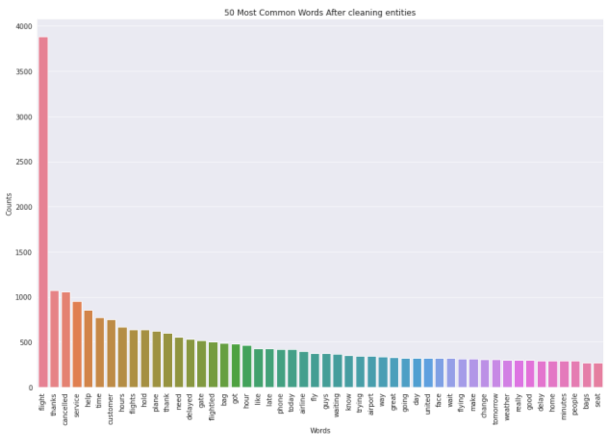
Word Cloud 2

  - Most Frequent Word in Negative Labelled Text - **Flight**

flight
customer
service
plane
hour
help
time
bag
cancelled
flightled

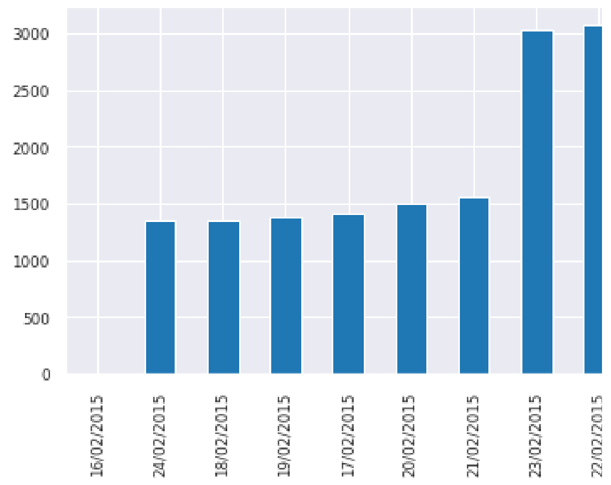
Word Cloud 3

- **50 most common words**
  Flight, thanks, cancelled, service, help, time, customer and others.



Plot 8

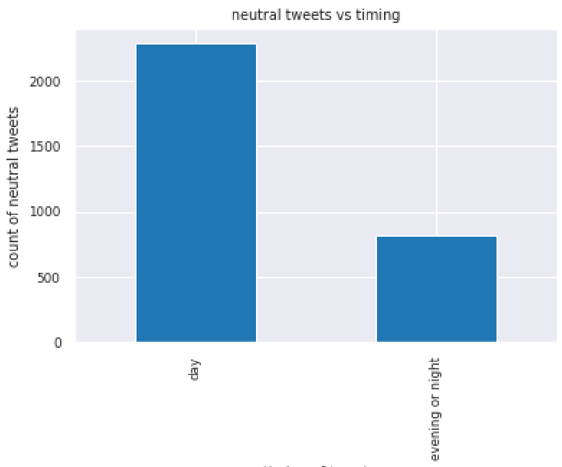
- **Tweets V/S Time distribution**
  In the given plot, we have clearly seen that most of the tweets are created on 24/02/2015



Plot 9

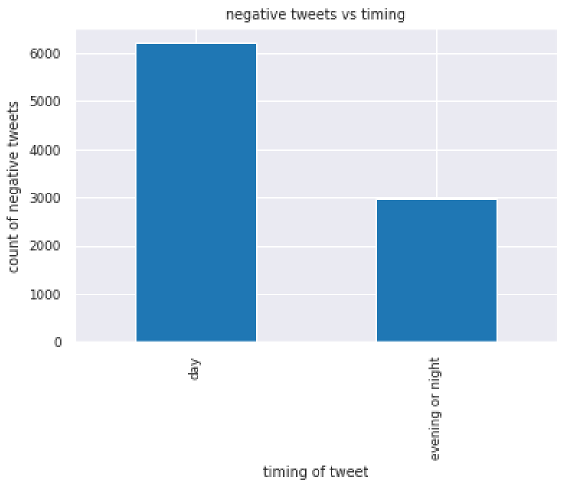
Our data had a tweet time range of 16/02/2015 to 22/02/2015

