Sentiment Analysis of Airline Reviews in Twitter Using NLP

Jossa Soto

501074572

Dr. Tamer Abdou

March 28, 2022

Table of Contents

[Abstract 3](#_Toc99316386)

[Literature Review 4](#_Toc99316387)

[Data Description and Exploratory Data Analysis (EDA) 7](#_Toc99316388)

[Methodology 11](#_Toc99316389)

[Feature Selection and Preprocessing 11](#_Toc99316390)

[Attribute Selection 11](#_Toc99316391)

[Text Preprocessing 11](#_Toc99316392)

[Feature Engineering and Model Building 13](#_Toc99316393)

[Results 15](#_Toc99316394)

[Train/Test split 15](#_Toc99316395)

[K-Fold Cross Validation 16](#_Toc99316396)

[Other metrics 19](#_Toc99316397)

[Additional tests 20](#_Toc99316398)

[Conclusion and Future Work 22](#_Toc99316399)

[References 23](#_Toc99316400)

# Abstract

Twitter is one of the most popular social networking platforms; it is a rapidly growing service with 211 million users as of late 2021 (Twitter, 2021). It follows a mini-blog style format where users can share status updates (“tweets”) in 280 characters or less. Both individuals and companies use the platform regularly. Individuals embraced Twitter as a tool to publicly voice their opinions on a variety of topics including social issues, performance of members in the entertainment industry, and company reviews. Many companies, in turn, use Twitter as a tool to reach their consumers for advertising, marketing, and general engagement. Consumer tweets are useful data points for companies as they help develop new products and services and improve existing ones. As such, data analytics is an important part of company research and development.

Sentiment analysis is the computational treatment of opinions, sentiments, and subjectivity of text (Medhat *et al.* 2014).  Sentiment analysis is one of the many natural language processing (NLP) techniques; this project overall aims to explore its utility in the classification of customer reviews. Model algorithms will be developed with the use of three classifier techniques: decision tree, random forest, and Naïve Bayes. The performance of the three algorithms’ accuracies will be evaluated through the output accuracy score and sensitivity and specificity of each individual class. To generate each algorithm, the dataset needs to be trained and tested; both the leave-on-out technique and k-fold algorithms will be applied.

For this study, tweets of airline reviews will be classified into three groups – positive, negative, and neutral – based on their sentiments. The study will focus on the application of statistical approaches including TF-IDF, bag-of-words, and n-grams to tokenize the data. The data will undergo each classifier to find the best combination of approach and classifier. This will also answer questions including: What are some of the most common words associated to each sentiment? Are there distinct differences between the three classifiers given this type of data?

The “Twitter US Airline Sentiment” dataset used in this study was retrieved from Kaggle. It contains 14,641 unique entries and 15 attributes that include the sentiment class, text (tweets), and sentiment confidence score. It can be found at https://www.kaggle.com/crowdflower/twitter-airline-sentiment.

Review, visualization, and modeling of data will be performed using R. Some key packages that will be used include “rpart” for decision tree usage, “randomForest” for random forest, and “ggplot2” for visualization.

The codes and resulted can be found on Github: https://github.com/mengziii/Big-Data-Final-JS

# Literature Review

In recent years, research towards sentiment analysis as part of Natural Language Processing (NLP) has grown. The research ranges from document level classification to learning the polarity of words and phrases (Kouloumpis *et al.* 2011). To perform sentiment analysis, a variety of approaches including machine learning, rule-based approach, and lexical based approach have been used (Devika *et al.* 2016).

Machine learning is the most popular approach, with numerous classification methods including Naïve Bayes, Maximum Entropy, K-Nearest Neighbourhood, and Support Vector Machine (SVM). The choice of method used often depends on the dataset. In many cases, more than one classification algorithm is performed to optimize the model and result in the highest accuracy score for the given data. The rule-based approach is used by defining various rules for getting the opinion. Sentences in every document are tokenized and subsequently tested. It contains a scoring system where the presence of the token results in a +1; if the overall score is greater than zero, then the document is considered positive. The lexical based approach involves calculating the orientation, that is the polarity and strength of words, phrases, or texts, in the document (Taboada *et.al* 2011). This project will be limited to the machine learning approach with a focus on the exploration and application of the decision tree, random forest, and Naïve Bayes classifiers.

Bayhaqy *et. al* (2018) performed sentiment analysis on e-commerce reactions from tweets. In their study, they compared decision trees, Naïve Bayes, and K-Nearest Neighbour classifiers. Prior to running any analysis, the data was processed and cleaned to ensure high accuracy upon analysis. Tweets frequently are composed of various characters and can include punctuation, emoticon, and HTML tags in addition to standard alphanumeric characters. For this study, URLs and punctuations were removed, and negations and emoticons were converted to text. Additionally, the tweets underwent **case folding, tokenization**, **filtering**, and **stemming**. Case folding converts all text to one case (e.g., lowercase) and stemming removes any prefixes and suffixes, which in turn reduces the word to its basic form. Filtering removes words that are of high frequency but does not contribute overall to the message of the text such as “the”'. These words are commonly referred to as “stop words''. Tokenization breaks the text into **n-grams** through statistical and semantic approaches. N-grams are the individual words or group of words that will be tested and evaluated to gather overall sentiment; some of the most common n-grams are unigram (one word) and bigrams (two words).

Bayhaqy *et. al* applied the **TF-IDF** (Term Frequency – Inverse Document Frequency) method, a statistical approach, to tokenize and weight the words. Weighting words score n-grams based on the frequency of occurrence. Term Frequency counts how often a term is found in a document while Document Frequency counts the number of documents that contain the term. The application of a statistical approach is useful, but the application of semantic approaches is also valuable. Since only TF-IDF was applied in this paper, the result is limited to frequency of the word, without the context of the words. This could lead to misclassification as a word such as “good” can be classified as positive for statements such as “good job” while negative for “good riddance”.

