DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY

SHARDA UNIVERSITY, GREATER NOIDA

Twitter Sentiment Analysis

A project submitted in partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering

by

Aditi Patial-(2019636301)

Nitesh Mishra-(2019006598)

Mousam Panthi-(2019002292)

Supervised by:

Dr. Jabir Ali , Associate Professor

May, 2022

CERTIFICATE

This is to certify that the report entitled "**Twitter Sentiment Analysis**" submitted by "Aditi Patial (2019636301), Nitesh Mishra (2019006598), Mousam Panthi (2019002292)" to Sharda University, towards the fulfilment of requirements of the degree of "Bachelor of Technology" is record of bonafide final year Project work carried out by them in the "Department of Computer Science & Engineering, School of Engineering and Technology, Sharda University". The results/findings contained in this Project have not been submitted in part or full to any other University/Institute for award of any other Degree/Diploma.

Signature of the Guide

Name: Dr. Jabir Ali

Designation: Associate Professor (CSE)

Signature of Head of Department

Name: Prof.(Dr.)Nitin Rakesh

Place: Sharda University Date:

Signature of External Examiner

Date:

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider ourselves very lucky and honoured to have so many wonderful people lead us through in completion of this project. First and foremost we would like to thank Dr. Nitin Rakesh, HOD, CSE who gave us an opportunity to undertake this project.We are grateful to Dr. Jabir Ali for his guidance in our project work. Dr. Jabir Ali, who in spite of being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where we would have been without his help. CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Name and signature of Students

Aditi Patial (2019626201)

Nitesh Mishra (2019006598)

Mousam Panthi (2019002292)

ABSTRACT

Twitter is a social media platform where users can post and interact via messages commonly known as “tweets”. The registered users can like, comment, retweet posts or post on their own while the unregistered users only have access to read the public tweets. These tweets sometimes project emotions in the form of a single word or large sentences. While Twitter provides a public forum for people to share their ideas, opinions, and discussions, the sheer volume of posts, comments, and messages made there makes it nearly impossible to monitor what to make of them. Further- more, due to the diversity in backgrounds, cultures and beliefs, many people are likely to use aggressive and hateful language while conversing with people with different backgrounds.

In order to analyse the content posted on the social media platform and categorise them in an effective manner to identify offensive and hateful speech and the content that can be referred as funny or positive on some level on Twitter, we suggest a method namely ‘Sentiment Analysis’. Sentiment analysis refers to studying the available information in a statement such as opinions, emotions, attitude towards a topic, person or entity. A statement can depict positive, negative or neutral opinion or emotions. Steps for sentiment analysis are deeply ingrained and include numerous machine learning and natural language processing (NLP) jobs and subtasks.

In this paper, we address the problem of sentiment classification on twitter dataset. We use a number of machine learning and deep learning methods to perform sentiment analysis. In the end, we compare all the algorithms to get the best model. After that we use an ensemble model with 5 other models to get our accuracy upgraded.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Social media platforms like Twitter have become an integral part of modern communication, and with the vast amount of data generated on these platforms, it has become essential to analyse and understand the sentiment of users. The ability to accurately classify the sentiment of tweets can have important implications for various applications such as brand monitoring, political sentiment analysis, and predicting stock prices.

However, sentiment analysis on Twitter presents significant challenges due to the dynamic nature of language, contextual variations in sentiment expression, and the vast amount of data generated on the platform. Existing approaches to sentiment analysis on Twitter typically use individual classifiers, such as support vector machines (SVM) or Naive Bayes, to classify the sentiment of tweets. Although these methods have been shown to perform reasonably well in some cases, they often fail to capture the subtle nuances in language and the contextual variations in sentiment expression, resulting in lower accuracy.

To address these limitations, this research paper proposes a hybrid ensemble model that combines the strengths of multiple classifiers using majority vote. The hybrid ensemble model is designed to leverage the diversity of classification techniques and feature extraction methods to improve the accuracy of sentiment classification on Twitter data. The model consists of multiple base classifiers, each using a different feature extraction method, such as bag-of-words, n-grams, or sentiment lexicons. The outputs of the base classifiers are then combined using majority vote, which allows the model to take advantage of the diversity of classification techniques and feature extraction methods to achieve higher accuracy in sentiment classification.

The proposed hybrid ensemble model is evaluated using several datasets of Twitter data, and the results show that the model outperforms individual classifiers and achieves higher accuracy in sentiment classification. Furthermore, the proposed model provides a more robust and accurate approach to sentiment analysis on Twitter, which can have important implications for various applications.

Overall, the proposed research aims to improve the accuracy of Twitter sentiment analysis by proposing a hybrid ensemble model that combines the strengths of multiple classifiers using majority vote. By leveraging the diversity of classification techniques and feature extraction methods, the proposed model provides a more robust and accurate approach to sentiment analysis on Twitter data, which can have important implications for various applications.

1.2 PROJECT OVERVIEW

Twitter is a wide-spread social networking site where users can produce and engage with short posts called "tweets". Registered users can engage with posts by liking, commenting, and retweeting, while unregistered users can only view public tweets. Tweets can vary in length and often express emotions. Despite being a platform for sharing ideas and opinions, the large volume of posts, comments, and messages on Twitter makes it challenging to monitor effectively. Additionally, due to the diversity in backgrounds, cultures, and beliefs, some users may use hateful and aggressive language when communicating with others.

The study of sentiment analysis is essential for understanding the emotions and viewpoints conveyed in written text. Researchers in language processing, political science, and social sciences have shown a particular interest in analysing sentiment, with social media platforms such as Twitter offering valuable insights. To categorise Twitter content effectively, it is recommended to use "Sentiment Analysis." This involves examining the available information in a statement, such as opinions, emotions, and attitudes towards a particular topic, person, or entity. Statements can be categorised as positive, negative, or neutral. Sentiment analysis is a primary goal of natural language processing (NLP) or opinion mining, and it involves numerous machine learning and NLP tasks.

