TWITTER SENTIMENT ANALYSIS

Anshaj Goyal
Faculty Of Engineering,
Dayalbagh Educational Institute, Agra - 282005
anshaj.goyal23@gmail.com

Yash Satwani
Faculty Of Engineering,
Dayalbagh Educational Institute, Agra- 282005
yash.satwani0@gmail.com

Shreya Sharma
Faculty Of Engineering,
Dayalbagh Educational Institute, Agra- 282005
qv.shreyasharma2001@gmail.com

Kush Kumar
Faculty Of Engineering,
Dayalbagh Educational Institute, Agra- 282005
kushkumar89792666@gmail.com

Deepak
Faculty Of Engineering,
Dayalbagh Educational Institute, Agra- 282005
dk5114360@gmail.com

1 Abstract

This project addresses the problem of training models for sentiment analysis of a given dataset tweets from the users for the US Airlint Insudtry. 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.

2 Introduction

2.1 Domain Introduction

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 this project was focused 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 the results of this problem is given below along with illustration of how to compute the required metric.

2.2 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 assuming 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

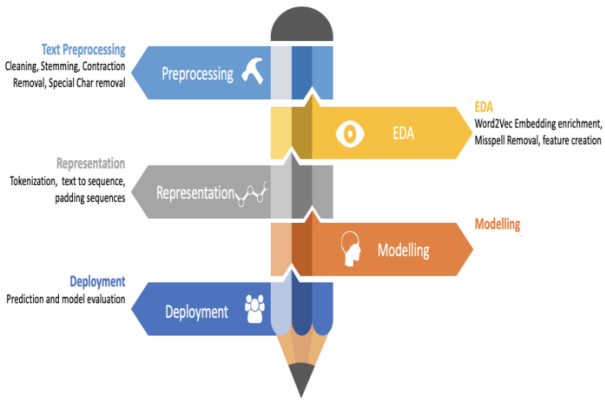

Figure 1 - Categori [1]es

3 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.

4 Feature Engineering

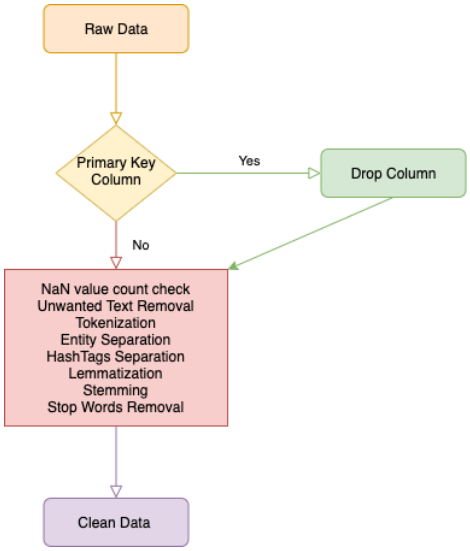

Figure 2 – Feature Engineering

4.1 Data Cleaning

Since this is a lot of information to handle therefore performing some filtering techniques in order to extract the important aspects of data becomes and discarding the rest becomes a mandatory step for proceeding futher.

- 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 tokens can be looked upon as individual components that make up a tweet.
- URL's and user references (identified by tokens "http" and "@") are removed as the primary objective from the solution of this problem is analysing the sentiment of the tweet.
- Punctuation marks and digits/numerals may be removed if for example the tweets need to be compared with 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 the need to deal with complex grammatical transformations of the word becomes irrelevant. In this use case employing the algorithm of "porter stemming" on both the tweets and the dictionary was necessary, 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 increased the value of text.
- Punctuation and StopWords Removal
- Lemmatization and tokenization

5 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.

5.1 Most Frequent tags V/S Tweets

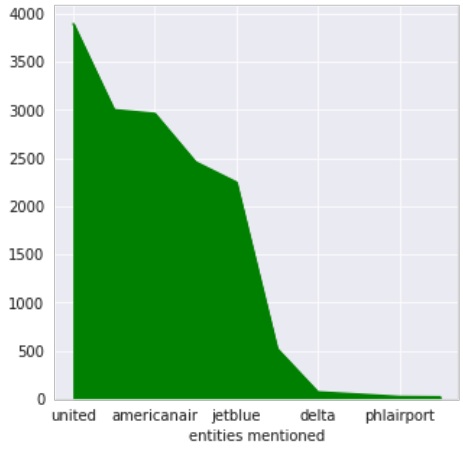

Figure 3 – Graph for Most Frequent tags V/S Tweets

Inference: From this area plot, what's observed is that the tags are of US based airlines which makes sense as the dataset given was titles US Airline Tweets Dataset. Also the most occurring tag is United Airways.

