Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

Sentiment Analysis of Tweets Using Supervised Machine Learning Techniques Based on Term Frequency

Deepti Aggarwal*

*Corresponding Author, Assistant Professor, JSS Academy of Technical Education, Noida. E-mail: Aggarwal.deepti@gmail.com

Vikram Bali

Professor, JSS Academy of Technical Education, Noida. E-mail: vikramgcet@gmail.com

Abhishek Agarwal

JSS Academy of Technical Education, Noida. E-mail: abhishek.agrawal0297@gmail.com

Kshitiz Poswal

JSS Academy of Technical Education, Noida. E-mail: kshitiz.poswal@gmail.com

Madhav Gupta

JSS Academy of Technical Education, Noida. E-mail: gupta.madhav951@gmail.com

Abhishek Gupta

JSS Academy of Technical Education, Noida. E-mail: guptaabhi9650@gmail.com

Abstract

World of technology provides everyone with a great outlet to give their opinion, using social media like Twitter and other platforms. This paper employs machine learning methods for text analysis to obtain sentiments of reviews by the people on twitter. Sentiment analysis of the text uses Natural language processing, a machine learning technique to tell the orientation of opinion of a piece of text. This system extracts attributes from the piece of writing such as a) The polarity of text, whether the speaker is criticizing or appreciating, b) The topic of discussion, subject of the text. A comparison of the work done so far on sentiment analysis on tweets has been shown. A detailed discussion on feature extraction and feature representation is provided. Comparison of six classifiers: Naïve Bayes, Decision Tree, Logistic Regression, Support Vector Machine, XGBoost and Random Forest, based on their accuracy depending upon type of feature, is shown. Moreover, this paper also provides sentiment

analysis of political views and public opinion on lockdown in India. Tweets with '#lockdown' are analysed for their sentiment categorically and a schematic analysis is shown.

Keywords: Feature representation; TFIDF; N-grams; Pre-processing; Tokenization; Word Cloud.

DOI: 10.22059/jitm.2021.80028

Introduction

Sentiment Analysis checks whether the written text is a positive, a negative or a neutral statement. The basic idea is to analyse text using natural language processing (NLP) and tell whether a piece of writing pertains to criticism or appreciation. Sentiment analysis therefore, helps data analysts of large organisations to understand and generate insights from public opinion. Analysts conduct modulated market research to monitor brand and product reputation among the public. Sentiment analysis uses machine learning techniques in which a model is trained and tested and the model is fitted for using it for further processes.

Twitter is a popular social media where people mention their opinions a lot. It enables the interaction of users with the whole world using messages known as tweets. Analysts and researchers have moved their attention towards social networking websites as the decision makers rely on statistics such as summarised opinions of people which is easily obtained by performing analysis on social media data. The reasons for which we concentrate on twitter are its popularity, tweets reflect the instantaneous opinions of people regarding a topic. Another reason is that varied topics get attention on it whether it is political, religion, sports etc. Also, people from every class of society have given attention to it whether it is business leaders, politicians or a clerk. The data from twitter is obtained using its API for python, which provides a json data consisting of tweet text, username, and tweet id. Sentiment analysis of text documents is done gradually in stages. It involves the following steps:

1. Each text document is broken down into its elements (phrases, words, sentences, tokens and parts of speech)
2. We identify elements bearing sentiment.
3. We assign a score representing the sentiment to each phrase and component (-1 to +1)
4. We can also combine scores for multi-layered sentiment analysis

In contrast with standard text-based categorization in which mere presence of words is indicative of category, sentiment analysis breaks down into components such as adjectives which help in polarity categorization. Sentiment of a word is with respect to the domain for example "unpredicTable" can be used for aftermaths of a newly implemented law. Sentiment analysis has challenges similar to emotion recognition problem that is deciding what we understand by sentiment of a given sentence. Is it categorical, and sentiment can be characterised as happy, sad, angry, or bored? Or is it dimensional, and sentiment needs to be evaluated on a multi-directional spectrum? In addition to the problem of defining sentiment, there are multiple layers of meaning to a text generated by humans. People use rhetorical devices to express opinions, like sarcasm, irony, and literal meaning of text can mislead sentiment analysis. The only way to really understand these devices are through context. But as we consider twitter people refer to topics and situations without properly referencing to them or writing about them in the text. This makes it a challenge to understand the context of text and hence generating true sentiment (Aggarwal, 2018).

Most of the current work in sentiment analysis is done using the definition that it is categorical. Sentiment is analysed as belonging to a certain class or category, to a certain degree. For example, a given sentence may be 55% happy, 23% sad, 79% delighted, and 52% optimistic. They don't add up to hundred, they're individual indications of what degree a sentence's sentiment is. So we need to specify ourselves on what threshold value of degree of a type of sentiment we assign the text as being positive or negative.

The last challenge is to decide on what parameters (frequency of elements or simply the presence of them) we should train the model we'd like to use. There are a number of pre-trained models available for use in popular Data Science languages. For example, TextBlob, which offers a simple API built in python that can be used to generate polarity index of sentiment for a text, and the Syuzhet package built in R provides a simple API for sentiment analysis in R (Mäntylä, Graziotin & Kuutila, 2018). These modules can help us evaluate your models, but for the best results of analysing sentiment for unknown data, we want to train our own models and test with hand categorized data (Doaa & Hussein, 2018).

Related Work

Fan et al. (2016) introduces a paper having a title Apply Word Vectors for Sentiment Analysis of APP Reviews using techniques like Stanford Tokenizer, for tokenization of given corpus dataset with 85% Accuracy.Word2vec can only improve macro average precision and macro average recall for sentiment analysis of short text. While using Word2Vec, the number of sentiment seeds has no influence on the building of sentiment lexicon. While comparing with other sentiment lexicons, that include lexicon of National Taiwan University and the lexicon built by PMI, word2vec has greater precision and recall rate. The building of lexicon by

word2vec is easier than the building of lexicon by PMI. Saif et al. (2016) introduces a paper with a title Contextual Semantics for Sentiment Analysis of Twitter using techniques like Senti Circles with 72.39% Accuracy, A lexicon- based approach that takes into account the contextual and conceptual semantics of words when calculating their sentiment orientation and strength. SentiCircles outperformed other lexicon labelling methods for both entity-level and tweet-level sentiment detection. SentiCircles gave better results than SentiStrength in 2 out of 3 datasets. SentiCircles falls marginally behind by 1% in F-measure in one of the datasets.