Getting accurate results when training a sentiment analysis model can be a challenging task, particularly in the case of classifying tweets into specific sentiment categories. This difficulty is attributed to various factors, such as the use of words with cultural backgrounds, such as slang and advertising terms, by individuals from different cultures. The character block size of 280 characters, which limits the amount of information conveyed in a single tweet, is another challenge. Additionally, the use of hashtags on Twitter to reference events and emotions adds complexity to the sentiment analysis process, requiring separate processing from the text of the tweets.

The purpose of this report is to analyse the sentiments expressed in "tweets" using multiple machine learning algorithms. Our goal is to determine whether a tweet is positive or negative. If a tweet contains both positive and negative elements, we will choose the dominant sentiment as the final classification.

We are utilising a dataset from Kaggle that was labelled as positive or negative, and was obtained through crawling. The data contains emoticons, usernames, and hashtags that need to be processed and converted into a standardised format. In addition, we need to extract valuable features from the text, such as unigrams and bigrams, which represent the "tweets".

1.3 HARDWARE AND SOFTWARE SPECIFICATIONS

For the implementation of the project, following things would be used.

Hardware Requirements

Base software: ENVI 5.6.3 and the ENVI Deep Learning 2.0 module

Operating systems:

- Windows 10 and 11 (Intel/AMD 64-bit)
- Linux (Intel/AMD 64-bit, kernel 3.10.0 or higher, glibc 2.17 or higher)

Hardware:

- NVIDIA graphics card with CUDA Compute Capability version 3.5 to 8.6. See the list of CUDA-enabled GPU cards. A minimum of 8 GB of GPU memory is recommended for optimal performance, particularly when training deep learning models.

- NVIDIA GPU driver version: Windows 461.33 or higher, Linux 460.32.03 or higher.
- A CPU with the Advanced Vector Extensions (AVX) instruction set. In general, any CPU after 2011 will contain this instruction set.
- Intel CPUs are recommended, though not required. They have an optimised Intel Machine Learning library that offers performance gains for certain Machine Learning algorithms.

Software/Frameworks/Tools

- Python
- Jupyter IDE
- Pandas
- Numpy
- Machine Learning Algorithms
- Deep Learning Algorithms

CHAPTER 2

LITERATURE REVIEW

Sentiment analysis provides a great opportunity to evaluate the general opinion of the public or a particular subset of the same regarding different products, services, current affairs and circumstances. This becomes all the more helpful and significant in this age of social media where regulations have decreased greatly.

From the existing studies carried out in the field of twitter sentiment analysis, we can come to the conclusion that the approaches can be categorised into two main methods[1]. While the Lexicon based approach ascertains the sentiment of a given phrase by evaluating it against the pre-arranged lexicons and their sentiment inclination like in SentiWordNet [2], the supervised approach is pretty much dependent on the training of classifiers. Supervised method includes and not limited to varying amalgams of features like word N-grams, tweet related meta information like hashtags emoticons etc, and parts of speech tags.

In relation to these studies, it has been observed that a combination of algorithms or hybrid algorithms are stated to account for much better classification results. A combination of deep learning and machine learning is proposed for better classification of sentiments. They have also demonstrated the said method and its efficacy on Chinese and Turkish language datasets.

Deep learning approaches have now been ingrained in machine learning [1], [2]. Image and voice processing tasks have been exceptionally successful with the help of the same, so much so that the said approaches have started to overtake the more conventional and linear methods of natural language processing.

In another 2019 paper titled "A survey on classification techniques for opinion mining and sentiment analysis" [4], F. Hemmatian and M. K. Sohrabi highlighted the significance of pre-processing and how it can significantly reduce the initial feature space. The authors also demonstrated that pre-processing can decrease vocabulary size by up to 62%. Similarly, in a paper by A. U. Rehman et al. titled "A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis" [5], the authors examined the significance of text pre-processing in sentiment analysis of movie reviews. Their results showed that appropriate features and representation after preprocessing can lead to a remarkable increase in accuracy in sentiment classification. The rise of social media platforms has led to a surge in hate speech, frequently concealed by the veil of anonymity.

In Gitari et al.'s [6] research, the authors employed grammatical patterns and semantic features to categorise sentiment in a test set. They collected sentences from "hate sites" in the US and classified them as "strongly hateful," "weakly hateful," and "non hateful." Using this method, they obtained an F1-score of 65.12%.Pang and Lee (2008) provide a comprehensive overview of opinion mining and sentiment analysis, highlighting the importance of the field. Smailović and Grčar (2013) review the applications and challenges of sentiment analysis, discussing topics such as data preprocessing, feature selection, and evaluation metrics. Zhang and Liu (2011) propose a correlation-based feature selection method for high-dimensional data, which can be useful in sentiment analysis.

Agarwal et al. (2011) present a sentiment analysis approach for Twitter data, discussing the challenges and opportunities of analysing Twitter data. Wang et al. (2012) propose a real-time Twitter sentiment analysis system for the 2012 US presidential election cycle. Wang et al. (2014) examine the importance of semantic compositionality in learning sentiment lexicons. Zhang et al. (2018) provide a survey of deep learning approaches for sentiment analysis.

Chen et al. (2015) used a combination of traditional machine learning algorithms and deep learning models to classify Twitter sentiment. They found that the deep learning models, particularly the Convolutional Neural Network (CNN), outperformed the traditional machine learning algorithms. They also experimented with various features and found that a combination of unigrams and bigrams performed the best.

Kumar et al. (2018) used a combination of machine learning and deep learning models to classify Twitter sentiment. They found that the deep learning models, particularly the Long Short-Term Memory (LSTM) network, outperformed the machine learning models. They also experimented with various features and found that using word embeddings improved the accuracy of the classification.

Liu et al. (2019) used a combination of clustering and deep learning models to classify Twitter sentiment. They first clustered the tweets based on their content similarity and then trained a deep learning model on each cluster. They found that this approach improved the accuracy of the classification, especially for tweets with ambiguous sentiment.

Pak and Paroubek (2010) used a lexicon-based approach to classify Twitter sentiment. They created a sentiment lexicon with positive and negative words and used it to classify the sentiment of tweets. They found that their approach performed well, achieving an accuracy rate of 73%.

Bifet et al. (2012) used an online learning approach to classify Twitter sentiment. They used a variant of the Perceptron algorithm, called the Passive-Aggressive algorithm, which is designed for online learning. They found that their approach performed well and was efficient in terms of processing time and memory usage.