- **Neutral tweets vs. Timing**



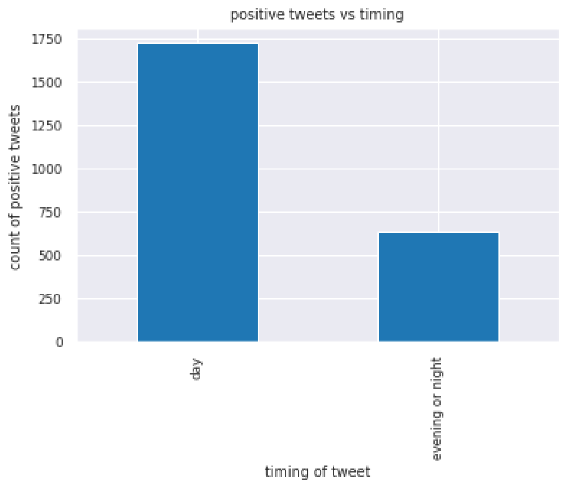
Plot 113

- **Negative tweets vs. Timing**



Plot 124

- **Positive tweets vs. Timing**

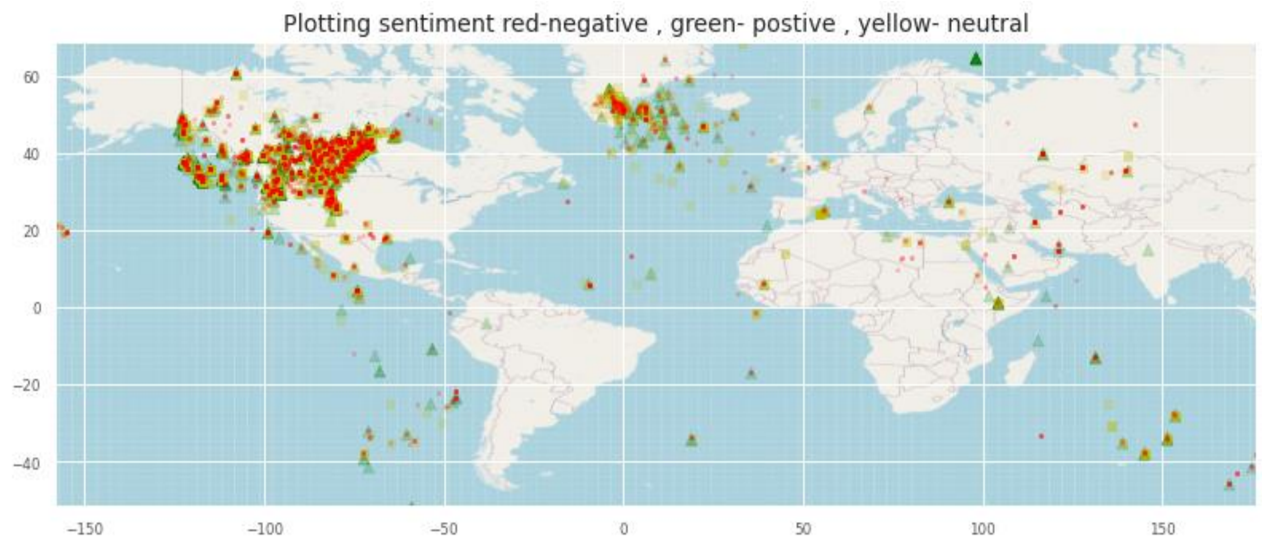


Plot 10

- **Latitude Longitude Map Visualisation**

  The column `*tweet_location*` had approximate locations of tweets origin in it and with help of `*geocode*` the latitude and longitude were calculated from it.

  The latitude and longitudes were pinned down on a world map and the approx. location of tweets was visualized.
  Some pins were in ocean as 18% of data in our dataset were merely false values.


Plotting sentiment red-negative , green- postive , yellow- neutral
60
40
20
0
−20
−40
−150
−100
−50
0
50
100
150
Plot 13

Converting text to features:

TFIDF approach

TF-IDF, short for term frequency–inverse document frequency, is a numeric measure that is use to score the importance of a word in a document based on how often did it appear in that document and a given collection of documents. TF-IDF, understand, tf basically means term frequency. in this we multiply term frequency and inverse document frequency to convert the sentences into vectors now one of the disadvantage of bag of words is that there is no semantic meaning much semantic difference because either we have values like ones or zeros all the values all the features basically have one value. The intuition for this measure is: If a word appears frequently in a document, then it should be important and we should give that word a high score. But if a word appears in too many other documents, it's probably not a unique identifier, therefore we should assign a lower score to that word. The math formula for this measure:

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

We have applied the *tfidfvectorizer* approach to covert text to features using sklearn library.

Word2vec approach

Machine learning and deep learning algorithms cannot accept text directly we need some sort of numerical representation so that the algorithms can process the data and in simple machine learning applications we use *countvectorizer* or *tf-idf* both of which do not preserve any relationship between the words this is where word embeddings come in the map all the words in a language into a vector space of a given dimension so word2vec is a popular method to generate word embeddings this converts words into vectors and with vectors we have multiple operations like add subtract calculate distance and that is how the relationship among the words are preserved so one example of this relationship is a very famous result of word2vec which says the vector of the word kill - word man + the word woman gives you the word vector of the word Queen and this relationship is preserved by word to work just by iterating through a large corpus of text like a Wikipedia or newspaper corpus .

We have applied word2vec embedding using *gensim* library. The trained *word2vec* model on famous Google News model, which was trained on about 100 billion words.

Miscellaneous Data Handling

Named Entity Recognition

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

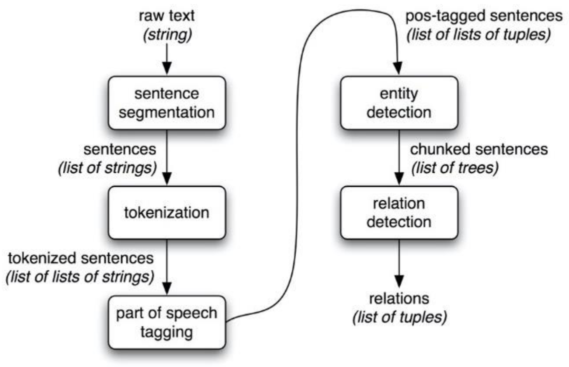


Figure 1

The basic technique that we use for entity detection is chunking which segments and labels multi-token sequences.
In the following figure shows the Segmentation and Labelling at both the Token and Chunk Levels, the smaller boxes in it show the word-level tokenization and part-of-speech tagging, while the large boxes show higher-level chunking. Each of these larger boxes is called a chunk. Like tokenization, which omits whitespace, chunking usually selects a subset of the tokens. Also, like tokenization, the pieces produced by a chunker do not overlap in the source text.

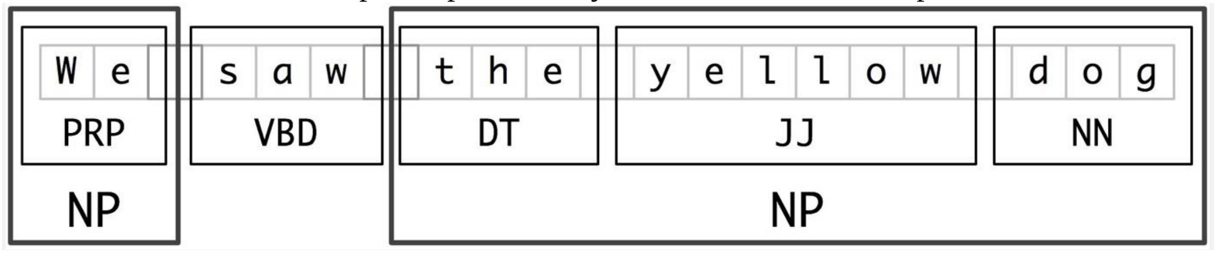


Figure 2

Stanford NER Approach

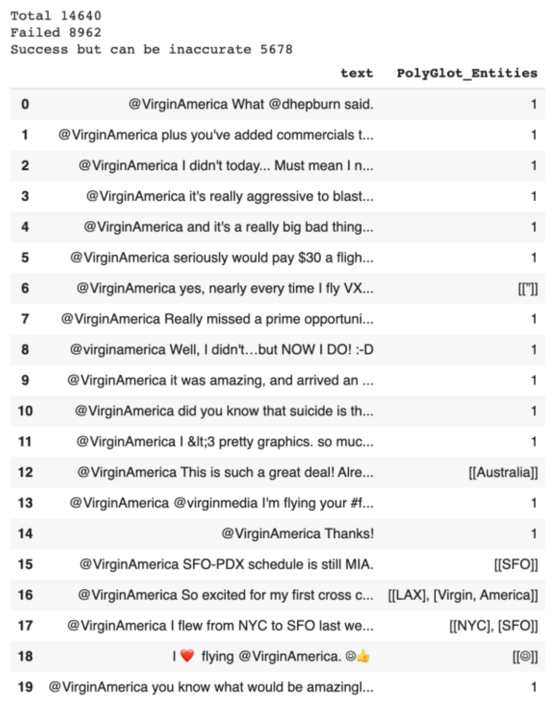
Stanford NER is a Java implementation of a Named Entity Recognizer. Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. Included with the download are good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION), and we also make available on this page various other models for different languages and circumstances, including models trained on just the CoNLL 2003 English training data.