Alsaeedi & Khan (2019) introduces a paper with a title A Study on Sentiment Analysis Techniques of Twitter Data using techniques Supervised Machine Learning approach, Ensemble Approaches, Lexicon-Based Methods with 80% Accuracy using Machine learning algorithms and with 85% Ensemble and hybrid-based algorithms. Ramanathan & Meyyappan (2019) with a title Twitter text mining for sentiment analysis on people's feedback about Oman tourism using techniques Lexicon based approach using three existing lexicons such as Sentistrength, SentiWordNet and Opinion lexicon. Naïve Bayes machine learning method to increase the performance of sentiment analysis Domain-specific analogy, express positive opinion about Oman tourism. Effect of only 4 factors is considered. Giachanou & Crestani (2016) reviews all the methods available for text classification for classifying tweets for their sentiments. The methods of lexicon-based approach, graph-based approach and machine learning classification model-based approach are reviewed. Although at times for a particular chunk of dataset lexicon-based approach performs better than machine learning classification-based approach, Lexicon based is less explored as compared to Machine learning techniques, so the precision and recall ratio is not suitable for shuffling the data.

Hasan et al. (2018) gives a comparison of techniques of sentiment analysis in the analysis of political views by applying supervised machine-learning algorithms such as Naïve Bayes and support vector machines (SVM). He uses sentiment lexicons such as W-WSD, SentiWordNet, TextBlob and validates them with test results from machine learning classifiers. Other classifiers such as tree based were not considered which might perform better, the comparison is not complete for context of supervised machine learning algorithms as all classification models are not compared (Kumar & Jaiswal, 2020). Salinca (2015) introduces a paper Business reviews classification using sentiment analysis using techniques Linear SVC and Naïve Bayes with 94.4 Accuracy. Time complexity is less as compared to another algorithm as we are using Naïve Bayes. If number of features is much greater than the number of samples, avoid over-fitting in choosing. Summary of the related work is provided in Table 1.

Table 1. Summary of Related Work

Title	Author	Techniques	Merits	Demerits
Real-Time Bag of words	J. Uijilings et al (Uijlings, Smeulders & Scha, 2009).	Descriptors SIFT and SURF, Random Forest	Discovered Pipelines with maximum accuracy.	The calculation of chi-square needs GPU support.
SoMaJo: State-of-the-art tokenization for German web and social media texts	T. Proisl et al (Proisl & Uhrig, 2016).	SoMaJo, a rule based tokenizer available as free software.	1. SoMaJo shared task on automatic linguistic annotation of social media. 2. It is easy to maintain and adapt.	Even after fixing obvious errors, there were 8 false positives due to rare and unsystematic problems.
Analyzing sentiments expressed on twitter by UK Energy Company consumers	V. Ikoro et al (Ikoro, Sharmina, Malik & Batista-Navarro, 2018)	Senti circles, lexicon analysis	A substantial difference in terms of topics being discussed in tweets	High frequency of words have the wrong polarity
Differential privacy-protecting K-means clustering algorithms	Y. Zhang et al (Zhang, Liu & Wang, 2018).	K-means	Avoids deviation of the center Point caused by too large random noise	Availability of the clustering results when the cluster is small can be improved.
Text Mining : Open Source Tokenization Tools – An analysis	Dr. S.Vijayarani et al (Vijayarani & Janani, 2016).	Tokenization, Text mining	Tells us about the need of tokenization, and then analyses some open source tokenization tools such as NLTK.	It is difficult to tokenize the document without any whitespace, special characters or other marks.

Data Description

Twitter is a social network on which people express themselves rather informally, not using proper English grammar. This is due to casual behaviour of people on social media. Twitter restricts users to write 140 characters in a tweet. Hence a single tweet can be considered as a sentence. Due to nature of this social network (short and precise message) people tend to make grammatical mistakes, spelling errors, use emoticons, type using lingos etc (Kumar & Jaiswal,2020). We have acquired a dataset of 122,750 tweets from Kaggle which was available to participants under a competition. The twitter does not allow to store tweets locally, hence the dataset consists of tweet ids and user name which has posted it. Using twitter API, we have extracted the exact tweets using their tweet ids to make the corpus of data. These tweets are hand classified for their sentiment as positive, negative. Some tweets are marked as 'junk', which indicates that the person has not expressed any sentiment rather has only shared a web link or photo. We remove these tweets marked as 'junk', as we need to perform text classification. Finally, we have a corpus of 10,753 tweets which are marked as being positive or negative with values '1' or '0'.

Pre-processing of Data

Raw tweets scrapped from twitter produce noisy data. Tweets often contain special characteristics such as retweets, emoticons, hashtags, user mentions, URLs etc. The raw tweet data has to be converted into a normalised version of text in only English. For the task we use various regular expressions to remove such characters which do not lie under the umbrella of text classification. We start with some basic pre-processing steps: Conversion to lowercase; Replace dots '.' with spaces; Replace multiple consecutive spaces with single space.

Twitter users often share web links in their tweets. These cannot be considered to be under umbrella of text classification as these link data not necessary symbolises the content it contains. Hence, we need to remove such links for which we use regular expressions. The regular expressions are used to match a certain type of content and when it matches, we can remove it. The regular expression used for URLs matching is ((www\. [\S] +) | (https? ://[\S] +)).

Emoticons are representation of facial expressions using the letters and punctuations. Making a dictionary of emoticons which maps them to their expressions is difficult as now emoticons are available which symbolises nouns, this number is ever increasing. However, we map some emoticons which are purely facial expressions and have been used for about a decade now (Mittal & Patidar, 2019). We replace the emoticons with their expression (an English dictionary word) using a dictionary created of emoticons and corresponding expression. To match the emoticons, we use regular expression and then replace them with corresponding expression. A list of regular expressions used to match emoticons with the English word (their expression) used to replace them is shown in Table 2.