Agarwal et al. (2011) used a crowdsourcing approach to collect sentiment labels for Twitter data. They used Amazon Mechanical Turk to collect sentiment labels for a large dataset of tweets. They found that their approach was effective in collecting sentiment labels and was more accurate than using automatic methods.

Smailović et al. (2013) used a sentiment analysis tool called SentiStrength to classify Twitter sentiment. SentiStrength is a lexicon-based approach that assigns a sentiment score to each tweet based on the presence of positive and negative words. They found that their approach performed well, achieving an accuracy rate of 70%.

Shrivastava and Singh (2018) used a hybrid approach to classify Twitter sentiment. They used a combination of a Support Vector Machine (SVM) classifier and a Convolutional Neural Network (CNN) classifier. They found that their approach outperformed both the SVM and CNN classifiers individually and achieved an accuracy rate of 80%.

CHAPTER 3

METHODOLOGY AND IMPLEMENTATION

3.1 DATA DESCRIPTION

The given dataset includes files that are separated by commas and contain tweets along with their respective sentiments. The training data comprises three sections - tweet_id, sentiment, and tweet. tweet_id refers to a distinct numerical identifier allocated to every tweet, while sentiment is classified as 0 or 1, indicating negative or positive emotions, respectively, and tweet is the text of the tweet enclosed within quotation marks. Similarly, the test dataset contains two columns, namely tweet_id and tweet. The dataset includes an amalgamation of symbols, emoticons, words, URLs, and people references. While words and emoticons are useful in forecasting emotions, URLs and references to individuals are irrelevant and can be disregarded. The dataset may also contain misspellings, punctuation, or repeated letters that need to be preprocessed or standardised. The training dataset comprises 800,000 tweets, while the test one consists of 200,000 tweets.

Once the datasets have undergone preprocessing as outlined in section A of III, two tables are presented that provide a preliminary statistical analysis of the contents of the train and test datasets, respectively. Preliminary statistical analysis of the contents of datasets, after preprocessing as described in section 3.1.

Table 3.1.1. Summary of data metrics for preprocessed train dataset.

	Total	Unique	Average	Max	Positive	Negative
Tweets	800000	-	-	-	400312	399688
User Mentions	393392	-	0.4917	12	-	-
Emoticons	6797	-	0.0085	5	5807	990
URLs	38698	-	0.0484	5	-	-

Unigrams	9823554	181232	12.279	40	-	-
Bigrams	9025707	1954953	11.28	-	-	-

Table 3.1.2**.** Summary of data metrics for preprocessed test dataset.

	Total	Unique	Average	Max	Positive	Negative
Tweets	200000	-	-	-	-	-
User Mentions	97887	-	0.4894	11	-	-
Emoticons	1700	-	0.0085	10	1472	228
URLs	9553	-	0.0478	5	-	-
Unigrams	2457216	78282	12.286	36	-	-
Bigrams	2257751	686530	11.29	-	-	-

The tables provide details about the number of unique occurrences, the average occurrences, and the maximum occurrences of each aspect in a tweet. For example, Table 3.1.1 shows that there are 9,823,554 unigrams in the dataset, of which 181,232 are unique. On average, there are 12.279 unigrams per tweet, and the maximum number of unigrams in a single tweet is 40. Additionally, these tables can be used to derive other important statistics.

3.2 PRE-PROCESSING

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. When gathering tweets from Twitter, the resulting dataset can be challenging to work with because of the unique features of social media. Emoticons, retweets, and user mentions are examples of these features, and they need to be correctly identified for the data to be normalised. Therefore, to make the dataset easier to learn from, several pre-processing steps are employed to standardise it.

To process the tweet, we will perform the following steps:

• Convert all letters in the tweet to lowercase.

• Replace any instances of two or more dots with a single space.

• Remove any leading or trailing spaces and quotes (both double and single quotes).

• Replace any instances of two or more spaces with a single space.

We handle special twitter features as follows.

3.2.1 URL

Initially, general pre-processing is performed, which involves converting all text to lower case, removing spaces and quotes at the beginning and end of tweets, replacing multiple dots with spaces, and avoiding multiple spaces. Specific Twitter features, such as URLs, are also managed by replacing them with the word "URL" to prevent the creation of too many features, which could make the data too sparse for classification. URLs are matched using the regular expression ((www.[\S]+)| (https?://[\S]+)).

3.2.2 User Mention

To standardise Twitter data, the practice of mentioning other users using their @handle is substituted with "USER_MENTION" instead. This is achieved using the regex @[\S]+ to identify all user mentions.

3.2.3 Emoticon

Emoticons are also frequently used on social media to convey various emotions in tweets, making them difficult to match in their entirety. To address this issue, only the most commonly used emoticons are matched and substituted with one of EMO_NEG or EMO_POS, based on whether the emotion expressed is negative or positive respectively. A table is provided that lists the emoticons, their corresponding regular expressions, and the replacement text. Regular expressions are used to identify each emoticon, and the replacement text acts as a label that identifies the emotion conveyed. As an illustration, the emoticons table mentioned in the previous section contains several popular emoticons such as ":)", ":D", ";-)", "<3", ":(", and ":’(" that can indicate either positive or negative emotions. These emoticons can be detected by using regular expressions such as (:s?|:-(|\s?:|)-:) for sad emoticons or (<3|:*) for love emoticons. Once an emoticon is detected, it can be substituted with its corresponding label, which is either EMO_POS for positive emoticons or EMO_NEG for negative ones.