5.2 Entity Counts

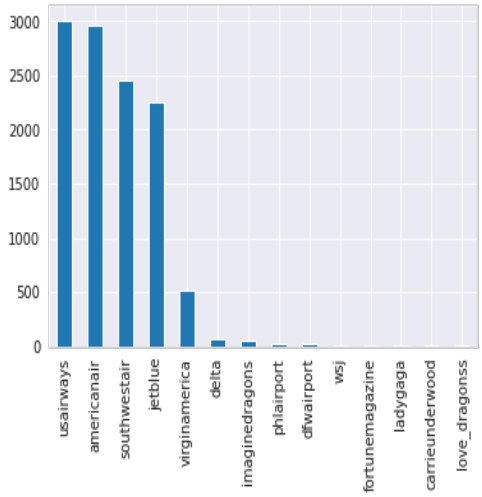

Figure 4 – Graph showing the counts of entities

- **Inference** – The dataset focuses on 4 major airlines or if scope is widened a little then 5 airlines.

5.3 Timezone V/S Tweet

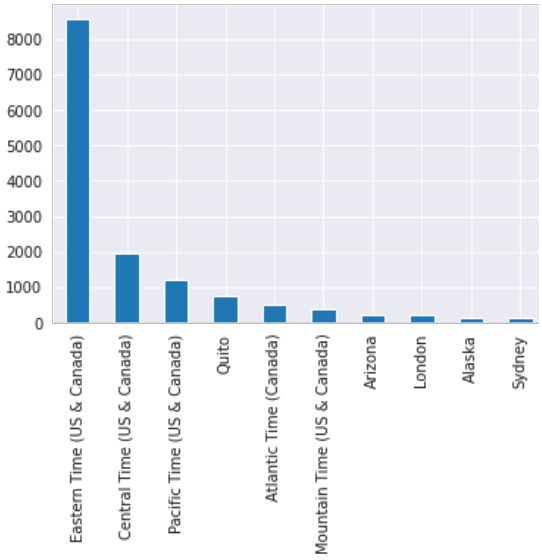

Figure 5 - Graph showin Timezone v/s Tweet

- **Inference** – Most tweets are coming from Eastern Timezone (US & Canada) as this also makes much sense that the dataset is again of US based airlines.

5.4 Target Label Class Distribution

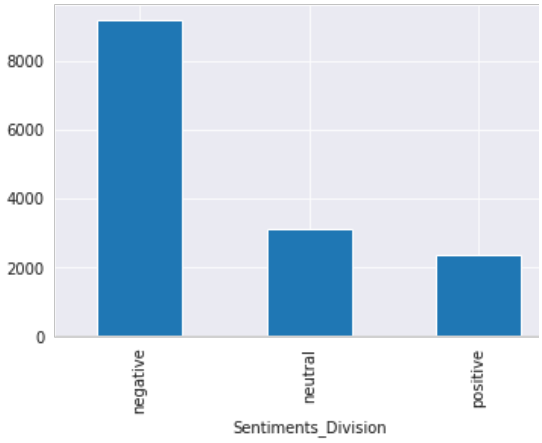

Figure 6 - Graph showing the distribution of Target label Class

- **Inference** – The visualization shows highly class imbalancy and it can cause a problem while model training for classification as imbalancy leads to biasness.

5.5 Word Clouds

- **Thank** and **Flight** are the most common words in the whole text column, one being a concrete noun and the other being adjective.

- People are using ***flight*** with other adjectives such as *help, time, cancelled, delay,* and other objects related to them as well such as *bag, luggage*.

Figure 7 - Word Cloud

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

- The use of word ***Today*** shows that most users are doing their tweets on the same day of their travel.

Figure 8 - Word Cloud

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

- Use of the word ***time*** is mostly seen in tweets which shows that the majority of negative tweets are for users who are complaining for a delayed flight or so.

- The word thank can also be seen but logically word thank is a positive labelled word but presence of this word here signifies that some tweets are sarcasm based negatively labelled

- Tweets are also frequently mentioning the word help that means the users are actually are actually relying on the airline's social media handle for solving their problems. This makes it clear that companies regardless of company providing a product/service they should be active on their social media handle also as it provides interactivitiy with their customers.

Figure 9 - Word Cloud

5.6 50 most common words

Flight, thanks, cancelled, service, help, time, customer and others.