Table 2. Regular expressions used to match the emoticons

Emoticon(s)	Regular Expression (Regex)	Replacement word
:), :), :-), (:, (:, (-:, :')	(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))	smile
:D, : D, :-D, xD, x-D, XD, X-D	(:\s?D|:-D|x-?D|X-?D)	laugh
<3, :*	(<3|:*)	love
:-(, : (, :(,):,)-:	(:\s?\(|:-\(|\)\s?:|\)-:)	sad
:,(, :'(, :"(	(:,\(|:\'\(|:"\()	cry
;-), ;), ;-D, ;D, (;, (-;	(:\s?\(|:-\(|\)\s?:|\)-:)	wink

Twitter users mention other users either to quote their words or to ask for their comments. These user mentions are not at all useful for text analysis nor does it produce any

sense of sentiment. The way other users are mentioned are using @handle, where handle is other user's unique username stored on twitter database. To match the user mention we use regular expression @[\S] + and remove them from tweets. Twitter uses hashtag followed by unspaced string of words to create a topic and thus uses them to show trending topics. Twitter users often use them to relate their tweet with the particular topic. As the topics are mixture of nouns and adjective and can be used for text classification, we remove the symbol of hashtag (#) prefixed to them and use them as it is. Regex used to match is #(\S+). Twitter lets its users to post tweets from other users as it is, which is called retweet option. A retweet starts with letters RT indicating a retweet. We want to use these retweets as it symbolises that many people have same sentiment and agree with tweet of other user. We will just remove the letters RT from tweet using regex which is \brt\b.

Feature Extraction

Features need to be extracted from the corpus so that we can use them for our machine learning models to train them and to predict using them (Aggarwal et al., 2019). For text classification we use words as features, we extract words from tweets as unigrams single words and bigrams group of two words. After the extraction of words, we need to perform stemming and lemmatization on the words before creating frequency distribution of these words. We use library functions for performing these tasks of tokenizing words and performing stemming and lemmatization.

Natural Language Toolkit or NLTK is a library in python which is used for performing various tasks on human language data. There are various suites provided for performing various tasks related to analysis linguistic structure of corpora. The lexical resources provide by Nltk can be used to perform tasks like tokenization, stemming, tagging, parsing, and semantic reasoning. Nltk tokenizer is used to convert the tweets into tokens. Tokenization is the process of dividing a tweet or string into smaller parts. The smaller parts thus obtained are called tokens. The tokenizer iterates through the text word by word. When it encounters a space in a sentence the letters up to the next space that is encountered is considered word or token (Reddy et al., 2019). These tokens are further used for classification. Nltk Tokenizer is used as follows:

```
from nltk.tokenize import word_tokenize words  = word_tokenize(sentence)
```

PorterStemmer under NLTK library is then used for stemming. Stemming reduces words to its root word. Stemming is useful in removing prefixes or suffixes of words to give the root words. These words are morphological variants of the stem words. Stemming reduces redundancy. For example, root of word of "gives", "given", "giving" is give.

Lemmatization brings together words with the same meaning. Lemmatization groups inflected words. Lemmatization is more necessary as it not only removes morphological variants but also removes inflection. This can be better understood with an example. For example, "best: good", "worse: bad" etc. The word "best" is a variant of "good" and the variant of "bad" is "worse". We use WordNetLemmatizer under NLTK library for lemmatization. It uses a built-in function under WordNet to identify morphological variants. The tokens or words which we get are then represented. Text representation is one of the most common issues in text classification. If we talk about one of the most common ways to represent raw text, extracting words or unigram is our way to go.

Unigrams and Bigrams

Unigram consists of a single word sequence. While unigrams are applicable to almost any kind of text, it's been an incomplete option for our classification tasks. Here's where our understanding of bigrams and n-grams comes into play. An n-gram is a sequence of "n" words. We can use n-grams as the basic building block of our model. They are collected from texts or tweets. An n-gram consists of a collection of sequence of n items. When we are using n-grams, special tokens such as _ can be sometimes used to show the beginning and starting of a sentence. But sometimes we need to have a large training dataset, because when the n-grams get larger more context gets added to our words. Bigrams consists of a sequence of two words. Here n is equal to 2 from Latin word "bi". We use bigrams as the sets of other n-grams were not present in more than half of tweets from which n-grams are obtained. For example:

Sentence: " Our algorithm is for sentiment analysis"

Unigrams: ["Our"] , ["algorithm"] , ["is"] , ["for"] , ["sentiment"] , ["analysis"]

Bigrams: ["Our algorithm"] , ["algorithm is"] , ["is for"] , ["for sentiment"] , ["sentiment analysis"]

Frequency distribution is done for unigrams and bigrams where low frequency words are removed. The top_n_words from unigrams and bigrams are collected. The unigrams and bigram words at the end of the corresponding frequency spectrum are noise as they occur very few times in tweets. They will not have a significant effect on classification hence they are removed (Nokel & Loukachevitch, 2015). Top 20 unigrams and their frequencies are shown and plotted on graph (Figure 1). The line of curve approaches towards zero, hence some unigrams have very less frequency, which will not affect our classification. So top_n_words are taken only.

[('i', 50081), ('the', 29777), ('to', 29563), ('you', 26869), ('a', 22065), ('it', 17440), ('and', 16353), ('my', 13779), ('for', 12548), ('is', 12055), ('in', 11807), ('that', 11411), ('im', 11186), ('me', 10980), ('of', 10602), ('have', 9954), ('on', 9621), ('so', 9144), ('but', 9026), ('be', 7435)]

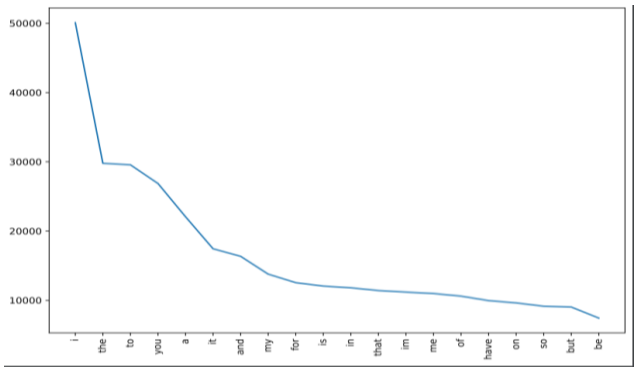
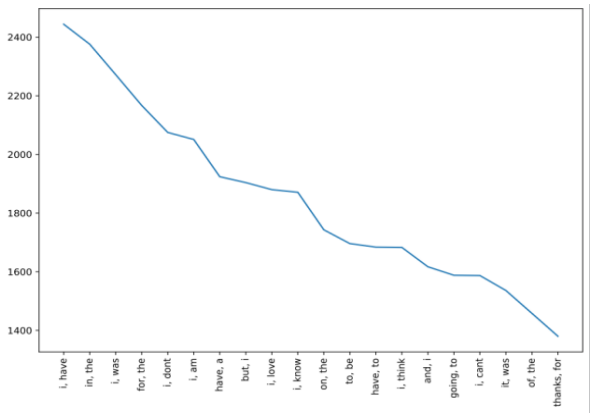


Figure 1. Frequencies of top 20 unigrams

Similarly, the top 20 bigrams and their frequencies are plotted on graph (figure 2).