It was concluded that the decision tree had the highest accuracy, followed by K-Nearest Neighbour, and Naïve Bayes with the lowest accuracy. Furthermore, Naïve Bayes has the highest precision score, and decision tree has the highest recall score. It was concluded that Naïve Bayes was the best classifier for social media datasets as it makes more accurate and precise predictions. The accuracy, precision, and recall scores of Naïve Bayes and decision tree are only a few percentage points apart. This could mean that there is a data dependency in these results so my own project could have different results.

For my own project, I will apply the TF-IDF method to calculate the frequency of key words but will also briefly explore **Word2Vec** as a semantic approach to fully understand the sentiments contained in the airline tweets. Word2Vec is a model used in word embedding, a collective term for the language modelling and features of learning in NLP. Word2Vec is used to represent words as a vector and consequently calculates the similarity of words. Jatnika *et. Al* (2019) examined the calculation of similarity between words in English using word representation techniques. They applied the Word2Vec model to generate word vectors then proceeded to calculate the cosine of similarity. Words that are similar such as “bicycle” and “motorcycle” tend to have like vector values and can be grouped together. To perform their experiment, windows size and vector dimensions were selected and combined to create different configurations. Their results show that large window size and higher vector dimensions yielded in the highest (Pearson) correlation score, but really large window sizes and vector dimensions can also lead to inaccurate results as more context can weaken the similarity value. Since the goal is to identify and group words that are similar, high correlation scores are desirable.

In an article by Verma *et. al* (2021) machine learning techniques were applied to develop an algorithm to quickly differentiate messages with useful content from those that are not in situations of major environmental disasters. They performed both the Naïve Bayes (NB) and Maximum Entropy (ME) methods of classification on the data; it was found that ME overall yielded more accurate results than NB. Moreover, individual classification features including n-grams and **part-of-speech (POS) tags** were reviewed for their effectiveness. POS tagging refers to the process of categorizing words in a text to correspond to their part of speech such as noun, verb, etc. It was found that there was a high baseline accuracy for the messages with only word (unigram) and raw frequency. The addition of bigrams did not contribute significantly to the accuracy. N-grams are widely used as seen in all studies and articles in this review. The difference in accuracy scores between unigrams and bigrams could be further explored to identify any patterns in their usefulness. The use of part-of-speech tags also increased the overall accuracy of the classification; however, when they focused on adjectives to test subjectivity and tone, it was concluded that the accuracy of the classifier did not improve significantly. This led to the hypothesis that the contextual information of the tweet is more important than the presence of a specific word. This hypothesis supports the need to perform both statistical and semantic analysis of textual data to output the most accurate results.

Kouloumpis *et. al* (2011) also investigated how features that were proven to be powerful in semantic analysis in other domains apply to unconventional texts like tweets. N-grams, lexicon, parts-of-speech, and microblogging features were tested. Three datasets were used in the experiment; one sourced through Twitter directly with the use of hashtags to identify specific sentiments tweets that already have sentiment. Their results showed that the combination of n-grams and lexicon features performed the best; on the other hand, the addition of past of speech resulted in a drop in performance. This led to their conclusion that the parts-of-speech tags feature may not be useful in the microblogging domain. Their conclusion needs to be researched further as there is a probability of poor-quality data that could have impacted the results.

In addition to machine learning, lexicon-based methods are utilized to perform sentiment analysis. Arun *et. al* (2017) performed sentiment analysis on tweets made in response to the demonetization of two bank notes in India in 2016. The demonetization event led to withdrawal of the legal tender of the ₹ 500 and ₹ 1,000 denominations of banknotes of the Mahatma Gandhi Series (Reserve Bank of India, 2016) which resulted in various reactions from the Indian population. According to the set of tweets that were used in this analysis, it was concluded that there is almost equal amount of negative (38%) and neutral (39%) sentiment among the public. Among those classified as negative sentiment tweets, the word “corruption” had the highest frequency. Like the e-commerce study by Bayhaqy *et. al*, the tweets underwent pre-processing prior to any analysis. N-grams were then applied so that conditional probabilities of the next word in the tweet can be calculated. In this study, the authors performed a lexical based approach by calculating **polarity scores**. Once the data was clean and the n-gram features were added to the datasets, polarity scores were calculated on a sentence and paragraph level. Polarity scores quantify the sentiment of the message with positive or negative values. Twitter only allows for 240 characters so while scores were calculated on both sentence and paragraph levels, the sentence level scores were more reliable and appropriate for the analysis. The project focuses on machine-learning methods so polarity scores will not be calculated but it is a useful method that can be used in other scenarios. The use of polarity scores is a powerful technique in sentiment analysis as there are many established lexicons available (such as the **affin** library in Python) that contain polarity scores making the process quick and efficient.

Taboada *et.al* (2011) extended the Semantic Orientation CALculator (SO-CAL), a lexicon-based approach to extract sentiment from text using adjectives, to include other parts of speech. SO-CAL calculates the sentiment following two assumptions: individual words have **prior polarity**, that is, the semantic orientation is independent of context, and that the orientation can be expressed as a numerical value. Unlike in statistical methods where training and testing the data is necessary, lexicon-based models do not require it. This is the result of the words are assigned their real-world polarity and not the polarity of the text that they are in. The SO-CAL features were tested with the use of four different documents, all of which contain equal number of positive and negative reviews. Parts of speech including adjectives, nouns, verbs, and adverbs were grouped in different variations then each combination’s performance was evaluated through the accuracy results of output from each document. Adjectives are the part of speech that is used in SO-CAL by default. Adjectives can be easily categorized into sentiments. Among adjectives are **seed words**; words that have strong affiliation to a sentiment – negative or positive. Examples include *abysmal* and *outstanding* as their usage express specific sentiments clearly unlike words such as *ok* which can be more ambiguous. Nouns, verbs, and adverbs are used less frequently as they require additional context to express a sentiment. It was concluded that this version of SO-CAL has statistically significant improvements compared to its previous iterations. It was also concluded that lexicon-based methods are robust and can be easily enhanced with a variety of knowledge sources.