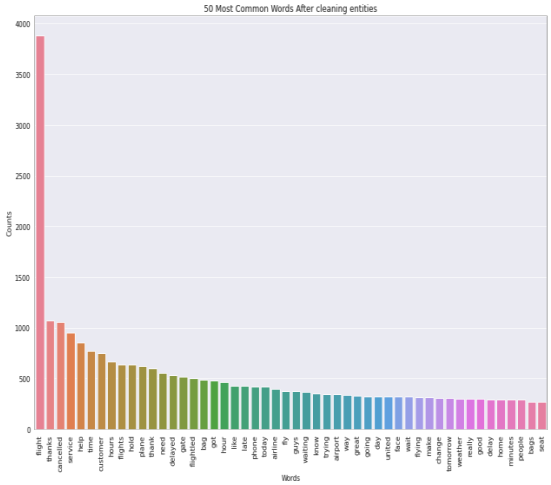

Figure 10 - Graph showing 50 common words

5.7 Tweets V/S Time distribution

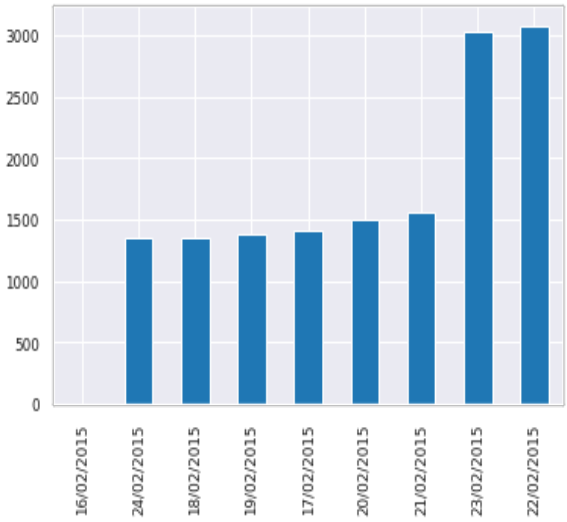

Figure 11 - Graph for Tweet v/s Time

- **Inference -** Our data had a tweet time range of 16/02/2015 to 22/02/2015
- In the given plot, most of the tweets are created on 22/02/2015 which was Sunday which shows that that people travel more often on weekends.

5.8 Positive tweets vs. Timing

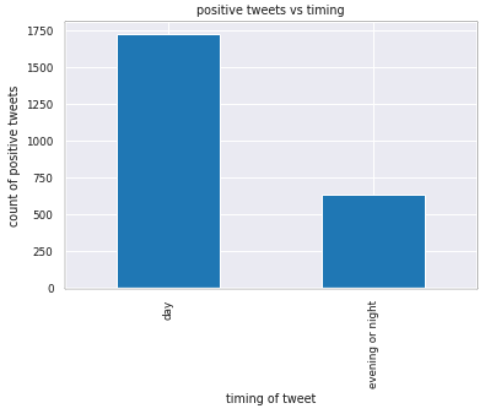

Figure 12 -Graph for Positive tweets v/s Timing

5.9 Neutral tweets vs. Timing

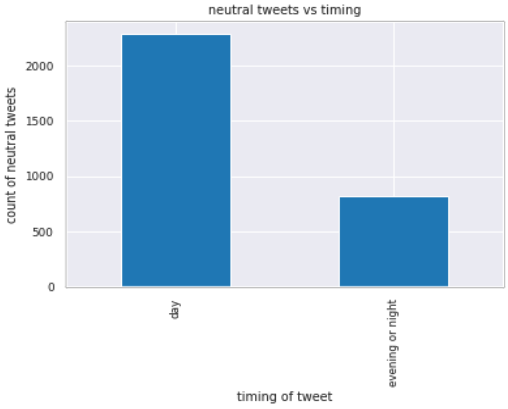

Figure 13 - Graph for Neutral tweets v/s Timing

5.10 Negative tweets vs. Timing

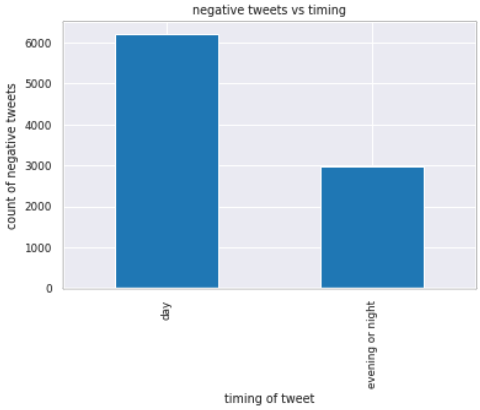

Figure 14 - Graph for Negative tweets v/s Timing

• **Inference –** From above three plots it's clear that regardless of sentiment more tweets are happening in day time rather than evening or night.