[(('i', 'have'), 2444), (('in', 'the'), 2376), (('i', 'was'), 2272), (('for', 'the'), 2167), (('i', 'dont'), 2075), (('i', 'am'), 2051), (('have', 'a'), 1924), (('but', 'i'), 1904), (('i', 'love'), 1880), (('i', 'know'), 1871), (('on', 'the'), 1743), (('to', 'be'), 1696), (('have', 'to'), 1684), (('i', 'think'), 1683), (('and', 'i'), 1617), (('going', 'to'), 1588), (('i', 'cant'), 1587), (('it', 'was'), 1536), (('of', 'the'), 1458), (('thanks', 'for'), 1380)]

Figure 2. Frequencies of top 20 bigrams



Feature Representation

To feed data to machine learning models which are statistical models we cannot directly use textual words. For this we need a representation of these features. We represent each tweet as a sparse vector representation or dense vector representation, which depends on the classification model used. For this we create list first for ranking. We use indices of the list as

ranks for unigrams and bigrams. Each unigram or bigram is stored at particular index in list representing its rank which is based on frequency of it in the tweets. We can observe from Figure 3 that the frequency distribution of terms follows Zipf's law. Zipf's law states, if a term x_1 is most frequent term and x_2 the second most frequent, so on, then the frequency of a term cf_i, frequency of i^{th} most frequent term is proportional to 1/i.

$$cf(i) \propto \frac{1}{i} \tag{1}$$

We can obtain an equation by taking a coefficient, C, of proportionality, to state it as

$$log\ log\ cf(i) = log\ log\ C\ + k\ log\ log\ i \tag{2}$$

where k = -1 because of inverse proportionality.

As the Figure 3 shows a negative slope representing Zipf's law, use of indices as ranks based on frequency is justified by the law.

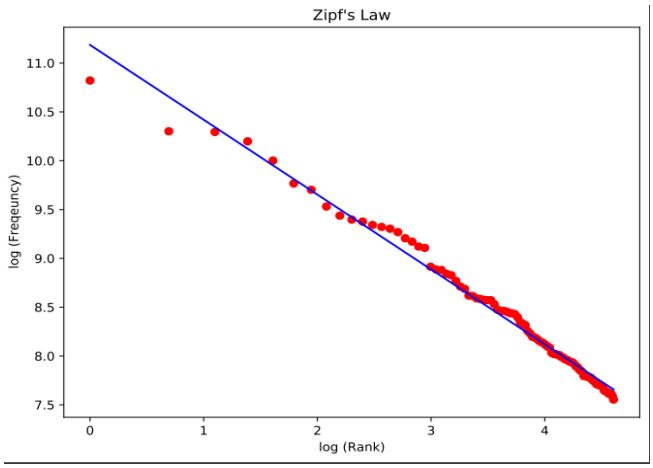


Figure 3. Frequencies follow Zipf's law

Sparse Vector Representation

Each tweet is represented as a feature vector where at the indices (representing unigram or bigram) if that unigram or bigram is present in tweet, a positive number is stored. The positive value is obtained using TF IDF, which assigns the frequency of the word or term and scales it according to the inverse document frequency of word or term, to assign subsequent higher values to terms of more importance.

TF IDF

It is a statistical method to provide a weight to a word or term that signifies the importance of that word or term in the corpus. The weight provided is computed as the number of times the word appears in the document and is scaled according to the frequency in whole corpora.

Here we are considering a single tweet as a document and the whole dataset of tweets as corpora. Variation of tf idf method have been used for ranking in search engine results (Ahuja *et al.*, 2019).

TF (term frequency): frequency of word or term inside a document. As the terms can occur more than once in a document based on its length whether its size is small or long. We generally divide the frequency by document length.

TF(t) = (Frequency of word in document) / (document length based on number of terms)

IDF (Inverse document frequency): Some words appear in every document of corpora which can be prepositions such as 'is', 'of', 'from', 'that'. These words have low importance so we need to weigh them down. This statistical measure gives a measure of how important a term is.

IDF(t) = (Total number of documents/Number of documents having the term t)

Dense Vector Representation

The list of ranking of words or terms is used for creating dense vectors. Each word is stored at a particular index based on its rank, which is a direct representation of its frequency i.e. most common word is assigned number 1, second most common word or term is assigned 2 and so on. Each tweet is represented by a vector of these indices, which forms a dense vector.

Methodology and Techniques

Once the features are extracted and represented as vectors, they can be used for classification purposes. There are multiple classification algorithms and hence to choose the best algorithm a comparison is to be made between their respective accuracies.

Naïve Bayes

Naïve Bayes classifier is a machine learning technique based on probability that is to do the classification of data. It is based on Bayes theorem. The assumption that the features present in a class are independent of each other is the reason why it is called "naïve". Features are the information pieces from the text. An equal share of each feature is there in the outcome of the classifier. For word classification, the Naïve Bayes classifier is a very popular alternative (Aggarwal et al., 2019). Naïve Bayes can be explained with an example.

Let us consider a classification of fruit. In order to classify a fruit as an orange the main two features are its colour and shape. The classification by the machine of a fruit as an orange is done if the color of the fruit is "orange" and the shape is "round". The features are

independent of each other and they also have an equal share in the outcome of the classification. Bayes theorem is based on conditional probability

$$P(A|B) = P(B|A) * P(A)/P(B) \quad (3)$$

Here P(A|B) gives the probability of occurrence of A when B has already occurred.