Most datasets contain high volumes of variables, some of which may not contribute to the model used in the analysis. To increase the computational efficiency of a machine learning algorithm, key variables must be selected from the dataset. Feature selection is the process of identifying these key variables and is an important step to various machine learning problems including sentiment analysis.

Isabella *et. al* (2011) executed a study to identify which feature selection method was the most appropriate for movie reviews using the KNN classifier The movies were sourced from IMDb (Internet Movie Database). A set of 200 reviews containing 40 positive and 40 negative reviews were used to evaluate the performance of the various following feature selectors: **Correlation based feature selector (CFS), Information Gain, Support Vector Machine (SVM),** and **Principal component analysis (PCA)**. CFS ranks feature subsets according to the correlation to the class. Those with high correlation with class are kept while those with low correlation or ones that display redundancy are eliminated. Information gain selects the features based on which ones minimize the entropy, that is, to limit the impurity of the elements. SVM transforms the train data into a higher dimension through nonlinear mapping. A hyperplane, a decision boundary, is outputted to separate the two classes. PCA reduces data dimensionality by transforming the original attribute space. It limits the attributes while maintaining the information and extracts the features that account for most the variance in the data.

The dataset was preprocessed before the IDF was calculated and each selector was used. It was observed that PCA achieved the maximum accuracy at 95%, outperforming the other feature selectors by as much as 27%. It was concluded that PCA was the most appropriate feature selector.

# Data Description and Exploratory Data Analysis (EDA)

The dataset used is “Twitter US Airline Sentiment” from Kaggle. There are 14, 640 observations and 15 variables. For this study, the data was sampled so the class is balanced, resulting in 7,363 records and all information reported in this section will be based on this subset.

1. ID (tweet\_id): variable ID
2. Airline sentiment (airline\_sentiment): class variable of three options – positive, negative, neutral
3. Airline sentiment confidence (airline\_sentiment\_confidence):
4. Negative reason (negativereason): categorized reason if the tweet was classified as negative
5. Negative reason confidence (negativereason\_confidence): confidence score of tweets classified as negative
6. Airline (airline): airline reviewed or tagged in the tweet
7. Airline sentiment gold (airline\_sentiment\_gold): class variable for gold members of airlines
8. Name (name): handle of the tweet creator
9. Negative reason gold (negativereason\_gold): categorized reason for the negative sentiment amongst gold members
10. Retweet count (retweet\_count): number of times that the tweet was retweeted
11. Text (text): tweet text
12. Tweet coordinates (tweet\_coord): exact coordinates of the author at the time the tweet was written
13. Tweet created (tweet\_created): date and time that the tweet was written
14. Tweet location (tweet\_location): geographical location of the author of the time the tweet was written
15. User time zone (user\_timezone): tweet author’s time zone

The data discovery, or exploratory analysis, phase consists of initial review of the data to eliminate inconsistencies, missing values, outliers and analyze distributions among other procedures. This is separate from the text pre-processing that will be required to perform NLP analysis. Figure 1 depicts the decisions made in response to the findings from this phase.

**Table 1. Data findings in the exploratory analysis phase.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Yes | No | N/A |
| Removal/replacement of missing values | ✓ |  |  |
| Removal of outliers |  | ✓ |  |
| Transformation of attributes (e.g. normalization) |  | ✓ |  |
| Imbalanced class distribution (e.g. airline sentiment) | ✓ |  |  |
| Change attribute type | ✓ |  |  |

At the start of the exploratory phase, the dataset was loaded and checked for each type of variable. All 15 variables started out as character datatypes; of the 15 variables, eight of them were converted so their appropriate data type including the class variable *airline sentiment* to factor. Some of the variables such as *negative reason* and *negative reason gold* were only applicable to the records that contain were classified as negative, hence a lot of the variables had blank entries. Likewise, there were other variables such as *tweet coordinates* and *tweet location* that had missing values such that the information was not available. As a result, all missing values (blanks or not available) were replaced with NAs. Records with NA values will be kept in the dataset as the majority of records have at least one NA across the variables removing them will result in a limited sample size. Table 2 summarizes the data type of each variable and the number of NAs within each variable. Similarly, variables that are unlikely to be useful in the sentiment analysis will be retained to maximize the variables available during feature selection.

**Table 2. Variable meta data and descriptive statistics of the Twitter airline dataset.**

|  |  |  |
| --- | --- | --- |
| Variable | Data Type | NAs |
| tweet\_id | Numeric | 0 |
| airline\_sentiment | Factor | 0 |
| airline\_sentment\_confidence | Numeric | 0 |
| negativereason | Factor | 4863 |
| negativereason\_confidence | Numeric | 3737 |
| airline | Character | 0 |
| airline\_sentiment\_gold | Character | 7347 |
| name | Character | 0 |
| negativereason\_gold | Factor | 7355 |
| retweet\_count | Integer | 0 |
| text | Character | 0 |
| tweet\_coord | Character | 0 |
| tweet\_created | Date | 0 |
| tweet\_location | Character | 0 |
| user\_timezone | Character | 0 |

The descriptive metrics including min, max, median, range, and standard deviation of *airline sentiment confidence, negative reason confidence,* and *retweet* count, the numerical variables, were also calculated and summarised in Table 3. It is worth noting that these metrics were not calculated for *tweet ID* as it is the identification variable. Additionally, boxplots were created (Figure 1) to visualise the spread of the data and identify any outliers. The *retweet* *count* box plot signal that there are many outliers in this variable which can be expected as many of the records have a value 0. The categorical variables *airline, airline sentiment, airline sentiment gold, negative reason gold,* and *negative reason* were charted into bar charts visualise their frequency (Figure 2). From these charts, we can see that the class distribution is imbalanced with a skew towards negative sentiments. To balance the data, the negative and neutral groups were undersampled so the working dataset is broken out approximately by 33% per group. Furthermore, *airline sentiment gold* and *negative reason* *gold* primarily consist of NAs, so these are unlikely to be useful for the sentiment analysis and therefore likely not to be included in the model.