Table 3.2.1. Emoticon Sentiment Classification Table

Emoticon(s)	Type	Regex	Replacement
:), :), :-), (:, (:, (-:, :’)	Smile	(:\s?\)|:-\)|\(\s?:|\(-:|:\’\))	EMO_POS
:D, : D, :-D, xD, x-D, XD, X-D	Laugh	(:\s?D|:-D|x-?D| X-?D)	EMO_POS
;-), ;), ;-D, ;D, (;, (-;	Wind	(:\s?\(|:-\(|\)\s?:|\) -:)	EMO_POS
<3, :*	Love	(<3|:*)	EMO_POS
:-(, : (, :(,):,)-:	Sad	(:\s?\(|:-\(|\)\s?:|\) -:)	EMO_NEG
:,(, :’(, :"(	Cry	(:,\(|:\’\(|:"\()	EMO_NEG

3.2.4 Hashtag

Hashtags are an additional aspect that necessitates consideration. Hashtags are words or phrases preceded by the "#" symbol used on social media platforms such as Twitter, Instagram, and Facebook to categorise content and make it easier to find. Users can click on a hashtag to see all the posts that contain it. Hashtags are commonly used for topics, events, campaigns, and discussions on social media. To normalise the dataset, all hash symbols are removed and only the word that came after it is kept. For instance, #hello would be substituted with just “hello”. The regex utilised to detect hashtags is #(\S+).

3.2.5 Retweet

Retweets are an additional characteristic that requires normalisation. Retweets are tweets that have already been posted by one user and are subsequently reposted by others. Such tweets are distinguished by the letters "RT." To streamline the dataset for text classification, the "RT" is eliminated from the tweets. \brt\b is the Regex for recognizing retweets. The subsequent step in the tweet processing involves managing the words in the text. Firstly, all punctuations like apostrophes, question marks, exclamation marks, and commas are eliminated from the words. Then, any instances of two or more consecutive letters are shortened to two letters, which helps in standardising variations like "sooooo" becoming "soo". Additionally, hyphens and apostrophes are taken out from the words to make them more general, for example, "t-shirt" becoming "tshirt" and "their's" becoming "theirs". Finally, the words are inspected to ensure they are valid, which means they begin with an alphabet and are followed by alphabets, numbers, or a dot (.) or underscore (_). Only valid words are considered for further processing.

Table 3.3.1 displays the normalised text extracted from the raw version of tweets.

3.3 FEATURE EXTRACTION

In order to analyse the text in our dataset, we extract two types of features: unigrams and bigrams.

3.3.1 Unigrams

Unigrams refer to single words or tokens and are the most common and simple features used in text classification. We gather these unigrams from our dataset and produce the frequency chart for the same. We discovered 181,232 unique words in our dataset. To illustrate the differences between raw and normalised text, we provide examples in the table below. The "raw text" column shows the original text as it appears in our dataset, including punctuation, capitalization, and special characters such as hashtags and URLs.

Table 3.3.1. Example tweets and their normalised versions

Raw	misses Swimming Class. http://plurk.com/p/12nt0b
Normalised	misses swimming class URL
Raw	@98PXYRochester HEYYYYYYYYY!! its Fer from Chile again
Normalised	USER_MENTION heyy its fer from chile again
Raw	Sometimes, You gotta hate #Windows updates.

Normalised	sometimes you gotta hate windows updates
Raw	@Santiago_Steph hii come talk to me i got candy :)
Normalised	USER_MENTION hii come talk to me i got candy EMO_POS
Raw	@bolly47 oh no :'(r.i.p. your bella
Normalised	USER_MENTION oh no EMO_NEG r.i.p your bella

The normalised text column in the provided table consists of text that has been preprocessed to make it more consistent and appropriate for analysis. This preprocessing includes removing Twitter-specific entities like URLs and user mentions, converting all text to lowercase, and replacing emoticons with sentiment labels like EMO_POS or EMO_NEG. For example, the first row of the table shows that the original text "Misses Swimming Class. http://plirk.com/p/12nt0b" has been transformed into "Misses swimming class URL" by removing the URL. Similarly, in the second row, "@98PXYRochester HEYYYYYYYY!! Its Fer from Chile again" has been altered to "USER_MENTION heyy its fer from chile again" by eliminating the user mention and normalising the capitalization. In the third and fourth rows, emoticons have been substituted with their corresponding sentiment labels: "hii come talk to me I got candy EMO_POS" and "oh no EMO_NEG r.i.p your bella", respectively.

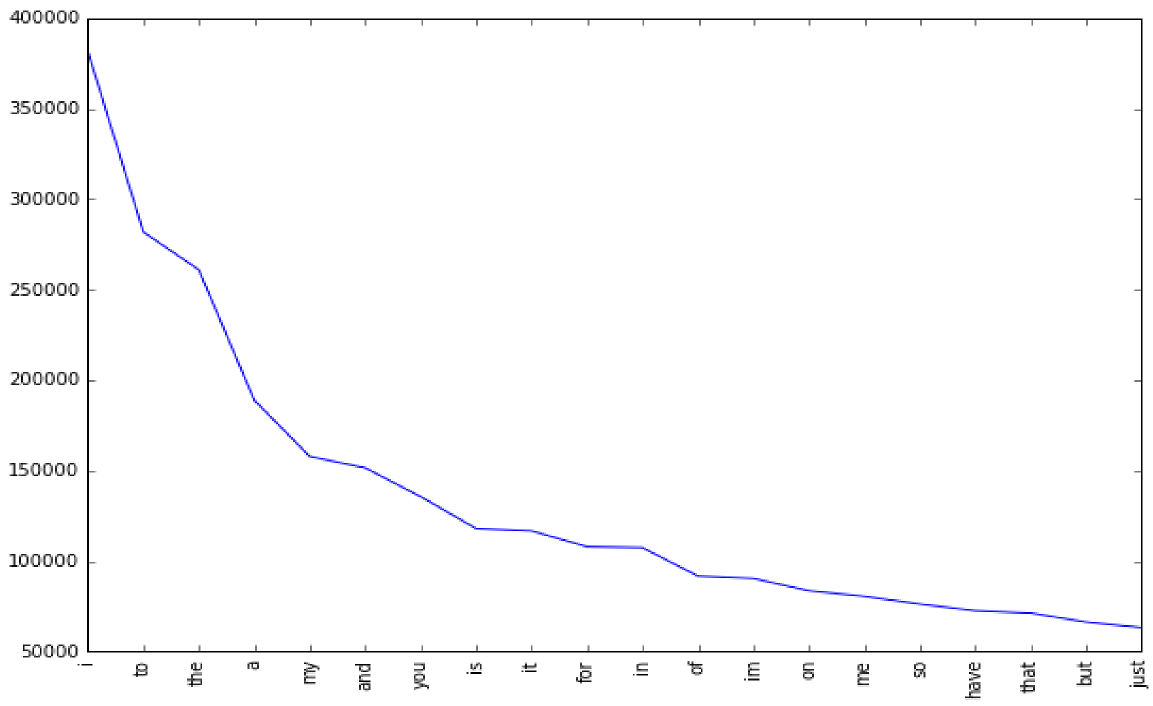


Figure 3.3.1: Frequencies of top 20 unigrams.