Multinomial Naïve Bayes

Multinomial Naive Bayes is used for discrete data. The frequency or the count of words is used for the purpose of classification. The algorithm makes use of the probability of occurrence of a value. The probability equates to zero when the value in the account does not occur even once. This when multiplied with other probabilities makes the whole information useless. Hence a pseudo count is introduced in every probability to make sure that none of them is ever zero. Histograms are used to represent feature vectors which comprise of the frequencies of a value.

We use the package of sklearn for classification model, the Multinomial from sklearn. naive Bayes is used for classification. We used the sparse vector representation for performing classification using Multinomial Naïve Bayes. We found that using bigrams along with unigrams increase the accuracy. The best validation accuracy achieved is 77.56% using sparse vector representation with both unigrams and bigrams included. Table 3 shows the precision, recall and f1 score for corresponding accuracy.

Table 3. Evaluation of Naïve Bayes Model using different scores

	Precision	Recall	F1-score	Support
Negative Sentiment	0.74	0.69	0.71	8158
Positive Sentiment	0.79	0.84	0.82	1182
Overall Accuracy			15512/20000	77.56%

Decision Tree

The decision tree is a tree-like model. It is built on a supervised machine learning technique. Supervised learning is a technique in which a machine is first fed output mapped to a certain input as a part of training. The trained data is used to give an output for a new input. Classification, regression, and prediction can be done with decision trees, though not for continuous values. In the tree-like structure, a node represents a test and the possible outcomes of the test are represented by the branches. A very intensive and comprehensive analysis is possible by making use of decision trees. The end product of the decisions taken

from the root to the leaf node give us the classification. There are 3 types of nodes in a decision tree, namely: 1. Decision nodes; 2. Chance nodes; 3. End nodes

Decision nodes are represented by squares, Chance nodes by a circle, and end nodes by triangles. Decision is advantageous as it does not require normalization or scaling of data. Missing values have no certain effects on the final outcome. The efforts of implementation of a decision tree as compared to other techniques in machine learning are far less. We use the package of sklearn for building our model, the Decision Tree Classifier from sklearn. tree is used for the purpose. We use the GINI for evaluating split between further nodes, and we chose the best split. We found that using bigrams along with unigrams didn't made any significant increase to the accuracy. The best validation accuracy achieved using decision trees is 70.2%. Table 4 shows the precision, recall and F1 score for corresponding accuracy.

Table 4. Evaluation of Decision Tree Classifier using different scores

	Precision	Recall	F1-score	Support
Negative Sentiment	0.68	0.51	0.58	8158
Positive Sentiment	0.71	0.84	0.77	1182
Overall Accuracy			14040/20000	70.2%

Random Forest

Random Forest is one of the best classification algorithms out there. Random Forest uses decision trees as the basic building block and it is a combination of various individual decision trees. So, the logic of the working of a random forest is a simple, but powerful one. In random forest, all the individual trees give out predictions for the given data and the final answer of random forest comes out to be the majority of the answers which we get from the individual trees. Suppose if there are 400 trees in a random forest and 340 trees predict output 1 and 60 trees predict output 0, then the final answer would be 0 from random forest. The reason why this model works so well is that there are a very large number of unrelated trees working together to find a common outcome and the trees which are a part of random forest helps to minimize each other's errors.

The possibility of predicting the correct outcome increases with the number of unrelated trees in random forest. And the way the trees are able to maintain the low correlation is because of bagging and feature randomness. A very small change can give us very different decision trees to what before they were. This process is called bagging. We have one important to remember here that we are not dividing the training data sample into small pieces but we feed the whole training data to each tree in random forest (Ren, Cheng & Han, 2017). Feature randomness is the process of choosing the feature by which the entities in training

data are most separable. Each tree gets to pick features only from a limited subset of features. This is how the trees in random forest are having such low correlation and high diversification and that's how random forest is able to achieve such accurate results. The package sklearn is used for implementing random forest algorithm, the Random Forest Classifier from sklearn. ensemble is used. For experimenting 10 estimators (trees) were used. We found that using bigrams along with unigrams didn't make any significant increase to the accuracy. The best validation accuracy achieved using random forest is 76.35%. Table 5 shows the precision, recall and F1 score for corresponding accuracy.

Table 5. Evaluation of Random Forest Classifier using different scores

	Precision	Recall	F1-score	Support
Negative Sentiment	0.72	0.70	0.71	8158
Positive Sentiment	0.79	0.81	0.80	1182
Overall Accuracy			15267/20000	76.355%

Logistic Regression

Logistic regression is one of the classification algorithms used in the classification of data. Logistic Regression is named after a function called as logistic function which can be understood as the basic function required for the efficient working of logistic regression (Nagar *et al.,* 2020). The logistic function is a sigmoidal function which is developed by the statisticians to fit the data points to its regression line. The value of logistic function always comes between 0 and 1 but it never becomes equal to 0 or 1 as shown in Figure 4. If we go on to plot the curve of logistic function, it would turn out to be an S shaped curve which can input any real valued number as a parameter. The formula for logistic function is 1/(1+e^-value) where 'e' is Euler's number and value are the real number which you want to put as parameter and get transformed.

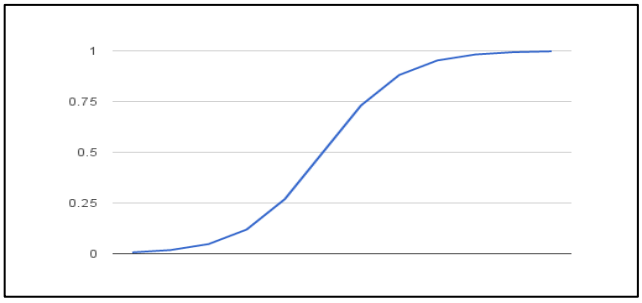


Figure 4. Logistic Function

The output of the logistic regression is in the form of log Odds. If we talk odds a little bit more, we can say that odds are just another means of defining the probability of an event.