Stanford NER is also known as *CRFClassifier*. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, by training your own models on labelled data, you can actually use this code to build sequence models for NER or any other task.

(CRF models were pioneered by Lafferty, McCallum, and Pereira (2001); see Sutton and McCallum (2006) or Sutton and McCallum (2010) for more comprehensible introductions.)The approach turned out to be not useful to us as the python library for it was obsolete and no more functional.

Polyglot Entity Recognition

Polyglot is a natural language pipeline that supports massive multilingual applications. Polyglot entity extraction works by taking a piece of text (tweets in our case), and annotating (or tagging) the text where it recognizes named entities. It extracts chunks of text as phrases and classifies them into pre-defined categories such as the names of persons, locations, and organizations.



```
Total 14640
Failed 8962
Success but can be inaccurate 5678
```

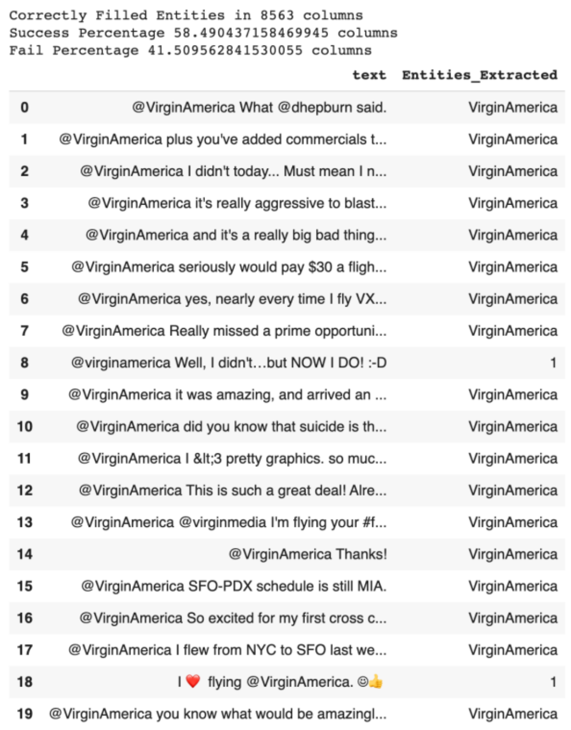
	text	PolyGlot_Entities
0	@VirginAmerica What @dhepburn said.	1
1	@VirginAmerica plus you've added commercials t...	1
2	@VirginAmerica I didn't today... Must mean I n...	1
3	@VirginAmerica it's really aggressive to blast...	1
4	@VirginAmerica and it's a really big bad thing...	1
5	@VirginAmerica seriously would pay $30 a fligh...	1
6	@VirginAmerica yes, nearly every time I fly VX...	[["]]
7	@VirginAmerica Really missed a prime opportuni...	1
8	@virginamerica Well, I didn't…but NOW I DO! :-D	1
9	@VirginAmerica it was amazing, and arrived an ...	1
10	@VirginAmerica did you know that suicide is th...	1
11	@VirginAmerica I <3 pretty graphics. so muc...	1
12	@VirginAmerica This is such a great deal! Alre...	[[Australia]]
13	@VirginAmerica @virginmedia I'm flying your #f...	1
14	@VirginAmerica Thanks!	1
15	@VirginAmerica SFO-PDX schedule is still MIA.	[[SFO]]
16	@VirginAmerica So excited for my first cross c...	[[LAX], [Virgin, America]]
17	@VirginAmerica I flew from NYC to SFO last we...	[[NYC], [SFO]]
18	I ❤️ flying @VirginAmerica. ☺👍	[[☺]]
19	@VirginAmerica you know what would be amazingl...	1

Figure 3

The results were not at all sufficient so polyglot turned out to be of no use in this use case.

NLTK NER

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. A software package for manipulating linguistic data and performing NLP tasks.



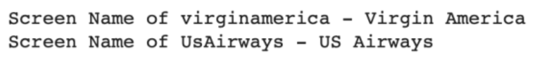
```
Correctly Filled Entities in 8563 columns
Success Percentage 58.490437158469945 columns
Fail Percentage 41.509562841530055 columns
```

	text	Entities_Extracted
0	@VirginAmerica What @dhepburn said.	VirginAmerica
1	@VirginAmerica plus you've added commercials t...	VirginAmerica
2	@VirginAmerica I didn't today... Must mean I n...	VirginAmerica
3	@VirginAmerica it's really aggressive to blast...	VirginAmerica
4	@VirginAmerica and it's a really big bad thing...	VirginAmerica
5	@VirginAmerica seriously would pay $30 a fligh...	VirginAmerica
6	@VirginAmerica yes, nearly every time I fly VX...	VirginAmerica
7	@VirginAmerica Really missed a prime opportuni...	VirginAmerica
8	@virginamerica Well, I didn't…but NOW I DO! :-D	1
9	@VirginAmerica it was amazing, and arrived an ...	VirginAmerica
10	@VirginAmerica did you know that suicide is th...	VirginAmerica
11	@VirginAmerica I <3 pretty graphics. so muc...	VirginAmerica
12	@VirginAmerica This is such a great deal! Alre...	VirginAmerica
13	@VirginAmerica @virginmedia I'm flying your #f...	VirginAmerica
14	@VirginAmerica Thanks!	VirginAmerica
15	@VirginAmerica SFO-PDX schedule is still MIA.	VirginAmerica
16	@VirginAmerica So excited for my first cross c...	VirginAmerica
17	@VirginAmerica I flew from NYC to SFO last we...	VirginAmerica
18	I ❤️ flying @VirginAmerica. ☺👍	1
19	@VirginAmerica you know what would be amazingl...	VirginAmerica

Figure 4

Entity Screen Names

Usage of twitter developer API to fetch screen names of the @Entities in the text in order to see the actual names and then send them to entity recognizer which may improve performance now.



```
Screen Name of virginamerica - Virgin America
Screen Name of UsAirways - US Airways
```

Figure 5

Classification

Pattern classification is the process through which data is divided into different classes according to some common patterns which are found in one class which differ to some degree with the patterns found in the other classes. The ultimate aim of our project is to design a classifier which accurately classifies tweets in the following four sentiment classes: positive, negative.