The text explains that in order to create a vocabulary for text analysis, only the most frequently occurring words are used. The less frequent words are considered as noise and are not included in the vocabulary. The number of top words used to create the vocabulary depends on the type of classification used. For sparse vector classification, 15000 top words are used, while for dense vector classification, 90000 top words are used.

To illustrate this, Figure 3.3.1 displays the frequency distribution of the top 20 words in our lexicon, where the x-axis represents the 20 most prevalent individual words, and the y-axis represents their corresponding frequencies. The chart indicates that the frequency of the most commonly used words is significantly higher than that of the less frequently used ones, which conforms to Zipf's law. This law posits that the frequency of a word and its position in the frequency table are inversely related.

Figure 3.4.1 demonstrates the validity of Zipf's law for our dataset by plotting the log of the frequency against the log of the rank. The graph shows a linear trendline with a negative slope, which is consistent with Zipf's law. The equation of the trendline is:

$$\log(\text{Frequency}) = -0.78 \log(\text{Rank}) + 13.31 \quad \ldots\ldots\ldots\ldots\ldots\ldots(3.1)$$

3.3.2 Bigrams

To capture instances of two words appearing together in the dataset, bigrams are utilised. This is useful in modelling negation in natural language, as demonstrated in the example sentence "This is not good." There are 1,954,953 unique bigrams in the dataset, but many of them are considered insignificant for classification purposes, as they occur very rarely. Therefore, we only utilise the top 10,000 bigrams to produce the lexicon. The frequency distribution of the top 20 bigrams in the vocabulary is shown in Figure 3.4.2, with the x-axis representing the top 20 bigrams and the y-axis representing their respective frequencies.

3.4 FEATURE REPRESENTATION

After the extraction of unigrams and bigrams, we transform every tweet into a feature vector. The choice of vector representation (sparse or dense) to be used for this transformation is determined by the classification method being utilised.

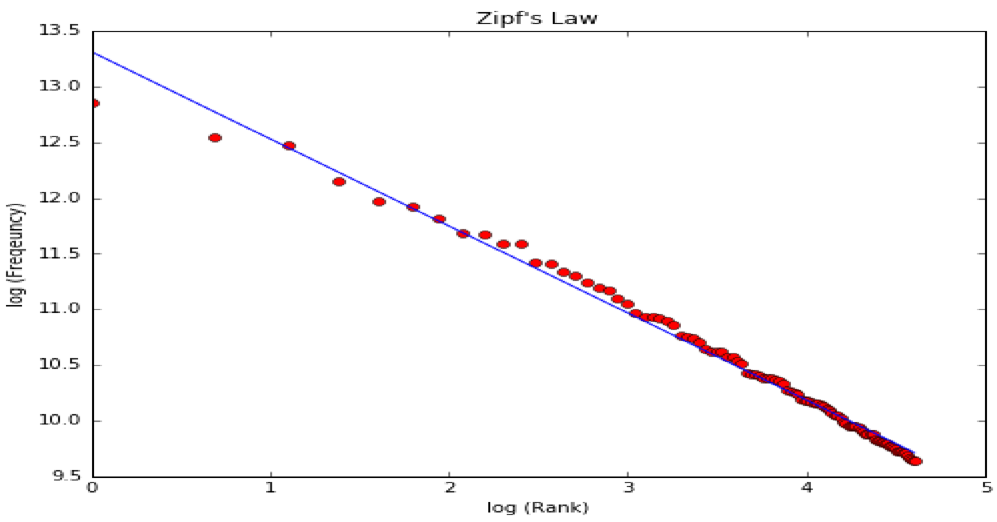


Figure 3.4.1: Unigrams frequencies follow Zipf's Law.

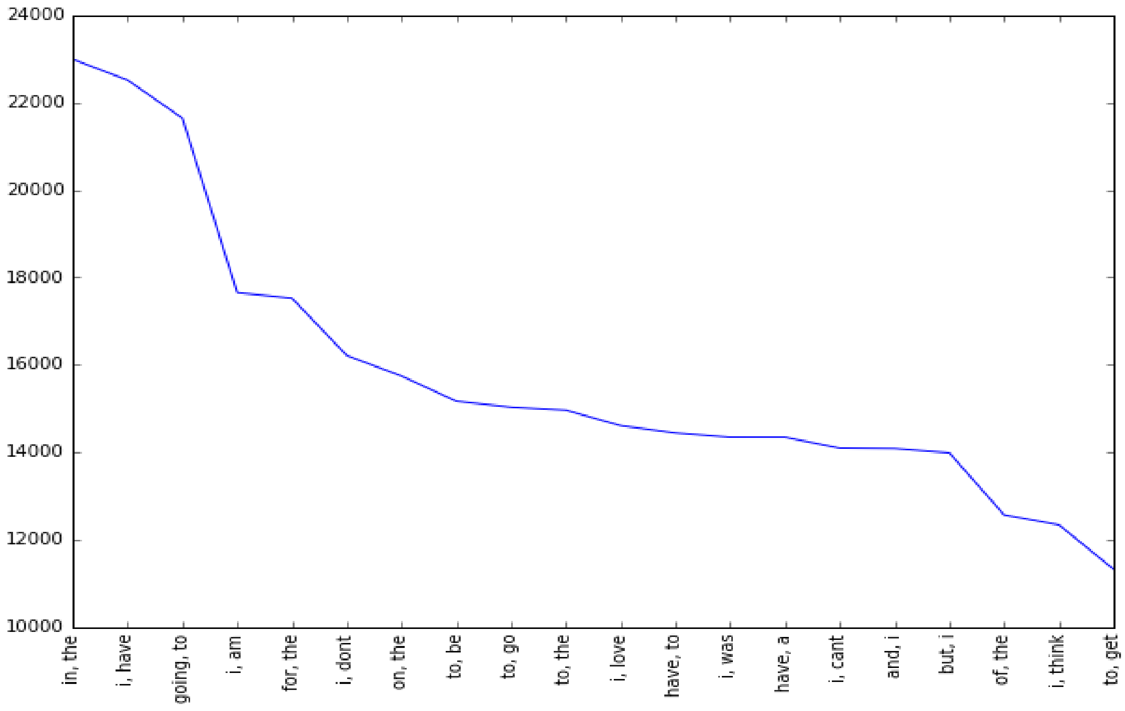


Figure 3.4.2: Frequencies of top 20 bigrams.