$$\text{Odds} = P(\text{Event})/[1-P(\text{Event})] \quad (4)$$

Using this probability value, we can predict growth of an organization. Let's understand this more clearly with an example, suppose I try to roll a dice 1000 times and get six 300 times. Based on the given example, the probability of me getting a six is 30% i.e., 0.3. So, the odds of me getting 6 would be

$$\text{Odds} = 0.3/1\text{-}0.3 = 0.3/0.7 = 0.42857 \quad (5)$$

We use odds over probability because unlike probability they are not bound between 0 and 1 which really helps us in regression analysis. If we talk about the representation used in logistic regression, an equation is used as the representation similar to which is used in the linear regression. In logistic regression, input values or input parameters are used linearly using some coefficient values and an output value is predicted. For example, here is a sample logistic regression equation

$$Y = e^{(b0+b1*x)}/[1+e^{(b0+b1*x)}] \quad (6)$$

Here b0 is the intercept term and b1 is the coefficient term for the input value x. The coefficient values in the logistic regression equation must be valued from the training data. The value of these coefficients can be known with the help of maximum-likelihood estimation. Maximum likelihood gives us the best value of coefficients for our training data. Output values in logistic regression can be predicted by just putting values in a given regression equation and solving it to get a final numerical value. Since we now need a binary answer for example 0 or 1, we can have a threshold value and if output value from logistic regression equation < threshold value, then the final answer is 0. If output value from logistic regression equation >= threshold value, then final answer is 1. This is how we are able to do classification using logistics regression. We implemented a logistic regression model using sklearn package. The performance was improved from when we use only unigrams to when we use both unigrams and bigrams. The best performance accuracy we achieved was 78.925%. Table 6 shows the precision, recall and f1 score for corresponding accuracy.

Table 6. Evaluation of Logistic Regression Model using different scores

	Precision	Recall	F1-score	Support
Negative Sentiment	0.78	0.79	0.78	8158
Positive Sentiment	0.82	0.87	0.84	1182
Overall Accuracy			15785/20000	78.925%

Support Vector Machine (SVM)

Support vector machine is highly used by many as it produces remarkable accuracy and can be implemented in a system with less computation power as compared to other algorithms. Support Vector Machine, can be used for both regression and classification tasks. But it is mostly used in classification problems. Forecasting wind speed, has also been a good application of SVM (Bali *et al.*, 2019). The main task of the SVM algorithm is to develop a plane in an N dimensional space that classifies the data points distinctively (Gopi *et al.*, 2020). The task is to develop a plane that can produce very high margin, i.e., very high distance between points of both classes. Increasing the margin distance gives some changes so that upcoming points can be classified with greater accuracy. Hyperplanes are decision boundaries that do the work of classification of the points. Points on both side of the hyperplane cannot be considered to be of same classes and the dimension of the hyperplane relies upon the number of parameters. If the number of parameters is two, then the plane is only a line. If the number of parameters is three, then the plane will be a 2D plane. It becomes very cumbersome to find dimensions when the number of parameters is greater than three.

For logistic regression, we consider the result of the linear function and reduce the value between 0 and 1 by applying the sigmoid function. If the reduced output is greater than a limiting value of 0.5, we devote it a label one, or we devote it a label zero. In Support Vector Machine we take the result of the linear function and if that result is more than one, we use it with one class and if the output is negative one, we use it with another class. Because the limiting values are changed to one and negative one in Support vector machine, we get this range of values between negative one to positive one which used as margin.

In the Support Vector Machine, we are trying to increase the difference between the points and the plane. The loss calculating function that try to increase the difference is hinge loss. The cost is zero if the calculated output value and the original value are of the similar sign that is either both are negative or both are positive. If both values are not of same sign then we find the value of loss. We also add an optimization parameter the cost function. The objective of the optimization parameter is to make an equilibrium between the maximization of margin and loss which we can make (Nokel & Loukachevitch, 2015).

The SVM classifier is used from the sklearn package in this work. The C term which is the parameter of penalty in error term is set to 0.1. Both the configurations are used to test and best results are obtained with both unigrams and bigrams of 78.31% accuracy. Table 7 shows the precision, recall and f1 score for corresponding accuracy.

Table 7. Evaluation of Support Vector Machine Classifier using different parameters

	Precision	Recall	F1-score	Support
Negative Sentiment	0.76	0.69	0.72	8158
Positive Sentiment	0.80	0.85	0.82	1182
Overall Accuracy			15662/20000	78.31%

XGBoost

It is a decision-tree-based library developed to be very adapTable and efficient. XGBoost offers a simultaneous tree optimization (also defined as GBM, GBDT) that solve several data analytics problems in a quick and efficient way. The similar code runs on almost every distributed environment like SGE, Hadoop, and MPI and can resolve problems beyond thousands of examples. The XGBoost performs so well than Gradient Boosting Machines (GBMs), while both are tree methods which boost the weak learners by implementing the gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems enhancement and algorithmic optimization (Gaye & Wulamu, 2019). Various methods used are

- Parallelization: XGBoost uses the process of sequential tree making, by applying method of simultaneous implementation. This can be achievable because of the interdependent characteristic of loops which is used for making base learners; the last loop that enumerates the last (leaf) nodes of a tree, and the second last loop that find the features.

- Tree Pruning: The eliminating condition for tree splitting within GBM framework is greedy in characteristic and rely on the negative loss situation at the junction of split. XGBoost uses ‘max_depth’ condition as specified in place of condition first, and start pruning tree backward.

- Hardware Enhancement: Hardware Enhancement in the algorithm has been developed to make efficient use of hardware capabilities. This can be achieved by efficient use of cache and by providing internal buffers to every thread to store gradient statistics.

For controlling overfitting, we set the max depth of tree to 25. The higher value of depth might result in learning relations of model tied to train data. We need to tune the number of estimators used for getting best results. At the value of 400 we observed that we get the best results. The best result we got is 76.655% while using both unigrams and bigrams together. Table 8 shows the precision, recall and f1 score for corresponding accuracy.

Table 8. Evaluation of XGBoost Classifier using different parameters

	Precision	Recall	F1-score	Support
Negative Sentiment	0.72	0.69	0.71	8158
Positive Sentiment	0.79	0.82	0.81	1182
Overall Accuracy			15331/20000	76.655%

Results and Discussion

In this work, a dataset of tweets which was pre labelled with corresponding sentiment, as positive or negative is used. The tweets are hand labelled and not using some algorithm, which makes it an authentic and accurate dataset. Analysis was done on the dataset by various techniques of feature extraction, our methodology was to pre-process the raw data and then extract features. Further the machine learning models were trained using sparse vectors of tweets created after applying tf idf on the features.