Following Machine Learning algorithms can be applied to arrive at the best result:

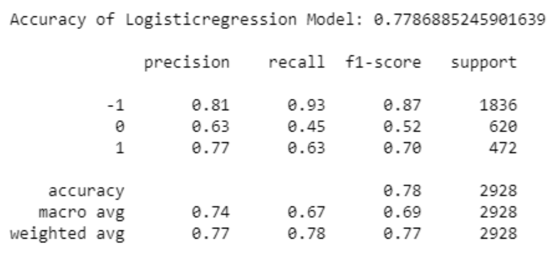
- Support Vector Machine
- Logistic Regression
- Decision Tree
- Random Forest

Modelling Efforts

Logistic regression classifier

Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features (or inputs), X. Here in our case we first trained the model on the TF-IDF vectorizer as an X label and then with word2vec as an X label, and airline sentiment as a target label respectively. sklearn library is used to implement LogesticRegression.

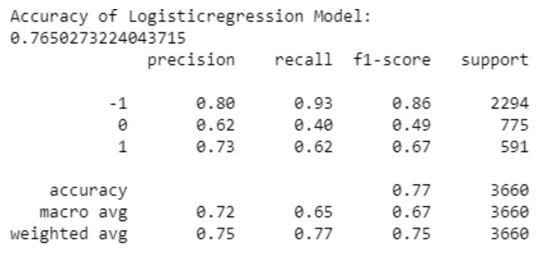
After taking the TFIDF features results are:



```
Accuracy of Logisticregression Model: 0.7786885245901639

              precision    recall  f1-score   support

          -1       0.81      0.93      0.87      1836
           0       0.63      0.45      0.52       620
           1       0.77      0.63      0.70       472

    accuracy                           0.78      2928
   macro avg       0.74      0.67      0.69      2928
weighted avg       0.77      0.78      0.77      2928
```

Result 1

After taking the word2vec vectorizer features results are:



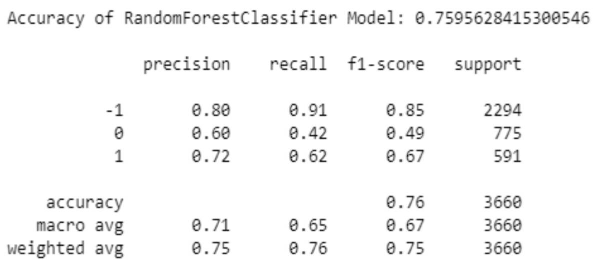
```
Accuracy of Logisticregression Model:
0.7650273224043715
              precision    recall  f1-score   support

          -1       0.80      0.93      0.86      2294
           0       0.62      0.40      0.49       775
           1       0.73      0.62      0.67       591

    accuracy                           0.77      3660
   macro avg       0.72      0.65      0.67      3660
weighted avg       0.75      0.77      0.75      3660
```

Result 2

Random Forest Classifier

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object. We first trained the model on the TF-IDF vectorizer as an X label and then with word2vec as an X label, and airline sentiment as a target label or Y label respectively. We are using sklearn library to implement machine learning module RandomForestClassifier.

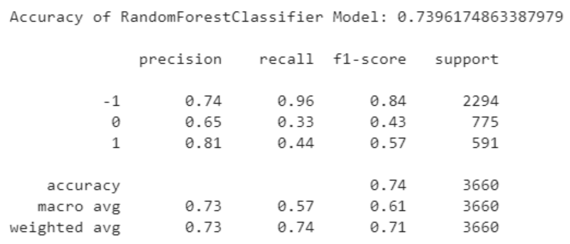
After taking the TFIDF features results are:



```
Accuracy of RandomForestClassifier Model: 0.7595628415300546

              precision    recall  f1-score   support

          -1       0.80      0.91      0.85      2294
           0       0.60      0.42      0.49       775
           1       0.72      0.62      0.67       591

    accuracy                           0.76      3660
   macro avg       0.71      0.65      0.67      3660
weighted avg       0.75      0.76      0.75      3660
```

Result 3

After taking the word2vec vectorizer features results are:



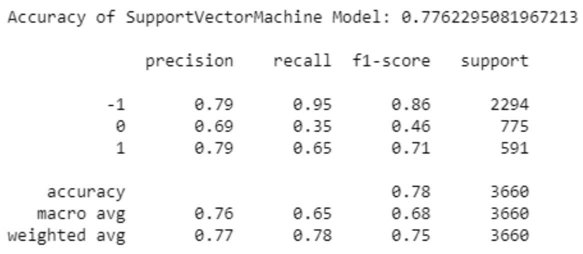
```
Accuracy of RandomForestClassifier Model: 0.7396174863387979

              precision    recall  f1-score   support

          -1       0.74      0.96      0.84      2294
           0       0.65      0.33      0.43       775
           1       0.81      0.44      0.57       591

    accuracy                           0.74      3660
   macro avg       0.73      0.57      0.61      3660
weighted avg       0.73      0.74      0.71      3660
```

Result 4

Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. . We are using sklearn library to implement machine learning module Support Vector machine. Similarly for SVM first trained the model on the TF-IDF vectorizer as an X label and then with word2vec as an X label, and airline sentiment as a target label or Y label respectively.

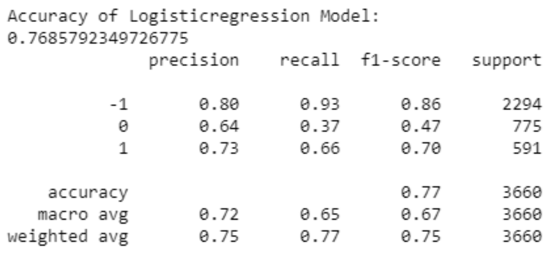
After taking the TFIDF features results are:



```
Accuracy of SupportVectorMachine Model: 0.7762295081967213

              precision    recall  f1-score   support

          -1       0.79      0.95      0.86      2294
           0       0.69      0.35      0.46       775
           1       0.79      0.65      0.71       591

    accuracy                           0.78      3660
   macro avg       0.76      0.65      0.68      3660
weighted avg       0.77      0.78      0.75      3660
```

Result 5

After taking the word2vec vectorizer features results are:



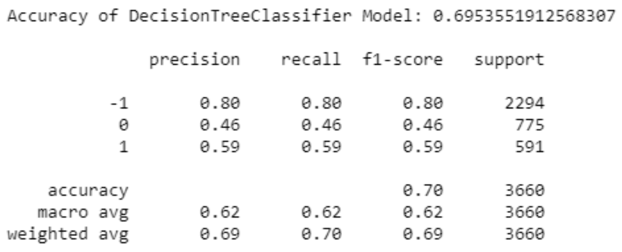
```
Accuracy of Logisticregression Model:
0.7685792349726775
              precision    recall  f1-score   support

          -1       0.80      0.93      0.86      2294
           0       0.64      0.37      0.47       775
           1       0.73      0.66      0.70       591

    accuracy                           0.77      3660
   macro avg       0.72      0.65      0.67      3660
weighted avg       0.75      0.77      0.75      3660
```

Result 6

DecisionTreeClassifier

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees. We are using sklearn library to implement machine learning module Decision tree classifier. Similarly for decision tree classifier we first trained the model on the TF-IDF vectorizer as an X label and then with word2vec as an X label, and airline sentiment as a target label or Y label respectively.