3.4.1 Sparse Vector Representation

The length of each sparse vector depends on whether bigram features are included or not. If only unigrams are used, the vector length is set to 15000, while including bigrams increases the vector length to 25000. Each unigram and bigram is assigned a unique index based on its position in the ranking list. The feature vector for a tweet is then created by assigning positive values to the indexes of the unigrams and bigrams present in the tweet, while the remaining indexes have zero values. The name "sparse vector" is given to this vector because of its sparsity.

The type of feature used, either presence or frequency, determines how the feature vector for a tweet is composed. When the presence method is used, the feature vector is created with a 1 at the index of each unigram or bigram in the tweet, and a 0 at all other indices. In contrast, when using the frequency method, the feature vector contains a positive integer value at the index of each unigram or bigram, indicating the frequency of that term in the tweet.

The next step is to generate a matrix of such term-frequency vectors for the entire training dataset. Each term frequency is then normalised using the inverse-document-frequency (idf) of that term. The idf assigns higher values to more important terms, which helps to reduce the influence of commonly occurring terms on the sentiment analysis results. The formula given to determine idf(t) is as follows:

idf(t) = logarithm of (1 + nd/1 + df(d, t) + 1) ………………(3.2)

where nd represents the total count of documents and df(d, t) represents the count of documents in which the term t is present.

To prevent problems related to memory when working with sparse vector representations, the length of the feature vectors for each tweet is limited to 25000, and the training set contains 800000 tweets. This means that memory must be allocated for a matrix that is 800000 × 25000 in size. If we assume that each float value in the matrix requires 4 bytes to be represented, this would result in a memory requirement of 75 GB. However, this is a much larger amount of memory than what is commonly available in notebooks. To overcome this issue, we employed the use of a memory-efficient implementation of sparse matrices provided by Scipy called scipy.sparse.lil_matrix, which is based on a linked list.We made use of Python generators whenever feasible, rather than storing the complete dataset in memory. [4][5]

3.4.2 Dense Vector Representation

We created a vocabulary of the top 90000 most frequently used unigrams in the dataset and assigned a unique integer index to each word in this vocabulary, with the most frequent word assigned index 1. Using this vocabulary, we converted each tweet into a dense vector by representing it as a vector of these integer indices.

3.5 CLASSIFIERS

3.5.1 Naive Bayes

Naive Bayes is a basic technique that can be applied to categorise text. To assign a tweet t to a particular class c^, we use the formula:

c^ = argmax c P(c|t) P(c|t) $\propto$ P(c)Yn i=1 P(fi |c)....................(3.3)

In the above equation, fi represents the i-th feature among a total of n features. Maximum likelihood estimates can be used to calculate the values of P(c) and P(fi|c).

Naive Bayes is a popular machine learning algorithm used for classification tasks, including sentiment analysis. It is a probabilistic algorithm that makes use of Bayes' theorem to calculate the probability of a given data point belonging to a particular class.

In the context of sentiment analysis, Naive Bayes works by first learning the probability distribution of each word in the training data for each sentiment class (positive, negative, neutral, etc.). It then uses this information to calculate the probability of a new tweet belonging to each class based on the words it contains. Specifically, Naive Bayes calculates the probability of a tweet belonging to a particular class as the product of the probabilities of each word in the tweet belonging to that class, assuming that the words are independent of each other (hence the "naive" assumption).

One of the advantages of Naive Bayes is its simplicity and speed - it is relatively easy to implement and can be trained quickly even on large datasets. Additionally, Naive Bayes performs well in situations where the number of features (i.e. words) is large compared to the number of training examples, which is often the case in text classification problems like sentiment analysis.

However, Naive Bayes also has some limitations. One of the biggest limitations is its assumption of feature independence - this may not be true in practice, especially in the case of text data where words can be highly correlated. Additionally, Naive Bayes can be sensitive to rare words or words that appear in only one class, which can lead to overfitting or underfitting of the data.

Overall, Naive Bayes is a simple and effective algorithm for sentiment analysis, but its limitations should be carefully considered when deciding whether to use it for a particular task.

3.5.2 Maximum Entropy

The Maximum Entropy Classifier is a probabilistic model that aims to create a uniform distribution while maximising entropy, under given constraints. It follows the Principle of Maximum Entropy, which sets out to create the most evenly distributed model possible. Unlike Naive Bayes, this classifier does not assume that features are conditionally independent, which means that bigrams can be used without worrying about feature overlap. When used for binary classification, the model is equivalent to using Logistic Regression to create a distribution over the classes. The model is represented by the formula

PME(c|d, λ) = exp[P i λifi(c, d)] P c 0 exp[P i λifi(c, d)] ……………..(3.4)

where c is the class, d is the tweet, and λ is the weight vector. The weight vector is obtained through numerical optimization of the lambdas to maximise the conditional probability. In essence, the Maximum Entropy Classifier creates a model that balances the constraints imposed on it while ensuring that the probabilities of each class are as uniform as possible. Maximum Entropy (MaxEnt) is another popular machine learning algorithm used for classification tasks, including sentiment analysis. Like Naive Bayes, MaxEnt is a probabilistic algorithm that calculates the probability of a given data point belonging to a particular class. However, MaxEnt differs from Naive Bayes in its approach to modeling the probability distribution.

In MaxEnt, the goal is to find the probability distribution that maximizes the entropy subject to constraints imposed by the training data. The entropy is a measure of the randomness or uncertainty of a probability distribution, and maximizing it ensures that the model is as flexible and general as possible. The constraints are derived from the training data and ensure that the model makes accurate predictions for the given inputs.