5.11 Sentiment V/S Airlines

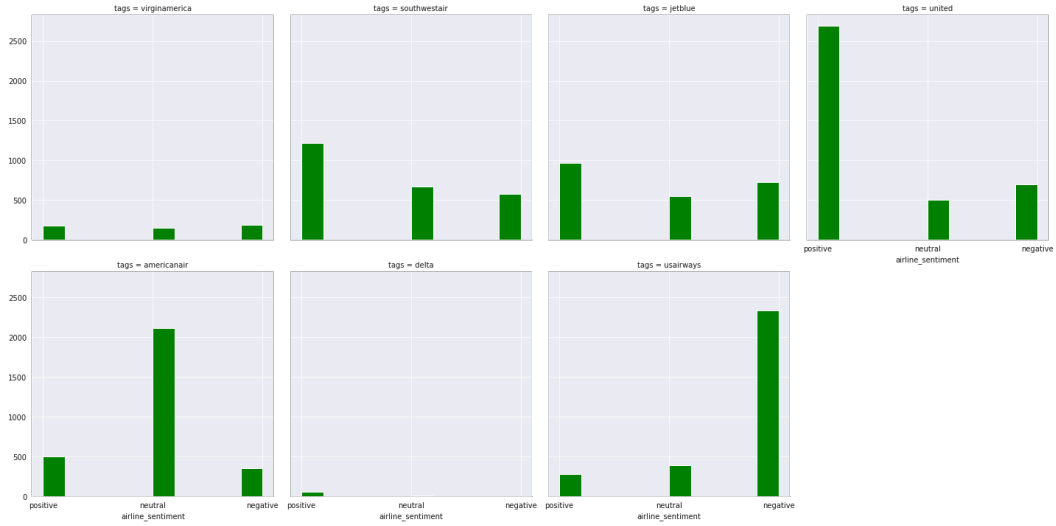

Figure 15- Graph showing Sentiment v/s Airlines

- **Inference** – Delta Airways has almost nill number of tweets which raises several concerns such as –
  - ➢ Is their social handle not active that customers are not mentioning it there
  - ➢ Or is the airline not getting users who are using their airlines which means it is in loss for sure.

5.12 Airline V/S Sentiment

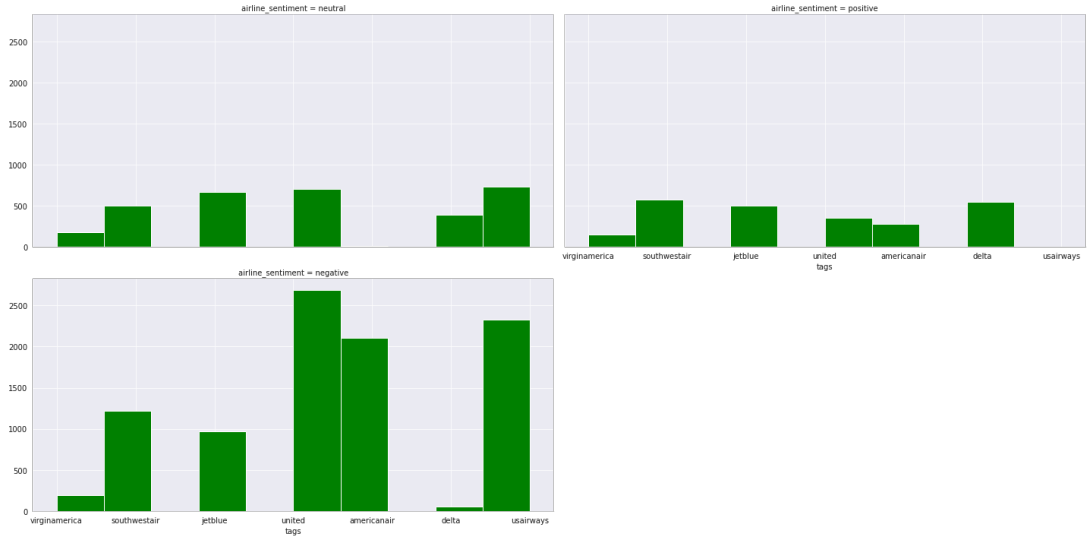

Figure 16 - Graph showing Airlines v/s Sentiment

- **Inference** – Negative Tweets are much higher in number for all airlines then their positive or neutral.
- It also shows that people are more inclined towards complaining for bad service but not praise for good service.

5.13 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.

- **Inference -** What was observed is that [“Cancellation”, “Flight”] are the most common topic among all the tweets which as one topic is an object and other is a sentiment based service of that object.