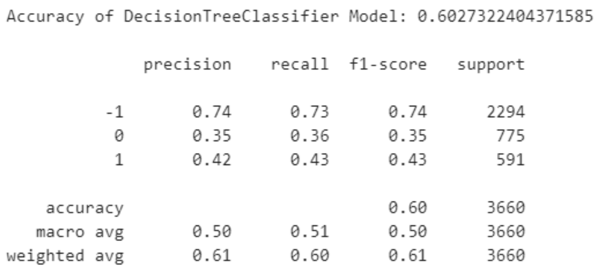
After taking the TFIDF features results are:



```
Accuracy of DecisionTreeClassifier Model: 0.6953551912568307

              precision    recall  f1-score   support

          -1       0.80      0.80      0.80      2294
           0       0.46      0.46      0.46       775
           1       0.59      0.59      0.59       591

    accuracy                           0.70      3660
   macro avg       0.62      0.62      0.62      3660
weighted avg       0.69      0.70      0.69      3660
```

Result 7

After taking the word2vec vectorizer features results are:



```
Accuracy of DecisionTreeClassifier Model: 0.6027322404371585

              precision    recall  f1-score   support

          -1       0.74      0.73      0.74      2294
           0       0.35      0.36      0.35       775
           1       0.42      0.43      0.43       591

    accuracy                           0.60      3660
   macro avg       0.50      0.51      0.50      3660
weighted avg       0.61      0.60      0.61      3660
```

Result 8

Deep Learning approaches

RNN with LSTM

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behaviour. Long short-term memory **(LSTM**) is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data

Steps involved in creating a deep learning model for text data are:

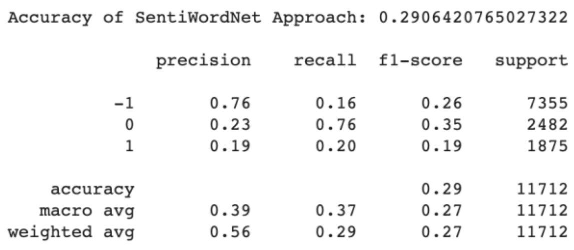
1. One hot encoding the text- The machine cannot understand words and therefore it needs numerical values so as to make it easier for the machine to process the data. To apply any type of algorithm to the data, we need to convert the categorical data to numbers. To achieve this, one hot ending is one way as it converts categorical variables to binary vectors. We are one hot encoding the corpus using keras.preprocessing.text
2. Embedding Representation - Embedding and padding the one hot representation of the text .An embedding is a dense vector of floating point values (the length of the vector is a parameter you specify). Instead of specifying the values for the embedding manually, they are trainable parameters (weights learned by the model during training, in the same way a model learns weights for a dense layer). Embedding is done with the help of tensorflow.
3. Deploying the LSTM neural network using from tensorflow.keras.layers import Bidirectional, LSTM.

After using the LSTM approach the accuracy comes out to be 0.211:

SentiWordNet

It is an opinion lexicon derived from the WordNet database where each term is associated with numerical scores indicating positive and negative sentiment information. The results indicate It could be used as an important resource for sentiment classification tasks.

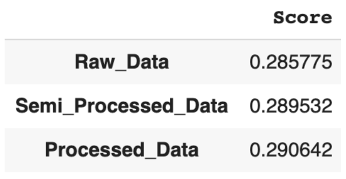
The sentences were tokenized and sent to it on word level and the positive and negative score were differenced and the final score was aggregated.
SentiWordNet was found not accurate when the target labels were non-binary.



```
Accuracy of SentiWordNet Approach: 0.2906420765027322

              precision    recall  f1-score   support

          -1       0.76      0.16      0.26      7355
           0       0.23      0.76      0.35      2482
           1       0.19      0.20      0.19      1875

    accuracy                           0.29     11712
   macro avg       0.39      0.37      0.27     11712
weighted avg       0.56      0.29      0.27     11712
```

Result 9

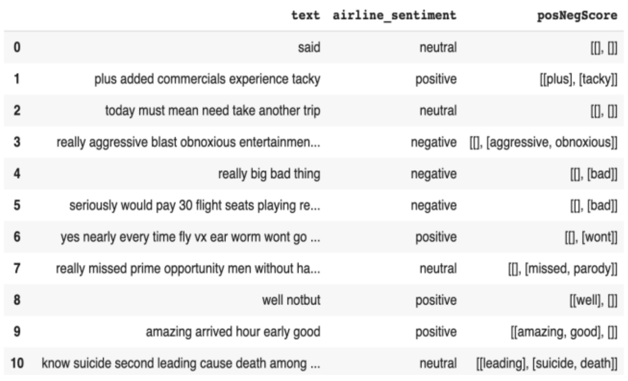
The operation was further carried on raw text, semi-processed text and processed text.



	Score
Raw_Data	0.285775
Semi_Processed_Data	0.289532
Processed_Data	0.290642

Result 10

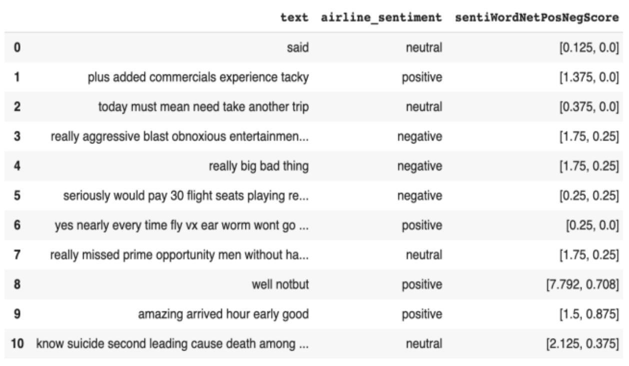
After that a slightly different approach was applied to get the positive text from negative labelled words and vice versa .The positive and the negative bucket list along with the dataset present with us was used to compare words in sentence strings.