**Table 3. Summary of numeric variables with outliers.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Airline Sentiment Confidence | Negative Reason Confidence | Retweet Count |
| Min | 0.3353 | 0 | 0 |
| Median | 1 | 0.652 | 0 |
| Max | 1 | 1 | 44 |
| Range | 0.6650 | 1 | 44 |
| Mean | 0.8768 | 0.506 | 0.099 |
| SD | 0.1745 | 0.4500 | 0.7609 |

**Figure 1: Boxplot charts of the airline sentiment confidence, negative reason confidence, and retweet count to identify the presence of outliers.**

**Chart, box and whisker chart

Description automatically generated**

**Chart, bar chart, waterfall chart

Description automatically generatedFigure 2. Bar charts of the categorical variables including Airline Sentiment, the class variable of the sentiments from the tweets.**

Chart, histogram

Description automatically generated

The following R functions were used in the data exploration and preparation steps:

* summary(): display the descriptive metrics of each variable included
* str(): display the structure of the data frame and in the individual variables
* sample(): sample the data randomly to reduce the sample size to 5000 records
* as.date(): convert tweet\_created to date from character type
* as.factor(): convert airline sentiment from character to factor type to properly
* is.na(): check for any NAs in the dataset
* Several ggplot functions including geom\_bar() for the bar charts and geom\_box() for the boxplots

# Methodology

**Figure 3. Proposed methodology to perform sentiment analysis on the Twitter airline data**

Diagram

Description automatically generated

## Feature Selection and Preprocessing

### Attribute Selection

In NLP, the needed variables are the ones that contain the class (*airline sentiment*) and text (*text*). These are the two variables that were used for the first stage of this project where the text is processed statistically and semantically.

### Text Preprocessing

For the text preprocessing stage, the tm library in R was used. Tm provides a wide array of functions including content\_transformer(), removePunctuation, and removeNumbers that eases the text processing phase to prepare the data for modelling. A custom function was also generated to remove extra elements that are unique to tweets.

**Lowercase/Uppercase Conversion**

Computer algorithms are often case-sensitive, that is, the machine reads similar words with different cases as different words. As an example, “Happy” and “happy” will be counted as two different words instead of one – “happy”. To minimize duplication and reduce the size of the corpus, it is best practice to convert the text into all lowercase or all uppercase before running any models. The use of lowercase or uppercase is user-preference; both will output the same result.

**Punctuation and Special Characters Removal**

Classifiers consider punctuation as words; however, punctuations are often eliminated as they typically do not contribute significantly to the model especially in statistical tests. Unlike words, punctuation is not as useful on their own as they do not convey as much sentiment without any words associated to them. As a result, punctuation must also be removed. In this study, since the text used is tweets, sentiments can be expressed through emojis and smiley faces which are a combination of punctuation (e.g., :)). There are a variety of R packages that could be used that match them to their corresponding words and thus can be used in the corpus. For the scope of this study, emojis were removed; however, a study can be taken further to improve the algorithms by keeping the emoticons.

**Numbers Removal**

Like punctuation, numbers are also considered as words by classifiers. The removal of numbers from the corpus needs to be done carefully as numbers can contribute to the model. This is especially true when the data is tokenized semantically as it provides additional information and emphasize the sentiment (e.g. “called 100x” is different from “called 1x”). There are also packages in R that can be utilized so machine learning algorithms can read numbers more effectively and can therefore contribute more meaningfully to the model. Numbers are removed for this particular study to minimize the words in consideration.

**URLs, Hashtags, and Mentions Removal**

Tweets are a unique corpus as they include more parts than usual documents such as books, reviews, and articles. They often include URLs, hashtags, mentions, and special characters (such as aforementioned emojis). Hashtags are used in Twitter to tag a tweet for convenient searching and associating to the same set of ideas (such as movements like “#metoo”). Mentions are a way for the user to tag another user (individuals or companies) to draw their attention to a specific tweet – these are identified by an “@” sign followed by the username. All these parts of a tweet could be read by the classifier as words, but they do not add additional information for the purpose of sentiment analysis.

**Stop Words**

Stop words are commonly used words such as “a”, “the”, “and” which often result in noise in the data and model. For this study, the names of the airlines were included in the dictionary of stop words as they do not contribute to the sentiment analysis.

**Stemming/Lemmatization**

Stemming and lemmatization reduce words to their basic forms to reduce the complexity and size of the corpus. Stemming is a cruder process where prefixes and suffixes are removed so only the root words is left, while lemmatization cross references the word to a dictionary of words to identify and match the words that are the same. As an example, words such as “likely”, “likes”, and “liked” are all reduced to “like” by stemming; “are”, “am”, and “I” are all converted to “be” through lemmatization.

## Feature Engineering and Model Building

After the data was cleaned and pre-processed, features for the data frames can be determined through various techniques: TF-IDF, Bag-of-Words, and N-gram. During this phase, the document-term matrices were created then reduced to remove words that do not contribute to the model as they have 0 instances in the documents. Alternatively, term-document matrices could be created – the results will be the same as TD matrices are transposed DT matrices.

**TF-IDF Weighting**

The Term Frequency – Inverse Document Frequency (TF-IDF) is a measure used to evaluate the importance of a word is to each document. A document-term matrix is determined then weights are applied for each term in a document. The weights assigned are calculated through the following formula:

TF-IDF returns high scores to words with low frequency, and when both parts (TF, IDF) individually have high scores. The *weightTfIdf* function from the tm package was used to apply the weights in our models.

Below is a table of a subset of documents and terms. Each entry in the table are the weights that are applied to each word within each document.