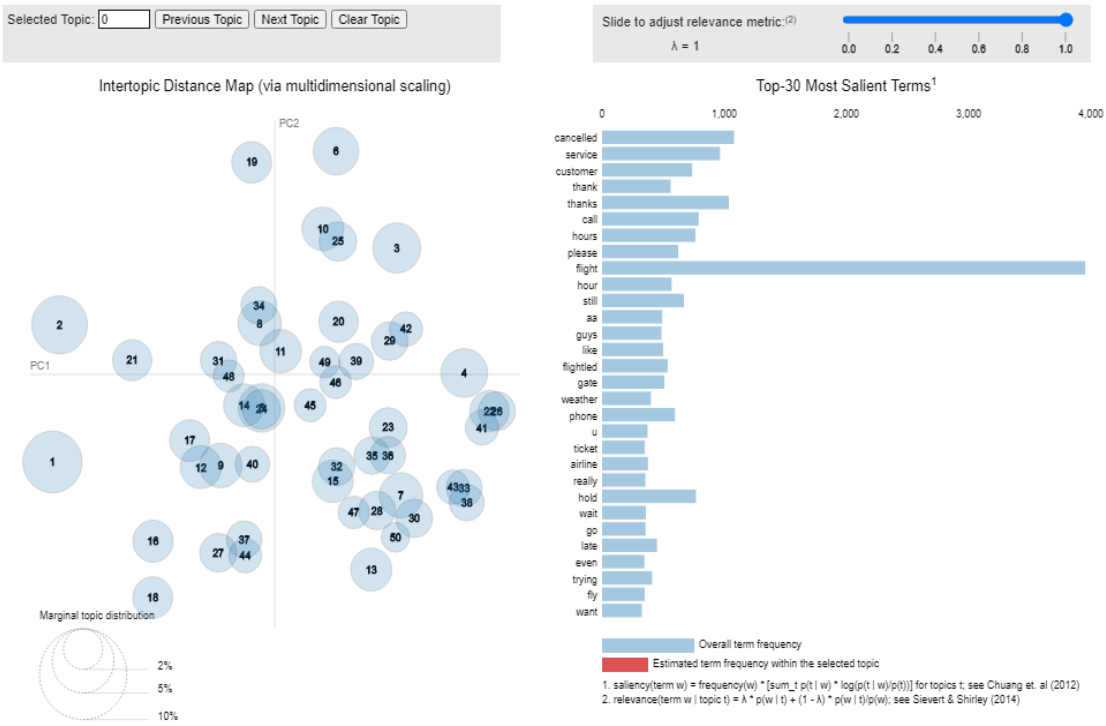

Figure 17- Graph showing top 10 topics

5.14 Timezone V/S Sentiment

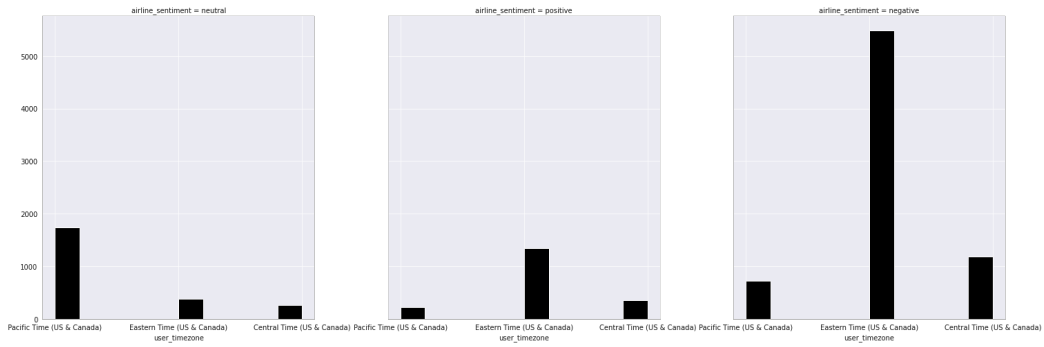

Figure 18- Graph showing Timezone v/s Sentiment (Pacific Time - Neutral, Eastern Time - Positive, Eastern Time - Negative.)

5.15 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.

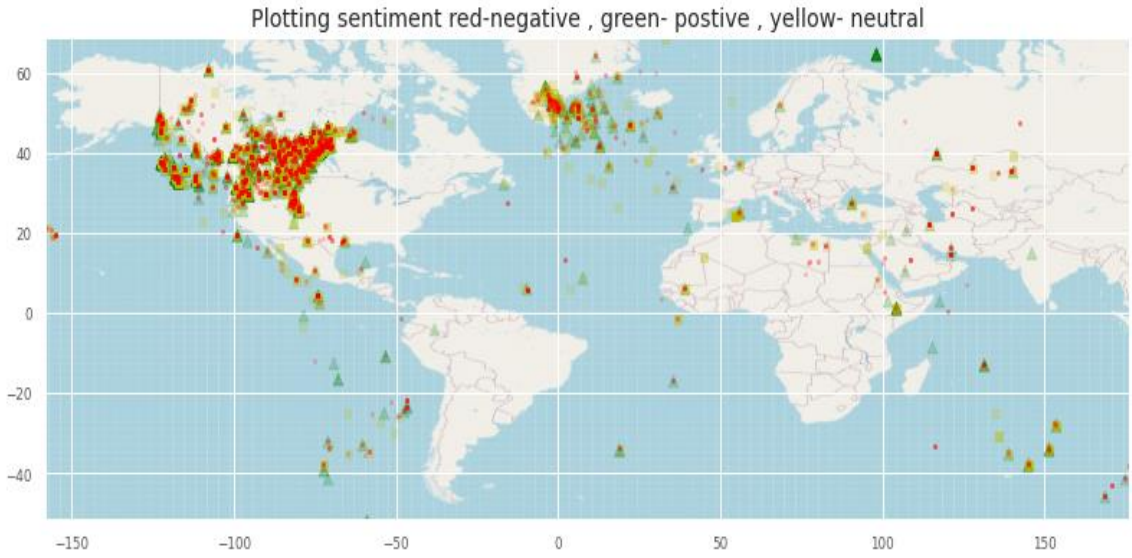

Figure 19- Visualisation of Tweet Location

6 Miscellaneous Data Handling

6.1 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.

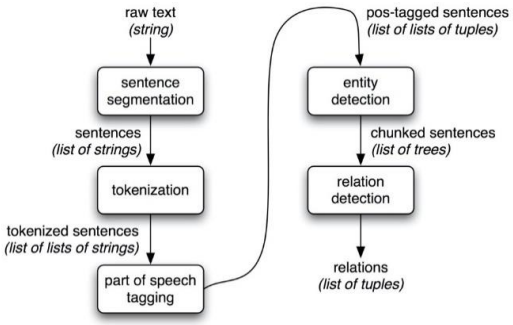

Figure 20 - NER Flowchar

The basic technique that which is used 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 21 - Tokenization

6.2 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 [2]. 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.