	text	airline_sentiment	posNegScore
0	said	neutral	[[], []]
1	plus added commercials experience tacky	positive	[[plus], [tacky]]
2	today must mean need take another trip	neutral	[[], []]
3	really aggressive blast obnoxious entertainmen...	negative	[[], [aggressive, obnoxious]]
4	really big bad thing	negative	[[], [bad]]
5	seriously would pay 30 flight seats playing re...	negative	[[], [bad]]
6	yes nearly every time fly vx ear worm wont go ...	positive	[[], [wont]]
7	really missed prime opportunity men without ha...	neutral	[[], [missed, parody]]
8	well notbut	positive	[[well], []]
9	amazing arrived hour early good	positive	[[amazing, good], []]
10	know suicide second leading cause death among ...	neutral	[[leading], [suicide, death]]

Figure 6

The sentences were then used to calculate the positive and negative score using SentiWordNet 3.0

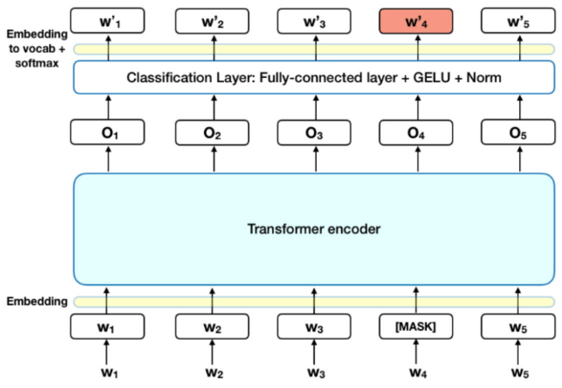


	text	airline_sentiment	sentiWordNetPosNegScore
0	said	neutral	[0.125, 0.0]
1	plus added commercials experience tacky	positive	[1.375, 0.0]
2	today must mean need take another trip	neutral	[0.375, 0.0]
3	really aggressive blast obnoxious entertainmen...	negative	[1.75, 0.25]
4	really big bad thing	negative	[1.75, 0.25]
5	seriously would pay 30 flight seats playing re...	negative	[0.25, 0.25]
6	yes nearly every time fly vx ear worm wont go ...	positive	[0.25, 0.0]
7	really missed prime opportunity men without ha...	neutral	[1.75, 0.25]
8	well notbut	positive	[7.792, 0.708]
9	amazing arrived hour early good	positive	[1.5, 0.875]
10	know suicide second leading cause death among ...	neutral	[2.125, 0.375]

Figure 7

BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. Bidirectional Encoder Representations from Transformers is a technique for NLP pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google. Google is leveraging BERT to better understand user searches.



W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

Figure 8

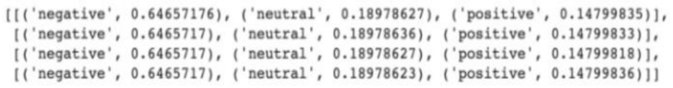
Using KTRAIN with given database

The training and testing dataset was divided in the ration 80:20. The new training dataset was sent for pre- processing and then to ktrain's text classifier for training and the learner model was instantiated.

However to train with epochs we needed to find the learner rate and for that we plotted the graph and the optimal learning rate (lesser loss and with optimal time) was 1e-3(10^-3) and the result for the

["*the service is good*","*The cost is expensive and customer service sucked,the flight was late but prices are ok, service is fine and cost is also fine*"]

Came out to be which was inaccurate



```
[[('negative', 0.64657176), ('neutral', 0.18978627), ('positive', 0.14799835)],
 [('negative', 0.6465717), ('neutral', 0.18978636), ('positive', 0.14799833)],
 [('negative', 0.6465717), ('neutral', 0.18978627), ('positive', 0.14799818)],
 [('negative', 0.6465717), ('neutral', 0.18978623), ('positive', 0.14799836)]]
```

Result 11

Using KTRAIN with data balancing

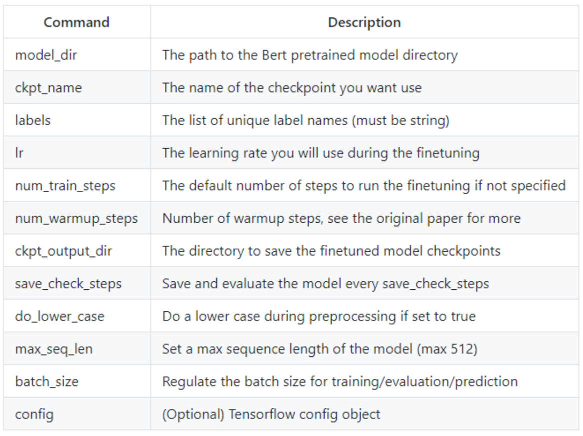
As mentioned earlier the data target class label was highly imbalances as there were *9300 Negative*, *2100 Positive and 3099 Neutral* which shows that the data is sort of biased towards a particular class label.

To handle the biasness, this time the learner model was trained with balanced data in the same ratio 80:20 but this time only 6000 rows of data was divided. 2000 For each class - +ve,-ve, neutral. And after applying the above method the optimal learner rate was optimally improved to 1e-6(10^-6). However the results were still not accurate to an acceptable mark as the results were biased towards neutral or positive class this time and none towards negative.

The conclusion of this method was that it is also not beneficial for out use case unless fine-tuned.

Bert using BertFTModel from Bert Library

This approach required GPU and so we used Google Collab GPU Instance, and the same dataset with text and target label columns was used. The uncased 12 layer bert model was downloaded from Google's resources. Another requirement was tensorflow-gpu==1.15.0 as the latest TensorFlow was not supported in Bert Library



Command	Description
model_dir	The path to the Bert pretrained model directory
ckpt_name	The name of the checkpoint you want use
labels	The list of unique label names (must be string)
lr	The learning rate you will use during the finetuning
num_train_steps	The default number of steps to run the finetuning if not specified
num_warmup_steps	Number of warmup steps, see the original paper for more
ckpt_output_dir	The directory to save the finetuned model checkpoints
save_check_steps	Save and evaluate the model every save_check_steps
do_lower_case	Do a lower case during preprocessing if set to true
max_seq_len	Set a max sequence length of the model (max 512)
batch_size	Regulate the batch size for training/evaluation/prediction
config	(Optional) Tensorflow config object