Two types of features for experimentation which are unigrams and bigrams are used. From Table 9 we can report that use of both the features together show an improvement to the classification problem of assigning sentiment (positive or negative or neutral). Comparison of various machine learning models is shown in Figure 5. For obtaining the accuracy we have taken both the features unigrams and bigrams to train the model using sparse vector representation which apply tf idf on these vectors. From this it is evident that Support vector machine and logistic regression perform best. Logistic regression model has the best accuracy percentage as visible in Table 9.

Table 9. Comparison of various classifiers based on their accuracy depending upon type of feature

Algorithms	Accuracy	
	Unigram	Unigram + Bigram
Naïve Bayes	75.2450%	77.560%
Decision Tree	70.1650%	70.20%
Random Forest	76.2950%	76.3350%
Logistic Regression	77.050%	78.9250%
Support Vector Machine (SVM)	77.0950%	78.310%
XGBoost	75.950%	76.6550%

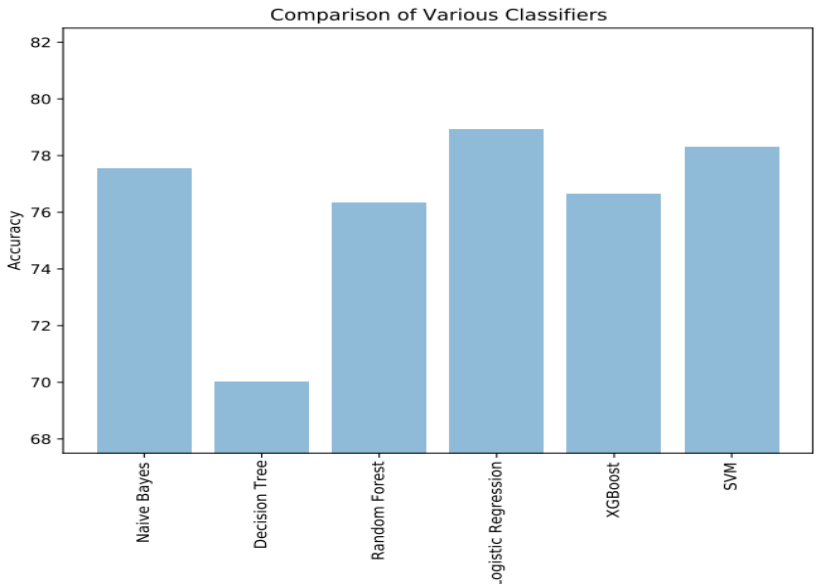


Figure 5. Comparison of accuracies of various models

Case Study: twitter hashtag lockdown

We further used the logistic regression model to apply it on a real time data from twitter. Tweets on the topic 'Lockdown' were obtained by scraping data from twitter with the geo location set to India. The process of scrapping data was conducted for a day to obtain unique tweets, a set of 1900 tweets on lockdown was obtained. The tweets were pre-processed and their sentiment were predicted using the model generated. The output from the model is categorical and is represented as types of sentiments and they are positive and negative. The analysis bore the results that show that there were at the time of analysis maximum number of positive tweets. The Table 10 shows the result of analysis with Figure 6 showing the pie chart representing the result of analysis. A word cloud of terms related to negative sentiment tweets, classified using the model, was created and is shown in Figure 7. It shows the terms most used in tweets on topic Lockdown, which were classified as negative sentiment tweets.

Table 10. Results of analysis of tweets obtained with keyword lockdown

	Positive	Negative	Neutral	Total
Percentage	48%	20%	32%	100%
Number of tweets	912	380	608	1900

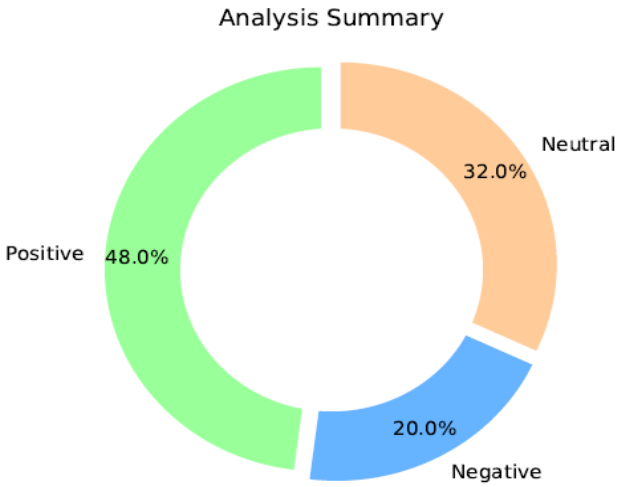


Figure 6. Result summary of analysis of tweets on case study

Figure 7. Word Cloud of terms used in tweets, classified as negative, on topic 'Lockdown'

The word cloud assigns different font size to each term according to its rank obtained. A max font size is set and assigned to word ranked as 1, the most frequent term, after the assigning max font size to rank 1 term subsequent terms are assigned a font size by multiplying the max font size with its frequency, which lies between 1 and 0 and is obtained by dividing the frequency of term by frequency of most frequent term.

This case study could really help the government to take constructive feedback on its applied policies as we are having a word cloud of negative tweets and then they can work on it and make a better decision next time. This method of getting public feedback is a much better method than the one being used currently because it requires less manpower and less resources.

Conclusion

We presented the results of various models for performing sentiment analysis of twitter data. We provide a survey of existing machine learning models using evaluation metrics, and by experimenting with different feature types in train data. We used two types of features for experimentation which are unigrams and bigrams and report that the use of both the features together shows an improvement to the classification problem of assigning sentiment (positive or negative or neutral). We investigated various statistical models for classification available and the results show that the logistic regression classification model performs best in the case of text classification. We also concluded through the results that analysing twitter data is no different than any other text analysis considering the informal language used to create tweets.

Considering the current approach of government for analysis of public opinion towards the laws and decisions implemented we have provided an efficient way to conduct such surveys using social media. Taking the example of India, NSSO conducts various surveys amongst which comes the survey of government policies and actions. These surveys are conducted physically requiring a lot of manpower and resources. The anonymity provided by the internet lets people be more honest and hence the surveys or analyses done on social media data are more efficient. Surveys conducted using twitter data are more efficient as Twitter contains the authenticated accounts of politicians, entrepreneurs, and Industry leaders which is not the case of other social media platforms.