6.3 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 22 - Entity Screen Names

6.4 Polyglot Entity Recognition

Polyglot is a natural language pipeline that supports massive multilingual applications [3]. 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 23 - After NER approach

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

6.5 NLTK NER

The Natural Language Toolkit, or more commonly NLTK [4], 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 24 - Result after NLTK NER

7 Converting text to features:

7.1 Proposed System

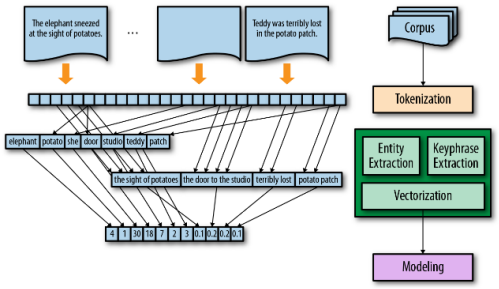

Figure 25 - Proprosed System

7.1.1 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 term frequency and inverse document frequency are multiplied to convert the sentences into vectors, now one of the disadvantage of bag of words is that there is no semantic meaning, but much semantic difference because in values like ones or zeros all the values, and 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 should be given a high score. But if a word appears in too many other documents, it's probably not a unique identifier, therefore a lower score to that word. The math formula for this measure:

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

7.1.2 Word2vec approach

Machine learning and deep learning algorithms cannot accept text directly but need some sort of numerical representation so that the algorithms can process the data and in simple machine learning applications. Here *countvectorizer* or *tf-idf* both of which do not preserve any relationship between the words is used. [5] This is where word embeddings come in to 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 one can 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 Word2Vec just by iterating through a large

corpus of text like a Wikipedia or newspaper corpus .

Word2vec embedding from *gensim* library is used in this problem. The trained *word2vec* model is of Google News data, which was trained on about 100 billion words.

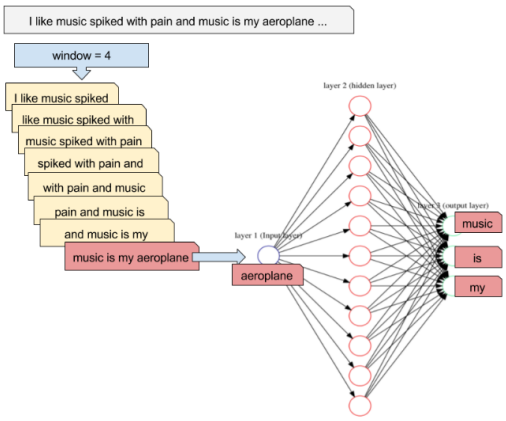

Figure 26 - Word2vec Approch

8 Classification Experiment Models

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

8.1 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. First the model was trained 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
```

Figure 27- Results after TFICF

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
```

Figure 28 - Result After Word2vec

8.2 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. First the model was trained 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 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

Figure 29 -Results after TFIDF

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

Figure 30 - Results after Word2Vec

8.2.1 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, each data item is plotted 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. First the model was trained 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 , similary here also sklearn library is used to implement SVM. 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

Figure 31- Results after TFIDF

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

Figure 32- Results after Word2Vec

8.2.2 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. First the model was trained 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.

After taking the **TFIDF** features results are:

Accuracy of DecisionTreeClassifier Model: 0.69535519125683

	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

Figure 33 -Results after TFIDF

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

Figure 34- Results after Word2Vec

8.3 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

Figure 35 - Result of SentiWordNet

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

Figure 36 - Result on semi-processed and processed text

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 37 - Result after applying different approach

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 38 - Result after using SentiWordNet3.0

9 Deep Learning approaches

9.1 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, the data needs to be converted to numerical and with categorical if possible. To achieve this, one hot ending is one way as it converts categorical variables to binary vectors. OneHot encoding is applied on 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 results are:

Accuracy of RNN with LSTM Model is 0.21174863387978143

	precision	recall	f1-score	support
-1	0.00	0.00	0.00	2294
0	0.21	1.00	0.35	775
1	0.00	0.00	0.00	591
accuracy			0.21	3660
macro avg	0.07	0.33	0.12	3660
weighted avg	0.04	0.21	0.07	3660

Figure 39 - Result of LSTM

9.2 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. [1]

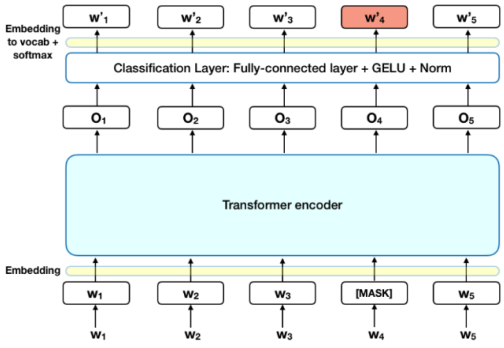

Figure 40 – BERT

9.2.1 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 the model learner rate needs to found out and for that loss v/s time is plotted on the graph and the optimal learning rate (lesser loss and with optimal time) is visualized.
The LR value taken here is 1e-5(10^-5) and the result for the