**Table 4. Subset of document-term matrix using the TF-IDF weights technique.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DOCS** | **even** | **ever** | **fli** | **just** | **never** | **bag** |
| 15 | 0 | 0 | 0 | 0.5671 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0.5612 | 0.3933 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0.3054 | 0 | 0.3759 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bag-of-Words Model**

The Bag-of-Words (BOW) represents the occurrence of words within a document. It counts the number of words and disregards any grammatical details and word order. Unlike higher value n-grams (e.g. bigrams and trigrams), the order and structure of the words are completely ignored by the BOW model and only accounts for the presence of specific words. Like TF-IDF, a document or term frequency matrix is created where each word is compared to a given dictionary of words. These matrices are then converted into vector for “scoring”. Once all vectors are created, the words are then scored based on their frequency.

Below is a subset of the document-term matrix for the BOW model. Unlike the TF-IDF weights document-term matrix, the entries are the sum of the instance of each word within the documents.

**Table 5. Subset of document-term matrix using the TF-IDF weights technique.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DOCS** | **even** | **ever** | **fli** | **just** | **never** | **bag** |
| 15 | 0 | 0 | 0 | 1 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 1 | 1 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 1 | 0 | 1 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

**N-gram**

N-gram is a sequence of *n* items from a sample of text or speech. Some of the more common n-gram types include unigram (n=1) and bigram (n=2). The machine assumes that each word is independent of one another in unigrams; for bigrams, pairs of words are identified that will help with sentiment analysis.

A subset of the document-term matrix for the unigram can be found below. Similar to the BOW model, the entries are the total count of the word within each document.

**Table 6. Subset of document-term matrix using the TF-IDF weights technique.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **DOCS** | **even** | **ever** | **fli** | **just** | **never** | **bag** |
| 15 | 0 | 0 | 0 | 1 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 1 | 1 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 1 | 0 | 1 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 4. Word clouds created after the data was tokenized by the TF-IDF weights, BOW, and unigram techniques, respectively.**

A picture containing scatter chart

Description automatically generated

**Classifiers**

Three supervised machine learning classifiers are examined in this project, namely the decision tree, random forest, and Naïve Bayes algorithms. Decision trees predict the value of a target by learning basic decision rules. DTs are one of the basic classifiers and is often a weaker algorithm. To improve results, the random forest was also examined as it is made of many decision trees. The Naïve Bayes classifier applies Bayes’ theorem of probability of one event’s relation to another event to classify. To evaluate the performance of each classifier on different model, all classifiers were applied on all three models, resulting in nine combinations.

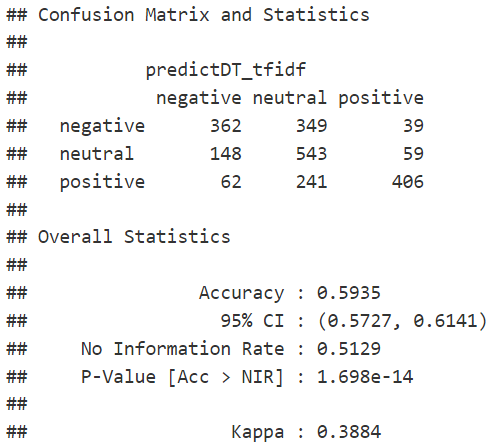
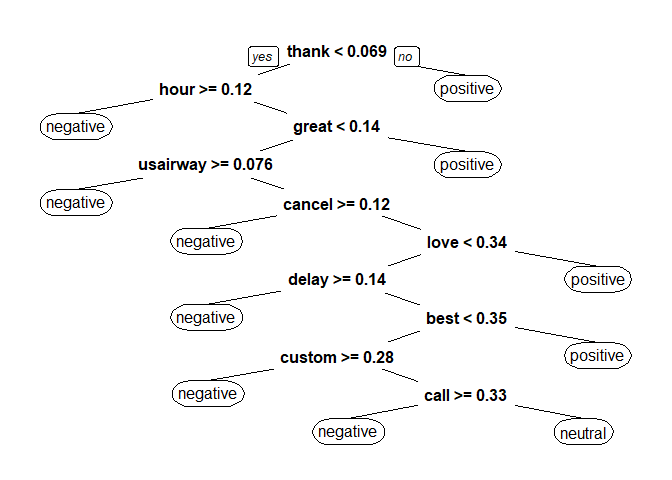
# Results

### Train/Test split

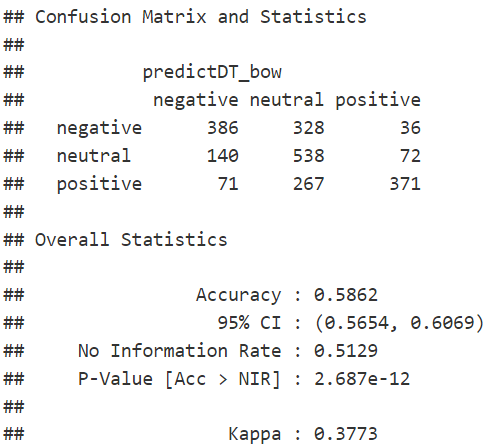
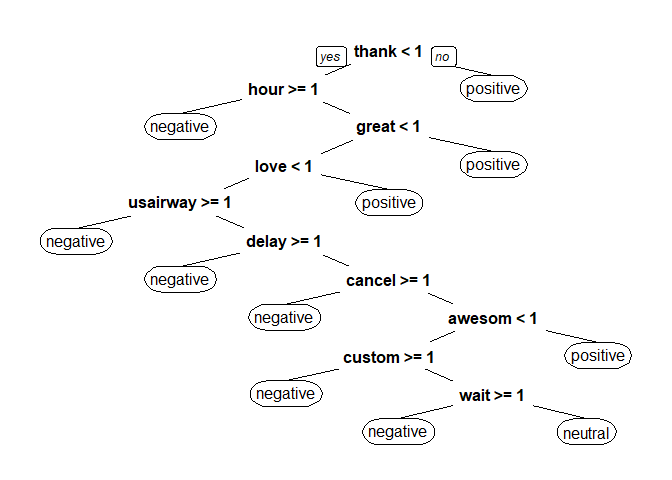
To create a baseline for the next iterations of testing, an initial test was performed for each combination of technique and algorithm using the Train/Test split method. 70% of the data was used of training and the remaining 30% reserved for testing. No limitations were implemented to the decision tree or random forest, that is, the trees were allowed to grow without specific pruning parameters nor number of decision trees within the random forest. Figures 4-7display the decision trees and confusion matrix for each technique.