Our case study on Tweets on the current topic of "Lockdown" showed results which can be utilised to make reforms in the decisions and can also help in future decision making. The user (government) can easily find out how they can help people that are dissatisfied at this time using the negative word cloud and hence, take necessary steps. The terms highlighted using word cloud can be used to improve the policies and cater the crowd who might be left out in the previous draft of policy. This way the application provides the government with a way to help people in need. In future work, we will explore semantics analysis and topic modelling. Also perform analysis using unsupervised way of machine learning to experiment and seek better accuracy.

References

Aggarwal, D. G. (2018). Sentiment Analysis: An insight into Techniques, Application and Challenges. *International Journal of Computer Sciences and Engineering*, 2018, 6(5), 697-703.

Aggarwal, D., Mittal, S., & Bali, V. (2019). An Insight into Machine Learning Techniques for Predictive Analysis and Feature Selection. *International Journal of Innovative Technology and Exploring Engineering*, 8(9S), 342-349.

Aggarwal, D., Mittal, S., & Bali, V. (2019). Prediction Model for Classifying Students Based on Performance using Machine Learning Techniques. *International Journal of Recent Technology and Engineering*, 8(2S7), 497-503.

Ahuja, R., Chug, A., Kohli, S., Gupta, S., & Ahuja, P. (2019). The Impact of Features Extraction on the Sentiment Analysis. *Procedia Computer Science*, 152, 341-348.

Alsaeedi, A. & Khan, Z. (2019). A Study on Sentiment Analysis Techniques of Twitter Data. *International Journal of Advanced Computer Science and Applications* (IJACSA), 10(2). 361-374.

Doaa, M., & Hussein, E. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, 30(4), 330-338.

Fan, X., Li, X., Du, F., Li, X. & Wei, M. (2016). Apply word vectors for sentiment analysis of APP reviews. *3rd International Conference on Systems and Informatics (ICSAI)*, Shanghai, 1062-1066.

Gangwar, S., Bali, V. & Kumar, A. (2019). Comparative Analysis of Wind Speed Forecasting Using LSTM and SVM. *EAI Endorsed Transactions on Scalable Information Systems*, 1-8.

Gaye, B., & Wulamu, A. (2019). Sentimental Analysis for Online Reviews using Machine learning Algorithms. *International Research Journal of Engineering and Technology*, 6(8), 1270-1275.

Giachanou, A. & Crestani, F. (2016). Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *ACM Computer Survey,* 49(2), 1-41.

Gopi, A.P., Jyothi, R.N.S., Narayana, V.L. (2020). Classification of tweets data based on polarity using improved RBF kernel of SVM. *International Journal of Information Technology*, https://doi.org/10.1007/s41870-019-00409-4

Hasan, A, Moin, S, Karim, A, & Shamshirband, S. (2018). Machine Learning-Based Sentiment Analysis for Twitter Accounts. *Mathematical and Computational Applications*, 23(1), 11. https://doi.org/10.3390/mca23010011

Ikoro, V., Sharmina, M., Malik, K. & Batista-Navarro, R. (2018). Analyzing Sentiments Expressed on Twitter by UK Energy Company Consumers. *Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 95-98.

Kumar, A., Jaiswal, A. (2020). Systematic literature review of sentiment analysis on Twitter using soft computing techniques. *Concurrency Computat Pract*, 32: e5107.

Mäntylä, M.V., Graziotin D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, 27, 16-32.

Mittal, A., & Patidar, S. (2019). Sentiment Analysis on Twitter Data: A Survey. *7th International Confrence on Computer and Communications Management, Association for Computing Machinery*, New York, 91-95.

Nagar, R., Aggarwal, D., Saxena, U. R. & Bali, V. (2020). Early Prediction and Diagnosis for Cancer Based on Clinical and Non-Clinical Parameters: A Review. *International Journal of Grid and Distributed Computing*, 13(1), 548-557.

Nokel, M.A., & Loukachevitch, N.V. (2015). Topic models: adding bigrams and taking account of the similarity between unigrams and bigrams. *Numerical methods and programming*, 16(2), 215-234.

Proisl, T. & Uhrig, P. (2016). SoMaJo: State-of-the-art tokenization for German web and social media texts. *Proceedings of the 10th Web as Corpus Workshop, Association for Computational Linguistics*, Berlin, 57-62.

Ramanathan, V. and Meyyappan, T. (2019). Twitter Text Mining for Sentiment Analysis on People's Feedback about Oman Tourism, *4th MEC International Conference on Big Data and Smart City (ICBDSC)*, Muscat, Oman, 1-5.

Reddy, A.,Vasundhara, D.N., & Subhash, P. (2019). Sentiment Research on Twitter Data. *International Journal of Recent Technology and Engineering*, 8(2S11), 1068-1070.

Ren, Q., Cheng, H., & Han, H. (2017). Research on machine learning framework based on random forest algorithm. *AIP Conference Proceedings*, 1-7.

Saif, H., He, Y., Fernandez, M., & Alani, H. (2016). Contextual semantics for sentiment analysis of Twitter. *Inf. Process. Manage*. 52(1), 5–19.

Salinca, A. (2015). Business Reviews Classification Using Sentiment Analysis. *17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, 247-250.

Uijlings, J.R.R., Smeulders, A. W. M., & Scha, R. J. H. (2009). Real-time bag of words, approximately. *ACM International Conference on Image and Video Retrieval (CIVR '09),* 6, 1–8.

Vijayarani, S. & Janani, R. (2016). Text Mining: open-Source Tokenization Tools – An Analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, 3(1), 37-47.

Zhang, Y., Liu, N., & Wang S. (2018). A differential privacy protecting K-means clustering algorithm based on contour coefficients. *PLoS One*. 13(11): e0206832

Bibliographic information of this paper for citing:

Aggarwal, Deepti; Bali, Vikram; Agarwal, Abhishek; Poswal, Kshitiz; Gupta, Madhav & Gupta, Abhishek (2021). Sentiment Analysis of Tweets Using Supervised Machine Learning Techniques Based on Term Frequency. *Journal of Information Technology Management*, 13(1), 119-141.