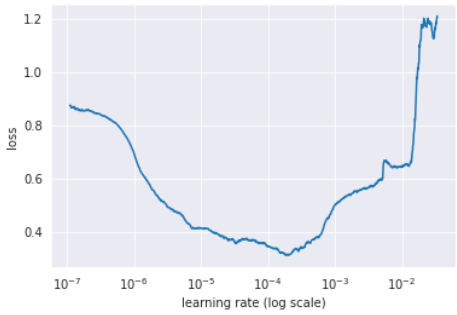

Figure 41 - Learning Rate Graph

```
The Accuracy of Ktrain Bert is 0.663433333
the service is good                                          positive   0.5919047
The cost is expensive and customer service sucked            negative   0.92250854
the flight was late but prices are ok                        negative   0.42837232
service is fine and cost is also fine                        negative   0.7474042
the flight was late very bad service                         negative   0.9489741
the flight was on time and they were very helpful and good   positive   0.7778431

service good                                                 positive   0.64996094
cost expensive customer service sucked                       negative   0.9616174
flight late prices ok                                        negative   0.88861686
service fine cost also fine                                  negative   0.91300166
flight late bad service                                      negative   0.9324886
flight time helpful good                                     positive   0.7085946
```

Figure 42 - Results after KTRAIN

9.2.2 Using KTRAIN with data balancing

As mentioned earlier the data target class label was highly imbalanced 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 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 also changed to 1e-4(10^-4). 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 our use case unless fine-tuned.

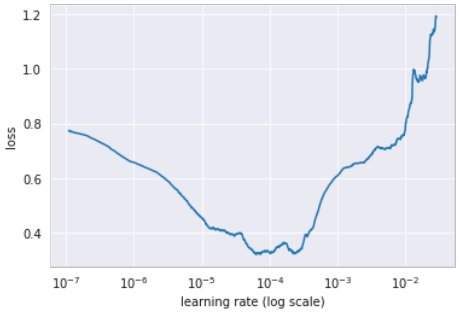

Figure 43 - Learning Rate Graph

```
Accuracy of Ktrain with data balancing is 0.583444

the service is good                                          negative   0.83204716
The cost is expensive and customer service sucked            negative   0.89305705
the flight was late but prices are ok                        negative   0.49836433
service is fine and cost is also fine                        negative   0.4783449
the flight was late very bad service                         negative   0.90718096
the flight was on time and they were very helpful and good   positive   0.8665206

service good                                                 negative   0.8993717
cost expensive customer service sucked                       negative   0.8860653
flight late prices ok                                        negative   0.64744014
service fine cost also fine                                  negative   0.7672852
flight late bad service                                      negative   0.90670586
flight time helpful good                                     positive   0.79948854
```

Figure 44 - Result after KTRAIN with Data Balancing

9.2.3 Bert using BertFTModel from Bert Library

This approach requires GPU and tensorflow version 1.15.0.

Google Collab GPU Instance is used to meet the above requirements, and the same dataset with text and target label columns was used. The uncased 12 layer bert model was downloaded from Google's resources.

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

Figure 45- Description of Commands

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 1 - Parameter values

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

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

Figure 46- Train - Test - Validate set

```
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 47 - BRET_FT_MODEL

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 use

10 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.

```
[ ] 1 # Regular Text without features extracted
    2 prediction = list(predictor(arr))
    3 for i in range(len(prediction)):
    4   print(arr[i],["neutral" if list(prediction[i]).index(max(list(prediction[i]))) == 0 else "positive" if \
    5                 (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

    1 # Processed and clean text bitch...
    2 prediction = list(predictor(arr1))
    3 for i in range(len(prediction)):
    4   print(arr1[i],["neutral" if list(prediction[i]).index(max(list(prediction[i]))) == 0 else "positive" if \
    5                 (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 48- Saving, Loading and Using model

11 Web Application and FrameWork

To access the tweets sentiment from the trained model Django framework was used to develop a front end website.

- It delivered sentiments of tweets from the model
- Hashtags and possible entities were displayed
- To compare our predicted sentiment, a commercial predictor's api was used
- Confidence was each sentiment from each model was displayed
- The framework was hosted on google colab with some tweaks and workarounds.