For all techniques, “thank”, “hour”, and “great” were the starting nodes to the decision trees. Deviations to the trees occur from the fourth level node and onwards. The accuracy scores are comparable at 0.5935, 0.5852, and 0.5862 for TF-IDF, BOW, and unigram, respectively.

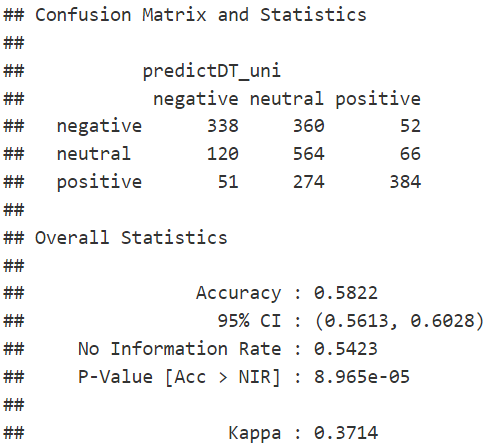
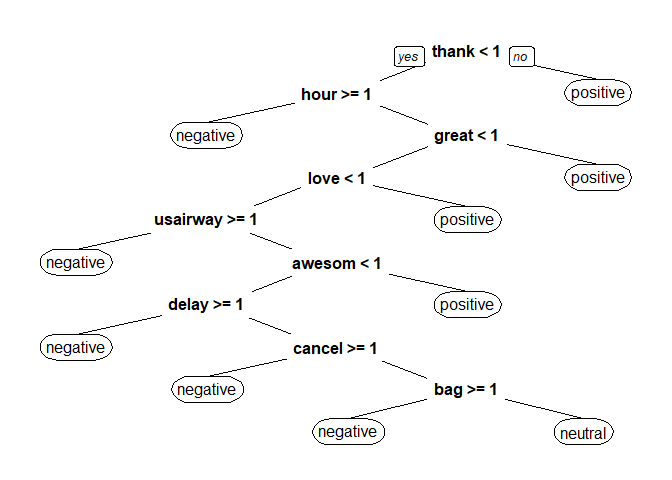
**Figure 5. Decision tree, confusion matrix and statistics for the TF-IDF model**



**Figure 6. Decision tree, confusion matrix and statistics for the BOW model**



**Figure 7. Decision tree, confusion matrix and statistics for the unigram model**



### K-Fold Cross Validation

For the second attempt, the data underwent k-fold cross validation with some parameter changes to the algorithms, especially for the random forest classifier. The random forest was tested at 10 trees initially. The k-fold cross validation method selects a subset of the dataset to use for testing while training with all other values, *k* times. Table 7 and 8display the accuracy scores for each fold for each classifier for 5-folds and 10-folds, respectively. The individual folds were tested for significant differences using the Kruskal-Wallis test and all pairs returned insignificant, that is, based on accuracy, the classifiers all performed the same for all classifier/technique combinations.

Within the k-fold cross-validation process, the *accuracy* with the highest value was used to select the optimal model. Like the Train/Test split method, once the cross validations were performed, confusion matrices were created to report the performance of each classifier for each of the models. Figures 8 and 9 display the individual matrices for the classifiers with the optimal model for 5-fold and 10-fold cross-validations, respectively.

#### 5-Fold Cross Validation

**Figure 8. Confusion matrices for the different classifier and models using the 5-fold cross-validation method to train and test.**



**Table 7.** **Accuracy scores for the 5-fold cross validation test for each technique/algorithm combination.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Decision Tree** | | | **Random Forest** | | | **Naïve Bayes** | | |
| **Fold** | **TF-IDF** | **BOW** | **Unigram** | **TF-IDF (10)** | **BOW (10)** | **Unigram (10)** | **TF-IDF** | **BOW** | **Unigram** |
| 1 | 0.489 | 0.553 | 0.548 | 0.616 | 0.659 | 0.652 | 0.571 | 0.643 | 0.621 |
| 2 | 0.526 | 0.545 | 0.498 | 0.636 | 0.644 | 0.662 | 0.559 | 0.613 | 0.641 |
| 3 | 0.553 | 0.540 | 0.554 | 0.650 | 0.636 | 0.637 | 0.566 | 0.627 | 0.597 |
| 4 | 0.499 | 0.511 | 0.494 | 0.629 | 0.623 | 0.621 | 0.541 | 0.616 | 0.651 |
| 5 | 0.497 | 0.479 | 0.490 | 0.657 | 0.633 | 0.665 | 0.572 | 0.600 | 0.606 |
| Optimal model accuracy | 0.513 | 0.526 | 0.515 | 0.637 | 0.639 | 0.647 | 0.561 | 0.620 | 0.623 |

#### 10-Fold Cross Validation

**Figure 9. Confusion matrices for the different classifier and models using the 5-fold cross-validation method to train and test.**