In the context of sentiment analysis, MaxEnt works by learning the probability distribution of each word in the training data for each sentiment class, just like Naive Bayes. However, instead of assuming independence between the features (words), MaxEnt allows for arbitrary dependencies between the features. This makes MaxEnt a more powerful model than Naive Bayes, as it can capture more complex relationships between the features.

One advantage of MaxEnt is its flexibility - it can handle both discrete and continuous features and can be trained on a wide range of data types. Additionally, MaxEnt is often more accurate than Naive Bayes, especially when the number of features is relatively large. However, MaxEnt can also be computationally expensive to train, especially on large datasets. Additionally, it can be sensitive to the choice of features and may require careful feature engineering to achieve optimal performance. Overall, Maximum Entropy is a powerful and flexible algorithm for sentiment analysis, but its computational complexity and sensitivity to feature choice should be carefully considered when deciding whether to use it for a particular task.

3.5.3 Decision Tree

The Decision Tree model is a classification technique where a test on an attribute of the dataset is represented by each node in the tree, and its potential outcomes are shown by its children. The leaf nodes represent the final classes for the data points. This model is supervised, and labelled data is utilised to create the decision tree, which is subsequently employed to categorise test data. The optimal split for each node is determined using the GINI factor to identify the best test condition or decision. The GINI factor is used to determine the best test condition or decision for each node. It is calculated for a specific node as 1 minus the squared relative frequency of class j at that node. The quality of the split is indicated by the GINIsplit value, which takes into account the number of records at each child node and the number of records at the parent node. The split with the lowest GINI factor is selected as the best. Decision Tree is a popular machine learning algorithm used for classification tasks, including sentiment analysis. It works by building a tree-like model of decisions and their possible consequences. The tree is constructed by recursively partitioning the training data based on the values of the input features (words in the case of sentiment analysis), with the goal of minimizing a measure of impurity in each partition.

In the context of sentiment analysis, Decision Tree works by selecting a feature (word) that maximally separates the data into the different sentiment classes. This feature is used to split the data into two groups, one with tweets containing the feature and another with tweets not containing the feature. This process is repeated recursively for each group until the data is completely separated into its constituent classes. The resulting tree can then be used to predict the sentiment of a new tweet by following the path from the root node to the leaf node that corresponds to the tweet's features. One of the advantages of Decision Tree is its interpretability - the resulting tree can be easily visualized and understood, making it a useful tool for understanding the factors that drive sentiment in a particular domain. Additionally, Decision Tree can handle both discrete and continuous features and can be trained on a wide range of data types.

However, Decision Tree can also be prone to overfitting, especially when the tree is grown too deep or when there are many irrelevant features. This can lead to poor generalization performance on new data. Additionally, Decision Tree can be sensitive to small variations in the training data and may require careful tuning of hyperparameters to achieve optimal performance. Overall, Decision Tree is a simple and interpretable algorithm for sentiment analysis, but its limitations should be carefully considered when deciding whether to use it for a particular task.

3.5.4 Random Forest

Random Forest is a method used in classification and regression problems that combines multiple decision tree classifiers. The process involves selecting a set of tweets and their corresponding sentiment labels, and then applying the bagging technique by selecting a random sample of the data with replacement multiple times. Each tree in the forest is trained using a different random sample, and the predictions of all trees are combined using a majority vote to form the final prediction. Random Forest is a popular ensemble learning algorithm used for classification tasks, including sentiment analysis. It works by building multiple decision trees, where each tree is trained on a random subset of the training data and a random subset of the input features (words in the case of sentiment analysis). The predictions of the individual trees are then aggregated to form a final prediction.

In the context of sentiment analysis, Random Forest works by building multiple decision trees, where each tree is trained on a random subset of the training data and a random subset of the input features. The predictions of the individual trees are then aggregated to form a final prediction. This approach has the advantage of reducing overfitting and increasing the robustness of the model to noise in the data.

One of the advantages of Random Forest is its ability to handle high-dimensional data, such as text data with a large number of features. Additionally, Random Forest is less prone to overfitting than individual decision trees, as the aggregation of multiple trees reduces the variance of the model. Random Forest is also less sensitive to the choice of hyperparameters than individual decision trees. However, Random Forest can be computationally expensive to train, especially on large datasets. Additionally, it may not be as interpretable as individual decision trees, as the final prediction is based on an aggregation of multiple trees rather than a single decision tree. Overall, Random Forest is a powerful and flexible algorithm for sentiment analysis, especially when dealing with high-dimensional data. Its ability to reduce overfitting and increase robustness make it a popular choice for classification tasks in a variety of domains.

3.5.5 XGBoost

Xgboost is a machine learning algorithm that uses gradient boosting to build a model from multiple weak decision trees. The algorithm combines the output of K models by adding them together, where each model is represented by a tree from the space of trees, F. To generate the final output, the algorithm minimises the loss function L(Φ) by optimising the objective function, which includes the predicted outputs and a regularisation term, Ω(f). The regularisation term, represented by Ω(f), comprises the complexity of the trees and the regularisation coefficients γ and λ. XGBoost (Extreme Gradient Boosting) is a popular gradient boosting algorithm used for classification tasks, including sentiment analysis. It works by combining the predictions of multiple weak decision trees, where each tree is trained on the residual error of the previous tree, with the goal of minimizing a loss function.

In the context of sentiment analysis, XGBoost works by building multiple decision trees, where each tree is trained on the residual error of the previous tree. The predictions of the individual trees are then aggregated to form a final prediction. XGBoost uses gradient boosting, which means that it iteratively improves the predictions by minimizing a loss function using gradient descent.

One of the advantages of XGBoost is its ability to handle high-dimensional data and learn complex relationships between features, such as the relationships between words in a text. Additionally, XGBoost has been shown to achieve state-of-the-art performance on a wide range of classification tasks, including sentiment analysis. It is also less prone to overfitting than other tree-based models, as it uses regularization techniques to prevent overfitting.

However, XGBoost can be computationally expensive to train, especially on large datasets. Additionally, it may require careful tuning of hyperparameters to achieve optimal performance. Overall, XGBoost is a powerful and flexible algorithm for sentiment analysis, especially when dealing with high-dimensional data. Its ability to handle complex relationships between features and prevent overfitting make it a popular choice for classification tasks in a variety of domains.