11.1 API backend

```
class work():
    def __init__(self):
        self.counter = 0
        self.model = instantiateModel()

    def index(self,request):
        text = request.POST.get('text')
        if not text: return render(request,'index.html')
        text = text.split("\n")
        text = [i for i in text if i not in ["","\r"]]
        self.model.set_sentence(text)
        bert_sentiments = self.model.returnSentiment()
        monkey = self.model.monkey()
        findHashTags = self.model.findHashTags()['tags']
        findatTheRates = self.model.findatTheRates()['@']
        twitName = self.model.twitName()['twitName']
        nltkEntity = self.model.nltkEntity()['nltk']
        sentence_bert,labelBert,confidenceBert = bert_sentiments['sentiment']
        sentence_monkey,labelMonkey,confidenceMonkey = monkey['sentiment']
        result = zip(sentence_bert,sentence_monkey,findatTheRates,twitName,\
            nltkEntity,findHashTags,labelMonkey,confidenceMonkey,labelBert,confidenceBert)
        return render(request, 'index.html', {"result":result, "result1":result})
```

Figure 49- API backend

Executing API:
POST /
Host:
User-Agent:
Connection:
Accept: */*
Accept-Encoding:
Content-Type: xxx/www-form-urlencoded
Content-Length:

Text=&submit=submit

11.2 Web UI

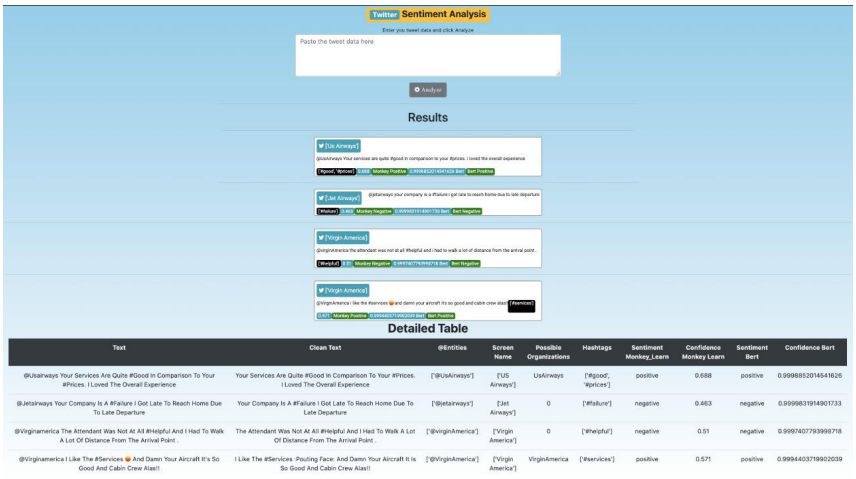

Figure 50- Web User Interface

Figure 51- Result on Web UI

12 Conclusion

Over the period of time given to this project several algorithms and models were tried for getting the classification model and following observations were taken in note:

1) Entity Extraction is a tedious task in twitter dataset as one cannot possibly incur every possible entity (Organization here) from any module.

2) SentiwordNet 3.0/2.0 didn't work well as our classification was multi label however the latter works on binary class.

3) NLTK outperformed every other module in entity recognition for our use case in comparision to Polyglot, Stanford, Spacy.

4) Logistic Regression worked better than any other standard trained models
5) Gridsearch (Parameter Tuning) in any standard model didn't make much difference in accuracy, prec, f1 score, recall.
6) Word2Vec was not beneficial in our use case as it cannot handle unknown or out-of-vocabulary words
7) RNN works well only when the input length of each text is same which was not possible in our case, and pre-setting a max_length would lead to sometimes ignorance of features.
8) Ktrain with Bert with original data length performed well in terms of actual accuracy and not accuracy score.
9) An assumption failed that was balanced data would perform well on ktrain which didn't .
10) Bert Model finetuning outperformed every model in terms of accuracy score and when actually used.
11) Bert model was performing almost similar with raw text(noisy data) and processed text(clean data).
12) Limitation of Bert was the higher usage of resources such as GPU requirement and also a depreciated version of tensorflow(1.15.0)
13) Saved model of Bert was at first bigger in size than the pre-trained model of google but when put to use and froze using tensorflow it's size drastically reduced to size lower than original model.
14) Bert Library Fine Tunded Model Out peformed Ktrain Bert with a increase of 0.35 in accuracy score

The use case of this project of sentiment analysis has a very broad scope. Here in this case the airlines could easily infer answers of various question from the sentiment of tweets.

- How the customers are feeling towards their brand?
- What issues are the customers facing the most?
- What are the average customer sentiments towards their brand?
- Are the customers satisfied?
- Which class of sentiment is covering their majority segment of customer?

13 References

[1] K. A.-A. R. I. Noureddine Azzouza, "TwitterBERT: Framework for Twitter Sentiment Analysis," 2019.

[2] M. a. P. Lafferty, "Stanford Named Entity Recognizer (NER)," 2001.

[3] V. K. B. P. a. S. S. Rami Al-Rfou, "POLYGLOT-NER: Massive Multilingual Named Entity Recognition," 2015.

[4] S. Li. [Online]. Available: https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da.

[5] N. L. Joshua Acosta, "Sentiment Analysis of Twitter Messages Using Word2vec," 2015.

[6] D. D. Rossiter. [Online]. Available: https://www.cse.ust.hk/~rossiter/independent_studies_projects/twitter_emotion_analysis/twitter_emotion_analysis.pdf.