**Table 8. Accuracy scores for the 10-fold cross validation for each technique/algorithm combination.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Decision Tree** | | | **Random Forest** | | | **Naïve Bayes** | | |
| **Fold** | **TF-IDF** | **BOW** | **Unigram** | **TF-IDF (10)** | **BOW (10)** | **Unigram (10)** | **TF-IDF** | **BOW** | **Unigram** |
| 1 | 0.491 | 0.537 | 0.510 | 0.626 | 0.613 | 0.654 | 0.534 | 0.620 | 0.640 |
| 2 | 0.537 | 0.477 | 0.542 | 0.654 | 0.644 | 0.667 | 0.545 | 0.578 | 0.649 |
| 3 | 0.537 | 0.543 | 0.556 | 0.658 | 0.624 | 0.643 | 0.577 | 0.610 | 0.601 |
| 4 | 0.572 | 0.483 | 0.491 | 0.626 | 0.662 | 0.639 | 0.577 | 0.635 | 0.630 |
| 5 | 0.496 | 0.486 | 0.488 | 0.645 | 0.639 | 0.645 | 0.548 | 0.635 | 0.621 |
| 6 | 0.552 | 0.488 | 0.492 | 0.643 | 0.645 | 0.635 | 0.544 | 0.611 | 0.618 |
| 7 | 0.535 | 0.551 | 0.481 | 0.650 | 0.637 | 0.648 | 0.549 | 0.636 | 0.617 |
| 8 | 0.499 | 0.565 | 0.486 | 0.650 | 0.674 | 0.671 | 0.583 | 0.624 | 0.594 |
| 9 | 0.503 | 0.564 | 0.540 | 0.606 | 0.660 | 0.635 | 0.562 | 0.611 | 0.681 |
| 10 | 0.543 | 0.486 | 0.500 | 0.640 | 0.599 | 0.634 | 0.573 | 0.625 | 0.584 |
| Optimal model accuracy | 0.526 | 0.518 | 0.509 | 0.640 | 0.640 | 0.647 | 0.559 | 0.618 | 0.624 |

### Other metrics

In addition to accuracy, the confusion matrices created using the Train/Test split and k-fold cross validation, were used to calculate the specificity and sensitivity of the models. Sensitivity, also known as recall, is the *true positive* rate of the considered class while specificity is the *true negative* rate of the considered class. Figure 10 displays the specificity scores of each classifier/model combination while figure 11 shows the sensitivity scores.

Based on these graphs, the decision tree returns the highest specificity scores for the neutral class across all three models for cross-validated trained and tested data. While negatives show more variation across all combinations, the highest specificity results from the Naïve Bayes classifier applied in BOW and unigram models using the 70%/30% train/test split. The specificity for the positive class also shows variations but scores highest in when the random forest classifier is applied, regardless of model for both 10-fold CV and train/test split.

On the other hand, sensitivity for the positive class is

**Figure 10. Scatterplot of specificity scores of each class within each classifier/model combination by train/test split method.**

Calendar

Description automatically generated

**Figure 11. Clustered bar chart of sensitivity scores of each class within each classifier/model combination.**

Calendar

Description automatically generated

### Additional tests

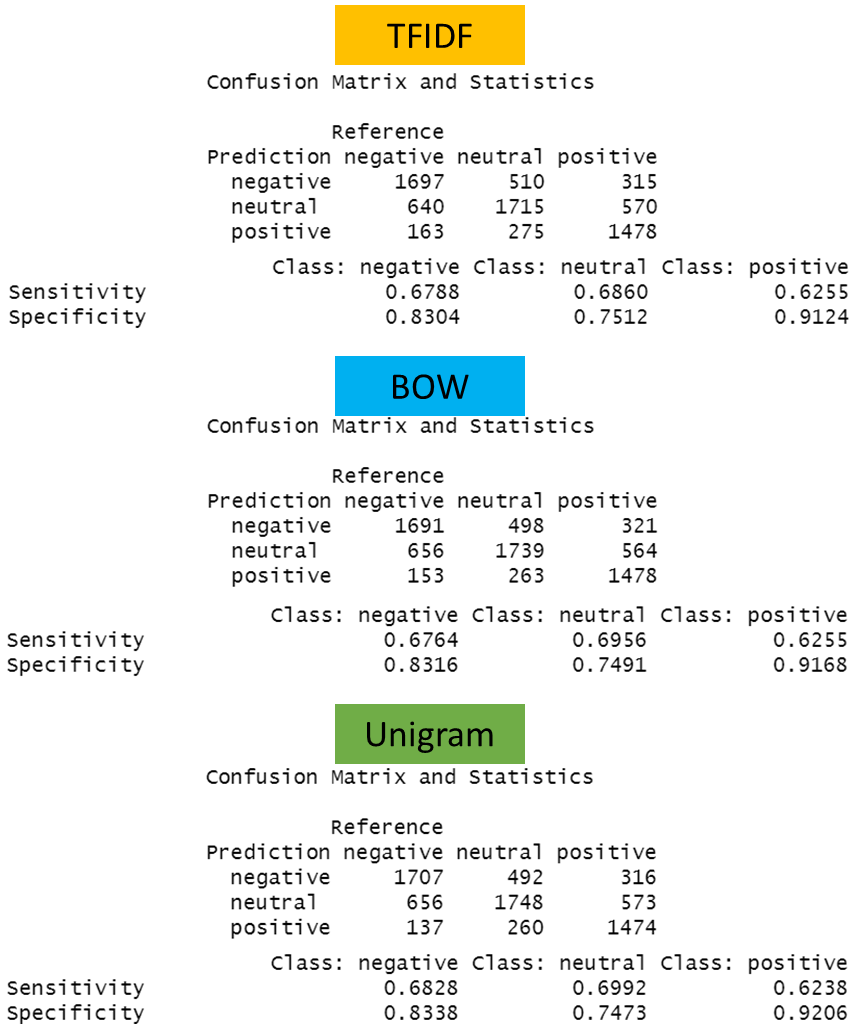
Identifying the best machine learning algorithm involves tuning numerous parameters and multiple testing. All tests that were done so far were done on a basic level, that is, at a minimum with little limitation to explore what is feasible for these classifiers. This, however, resulted in lower accuracy scores so the classifiers were tuned to improve accuracy and result in higher F-scores.

#### Random Forest: Trees

For the random forest algorithm, the number of trees was increased from 10 to 50 trees. The process of identifying the optimal number of trees for a random forest classifier can be difficult and time-consuming as it is an iterative process. There are packages in R that assist with this process; however, due to processor limitations, this project maxes at 50 trees at 10-fold cross validation. Table 9 outlines the individual accuracy scores for each model, given the number of trees is different.