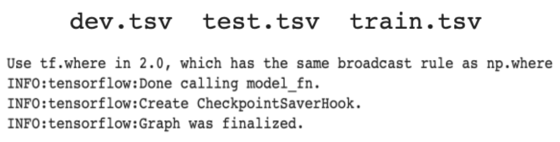
Table 1

Labels	*[1,0,-1]*
NUM_TRAIN_STEPS	*30000*
NUM_WARMUP_STEPS	*1000*
SAVE_CHECK_STEPS	*1000*
DO_LOWER_CASE	*FALSE*

MAX_SEQ_LEN	*50*
BATCH_SIZE	*32*
TRAIN_SIZE	*0.80*
VAL_SIZE	*0.05*
TEST_SIZE	*0.1*
Learner_Rate	*1e-5*

Table 2

The data was itself divided into train test and validation by the model



```
dev.tsv   test.tsv   train.tsv
```

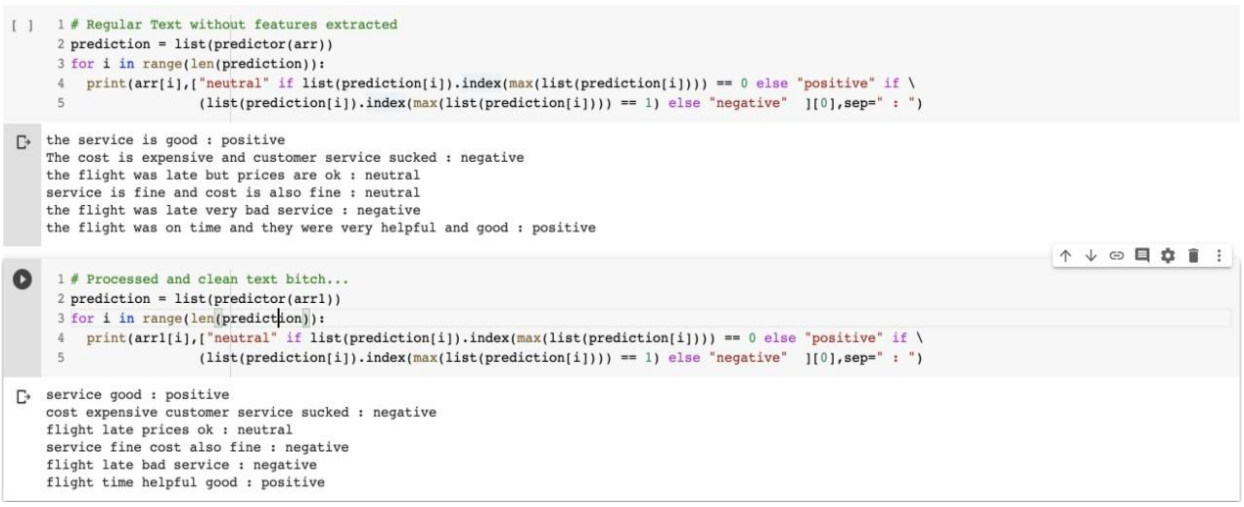
```
Use tf.where in 2.0, which has the same broadcast rule as np.where
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
```

Figure 9

The model was tuned to these parameters and the trainer and predictor was obtained from the BERT_FT_MODEL.
After training the final fine tuned model came out to be accurate up to 95%.The model was frozen and saved for further us

Saving, Loading and Using the Model

The meta graph was imported from the saved model and then appended to the graph and saved over cloud storage. To load and use the model the array of sentences are passed to the predictor and features are extracted from it. Model processed those sentences through *run_classifier.py* and then passes them for prediction.



```
# Regular Text without features extracted
prediction = list(predictor(arr))
for i in range(len(prediction)):
  print(arr[i],["neutral" if list(prediction[i]).index(max(list(prediction[i]))) == 0 else "positive" if \
               (list(prediction[i]).index(max(list(prediction[i]))) == 1) else "negative"  ][0],sep=" : ")
```

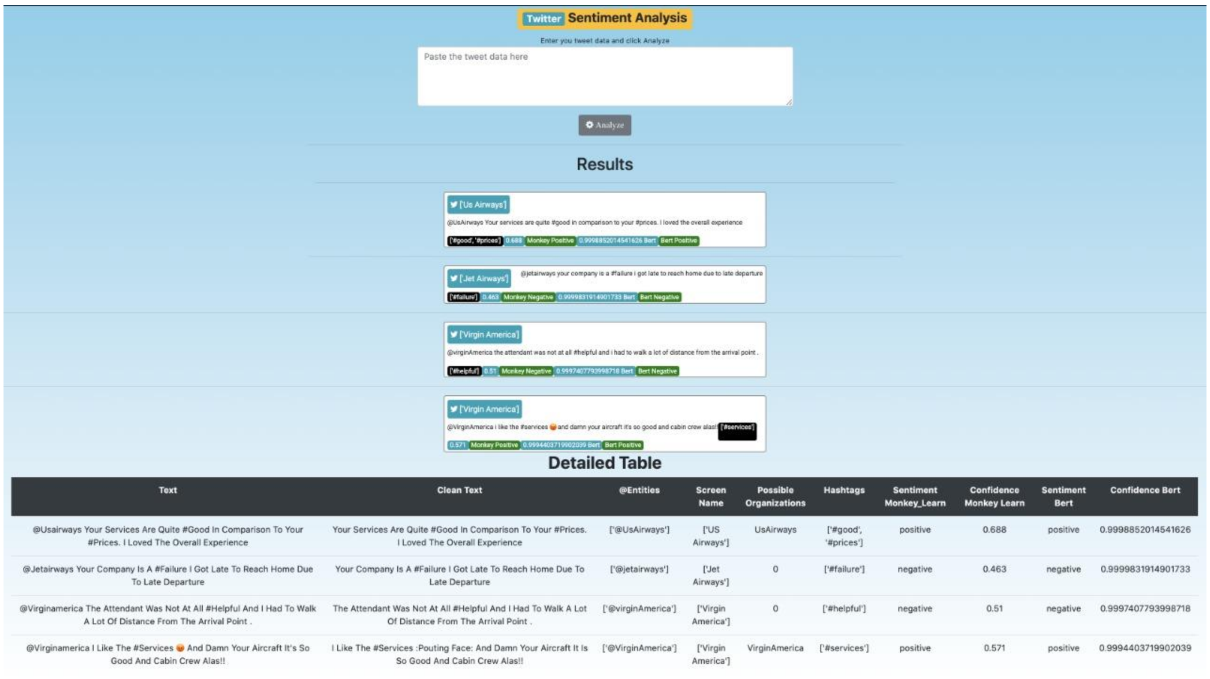
```
the service is good : positive
The cost is expensive and customer service sucked : negative
the flight was late but prices are ok : neutral
service is fine and cost is also fine : neutral
the flight was late very bad service : negative
the flight was on time and they were very helpful and good : positive
```

```
# Processed and clean text bitch...
prediction = list(predictor(arr1))
for i in range(len(prediction)):
  print(arr1[i],["neutral" if list(prediction[i]).index(max(list(prediction[i]))) == 0 else "positive" if \
               (list(prediction[i]).index(max(list(prediction[i]))) == 1) else "negative"  ][0],sep=" : ")
```

```
service good : positive
cost expensive customer service sucked : negative
flight late prices ok : neutral
service fine cost also fine : negative
flight late bad service : negative
flight time helpful good : positive
```