There is an overall increase in accuracy when the number of trees is increased. We see that in the optimal model, the accuracy is higher, regardless of the tokenization technique applied to the corpus. The BOW technique results in the highest difference with a 0.03 points increase going from 10 trees to 50 trees.

**Figure 12. Confusion matrices and metrics for 50 trees in the random forest classifier for the TF-IDF weights, BOW, and unigram models.**



**Table 9. Accuracy scores for the 10-fold cross validation for each technique.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Random Forest** | | | | | |
| **Fold** | **TF-IDF (50)** | **BOW (50)** | **Unigram (50)** | **TF-IDF (10)** | **BOW (10)** | **Unigram (10)** |
| 1 | 0.643 | 0.699 | 0.669 | 0.631 | 0.616 | 0.634 |
| 2 | 0.646 | 0.673 | 0.652 | 0.638 | 0.640 | 0.657 |
| 3 | 0.664 | 0.658 | 0.669 | 0.668 | 0.630 | 0.632 |
| 4 | 0.644 | 0.652 | 0.652 | 0.634 | 0.649 | 0.643 |
| 5 | 0.671 | 0.666 | 0.655 | 0.647 | 0.637 | 0.645 |
| 6 | 0.700 | 0.645 | 0.671 | 0.638 | 0.636 | 0.621 |
| 7 | 0.639 | 0.669 | 0.684 | 0.639 | 0.635 | 0.635 |
| 8 | 0.697 | 0.667 | 0.675 | 0.636 | 0.664 | 0.665 |
| 9 | 0.664 | 0.677 | 0.685 | 0.617 | 0.658 | 0.645 |
| 10 | 0.674 | 0.662 | 0.672 | 0.636 | 0.607 | 0.626 |
| Optimal model accuracy | 0.664 | 0.667 | 0.669 | 0.640 | 0.637 | 0.647 |

#### Bigram

Unlike unigrams where words are split into individual words, bigrams identify and consider pairs of words that result in splits with different meanings. As an example, “New York” as a unigram will be the words “new” and “York” which are different meaning from the intended term which is the state in the United States.

Since the random forest classifier returned high accuracy scores for unigrams, bigrams were also tested using this classifier. Similar to the other random forest tests, the bigram was also tested with different parameters: train/test splitting techniques and number of trees. With the train/test split technique, the accuracy returned was 0.6786 which is better than the 10-fold cross validation accuracy scores for 10 trees and 50 trees of 0.651 and 0.665, respectively. Against the unigram, the bigram performed slightly worse for both 10 and 50 trees.

# Conclusion and Future Work

The project explored the topic of Natural Language Processing (NLP) in the R language using various classifiers (decision tree, random forest, and Naïve Bayes) and tokenization techniques including TF-IDF, BOW, and n-grams.

Some challenges and limitations were encountered throughout the project, especially within the computer processing power. Running the classifiers with no limits set to the parameters resulted in long run times so the classifiers were restricted. This resulted in tests that may not be optimized to the model such as performing tests with the ideal k number of folds and running a model with the ideal number of tress in random forest. For future revisions of this study, additional tuning techniques should be applied for a more robust. For example, prune individual trees within the random forest.

# References

Arun, K., Srinagesh, A., & Ramesh, M. (2017). Twitter Sentiment Analysis on Demonetization Tweets in India Using R Language. *International Journal of Computer Engineering and Information Technology*, *9*(6), 119–124. Retrieved January 30, 2022, from https://www.proquest.com/openview/8161cd322db72b061e2f4cc27cae3bfc/1?pq-origsite=gscholar&cbl=2044551.

Bayhaqy, A., Sfenrianto, S., Nainggolan, K., & Kaburuan, E. R. (2018, October). Sentiment analysis about E-commerce from tweets using decision tree, K-nearest neighbor, and naïve bayes. In *2018 international conference on orange technologies (ICOT)* (pp. 1-6). IEEE.

Eight, F. (2019, October 16). *Twitter us airline sentiment*. Kaggle. Retrieved January 20, 2022, from https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Isabella, J., & Suresh, R. M. (2011). Opinion mining using correlation based feature selection. *PsycEXTRA Dataset*. https://doi.org/10.1037/e605252012-003

Jatnika, D., Bijaksana, M. A., & Suryani, A. A. (2019). Word2Vec model analysis for semantic similarities in English words. *Procedia Computer Science*, *157*, 160–167. https://doi.org/10.1016/j.procs.2019.08.153

Kouloumpis, E., Wilson, T., & Moore, J. (2021). Twitter Sentiment Analysis: The Good the Bad and the OMG!. Proceedings of the International AAAI Conference on Web and Social Media, 5(1), 538-541. Retrieved from https://ojs.aaai.org/index.php/ICWSM/article/view/14185

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, *37*(2), 267–307. https://doi.org/10.1162/coli\_a\_00049

Twitter (2021) *Q3 2021 Letter to Shareholders* [Press Release]. https://s22.q4cdn.com/826641620/files/doc\_financials/2021/q3/Final-Q3'21-Shareholder-letter.pdf

Verma, S., Vieweg, S., Corvey, W., Palen, L., Martin, J., Palmer, M., Schram, A., & Anderson, K. (2021). Natural Language Processing to the Rescue? Extracting "Situational Awareness" Tweets During Mass Emergency. Proceedings of the International AAAI Conference on Web and Social Media, 5(1), 385-392. Retrieved from https://ojs.aaai.org/index.php/ICWSM/article/view/14119

Withdrawal of Legal Tender Status for ₹ 500 and ₹ 1000 Notes: RBI Notice. (2016). *Reserve Bank of India*. Retrieved January 31, 2022, from https://rbi.org.in/Scripts/BS\_PressReleaseDisplay.aspx?prid=38520.