Figure 10

Web Application


Twitter Sentiment Analysis

Enter you tweet data and click Analyze

Paste the tweet data here

Analyze

Results

['Us Airways']
@UsAirways Your services are quite #good in comparison to your #prices. I loved the overall experience
['#good', '#prices'] 0.688 Monkey Positive 0.9998852014541626 Bert Bert Positive

['Jet Airways'] @jetairways your company is a #failure i got late to reach home due to late departure
['#failure'] 0.463 Monkey Negative 0.9999831914901733 Bert Bert Negative

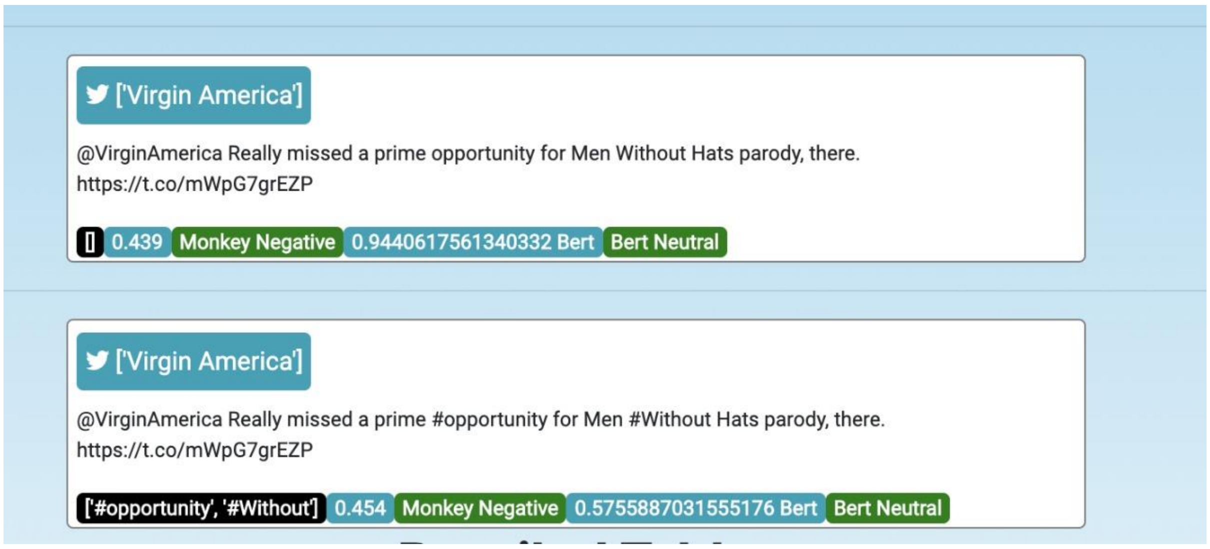
['Virgin America']
@virginAmerica the attendant was not at all #helpful and i had to walk a lot of distance from the arrival point .
['#helpful'] 0.51 Monkey Negative 0.9997407793998718 Bert Bert Negative

['Virgin America']
@VirginAmerica i like the #services 😡 and damn your aircraft it's so good and cabin crew alas!! ['#services']
0.571 Monkey Positive 0.9994403719902039 Bert Bert Positive

Detailed Table

Text	Clean Text	@Entities	Screen Name	Possible Organizations	Hashtags	Sentiment Monkey_Learn	Confidence Monkey Learn	Sentiment Bert	Confidence Bert
@Usairways Your Services Are Quite #Good In Comparison To Your #Prices. I Loved The Overall Experience	Your Services Are Quite #Good In Comparison To Your #Prices. I Loved The Overall Experience	['@UsAirways']	['US Airways']	UsAirways	['#good', '#prices']	positive	0.688	positive	0.9998852014541626
@Jetairways Your Company Is A #Failure I Got Late To Reach Home Due To Late Departure	Your Company Is A #Failure I Got Late To Reach Home Due To Late Departure	['@jetairways']	['Jet Airways']	0	['#failure']	negative	0.463	negative	0.9999831914901733
@Virginamerica The Attendant Was Not At All #Helpful And I Had To Walk A Lot Of Distance From The Arrival Point .	The Attendant Was Not At All #Helpful And I Had To Walk A Lot Of Distance From The Arrival Point .	['@virginAmerica']	['Virgin America']	0	['#helpful']	negative	0.51	negative	0.9997407793998718
@Virginamerica I Like The #Services 😡 And Damn Your Aircraft It's So Good And Cabin Crew Alas!!	I Like The #Services :Pouting Face: And Damn Your Aircraft It Is So Good And Cabin Crew Alas!!	['@VirginAmerica']	['Virgin America']	VirginAmerica	['#services']	positive	0.571	positive	0.9994403719902039

Figure 11


['Virgin America']

@VirginAmerica Really missed a prime opportunity for Men Without Hats parody, there. https://t.co/mWpG7grEZP

[] 0.439 Monkey Negative 0.9440617561340332 Bert Bert Neutral

['Virgin America']

@VirginAmerica Really missed a prime #opportunity for Men #Without Hats parody, there. https://t.co/mWpG7grEZP

['#opportunity', '#Without'] 0.454 Monkey Negative 0.5755887031555176 Bert Bert Neutral

Figure 12

Conclusion

Over the period of time given to this project several algorithms and models were tried for getting the classification model and at the end fine tuning Bert Model was found to be more accurate and also required more resources than an ordinary model.

References

- https://nlp.stanford.edu/software/CRF-NER.html
- https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da
- https://arxiv.org/pdf/1509.04219.pdf
- https://www.cse.ust.hk/~rossiter/independent_studies_projects/twitter_emotion_analysis/twitter_emotion_analysis.pdf
- https://towardsdatascience.com/3-ways-to-optimize-and-export-bert-model-for-online-serving-8f49d774a501
- https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-5-50b4e87d9bdd
- https://www.kaggle.com/menion/sentiment-analysis-with-bert-87-accuracy
- https://towardsdatascience.com/multi-class-sentiment-analysis-using-bert-86657a2af156