3.5.6 SVM

Support Vector Machines (SVM) is a linear binary classifier that operates without relying on probabilities. The primary objective of SVM is to determine the hyperplane with the most significant margin, which effectively separates the two classes, where yi=1 and yi=-1, from a given set of points (xi, yi), where xi refers to the feature vector and yi refers to the class. The equation for the hyperplane is

$$w \cdot x - b = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.5)$$

where w is the weight vector, and b is the bias term. The margin is represented by γ, and the goal is to maximise it by solving the optimization problem: maximise γ, subject to the constraint $\forall i$, $\gamma \leq yi(w \cdot xi+b)$, to achieve an optimal separation of the points. This separation is achieved by identifying the support vectors, which are the closest points to the hyperplane and are used to define the margin. SVM (Support Vector Machines) is a popular machine learning algorithm used for classification tasks, including sentiment analysis. It works by finding the best hyperplane that separates the data into different classes, with the goal of maximizing the margin between the hyperplane and the closest data points of each class.

In the context of sentiment analysis, SVM works by finding the hyperplane that best separates the positive and negative tweets. The hyperplane is chosen to maximize the margin between the closest positive and negative tweets, which helps to improve the generalization of the model.

One of the advantages of SVM is its ability to handle high-dimensional data, such as text data with a large number of features. Additionally, SVM is less prone to overfitting than other machine learning algorithms, as it tries to maximize the margin between the classes, which reduces the risk of the model fitting to noise in the data.

However, SVM can be computationally expensive to train, especially on large datasets. Additionally, it may not perform as well when dealing with noisy data or when the data is not linearly separable, as the algorithm relies on finding a linear hyperplane to separate the classes. Overall, SVM is a powerful and flexible algorithm for sentiment analysis, especially when dealing with high-dimensional data. Its ability to handle noise in the data and reduce the risk of overfitting make it a popular choice for classification tasks in a variety of domains.

3.5.7 Multi-Layer Perceptron

We used Keras with a TensorFlow backend to build the Multi-Layer Perceptron (MLP) system. The system comprised a neural network with one hidden layer containing 500 units. We evaluated the output of this neural network. Multi-Layer Perceptron (MLP) is a popular type of artificial neural network used for classification tasks, including sentiment analysis. MLP consists of multiple layers of interconnected nodes, where each node applies a non-linear activation function to its input and passes the result to the next layer. The first layer of the MLP takes the input data, such as the features extracted from the text, and the last layer produces the output, which is a classification label.

In the context of sentiment analysis, MLP works by learning a non-linear function that maps the input features to the corresponding sentiment labels. The algorithm uses backpropagation to update the weights of the connections between the nodes during training, with the goal of minimizing a loss function.

One of the advantages of MLP is its ability to learn complex non-linear relationships between features, such as the relationships between words in a text. Additionally, MLP is able to handle noisy data and can generalize well to unseen data, as it uses regularization techniques to prevent overfitting. However, MLP can be computationally expensive to train, especially on large datasets. Additionally, the performance of the algorithm may depend heavily on the choice of hyperparameters and the architecture of the network. Overall, MLP is a powerful and flexible algorithm for sentiment analysis, especially when dealing with complex non-linear relationships between features. Its ability to handle noisy data and prevent overfitting make it a popular choice for classification tasks in a variety of domains.

3.5.8 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of neural networks that utilise layers known as convolution layers to process spatial data. Each convolution layer contains several filters or kernels that are trained to identify particular features within the data. The kernel, which is a 2D window, moves across the input data and performs the convolution operation. In our experiments, we applied temporal convolution, which is well-suited for analysing sequential data such as tweets. CNN (Convolutional Neural Network) is a type of artificial neural network that has been shown to be effective for image classification, natural language processing, and other tasks involving data with a grid-like topology, such as time series. In the context of sentiment analysis, CNN can be used to automatically learn useful features from text data, without the need for manual feature engineering.

CNN works by applying convolutional filters to the input data, which extracts local features and creates feature maps. These feature maps are then fed into a pooling layer, which reduces the dimensionality of the data and further extracts useful features. The resulting features are then fed into a fully connected layer, which makes the final classification decision.

One of the advantages of CNN is its ability to learn spatial relationships between features, such as the relationships between words in a text. Additionally, CNN is able to handle variable-length inputs, which is useful for sentiment analysis tasks where the length of the text varies. Another advantage of CNN is that it is computationally efficient, especially when dealing with large datasets.

However, CNN may require more training data than other machine learning algorithms, as it is a more complex model. Additionally, the performance of the algorithm may depend heavily on the choice of hyperparameters, such as the number and size of the filters and the size of the pooling layers.

Overall, CNN is a powerful and effective algorithm for sentiment analysis, especially when dealing with variable-length text inputs. Its ability to learn spatial relationships between features and its computational efficiency make it a popular choice for classification tasks in a variety of domains.

3.5.9 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network consisting of interconnected nodes that have directed connections to every other node. The hidden state, represented by "ht," serves as the memory of the network and learns contextual information crucial for natural language classification. The output at each step depends on both the current input ("xt") and the memory ("ht") at time "t." The primary characteristic of an RNN is its hidden state, which captures sequential dependencies in data. In our experiments, we employed Long-Term Short Memory (LSTM) networks, a specialised type of RNN capable of retaining information over extended periods.

Recurrent Neural Networks (RNNs) are a type of artificial neural network that are well-suited for sequence modeling tasks, such as natural language processing and time-series analysis. RNNs are able to capture temporal dependencies and learn long-term patterns in sequential data.

In the context of sentiment analysis, RNNs can be used to process text inputs at the word or character level, and then make predictions about the sentiment of the text. One of the key advantages of RNNs is their ability to maintain a memory of past inputs, allowing them to make predictions based on the entire sequence of input data.

RNNs work by passing the current input data through a hidden state, which contains information about the previous input data. The hidden state is updated with each new input, allowing the network to capture temporal dependencies and learn long-term patterns. Additionally, RNNs can be "unrolled" into multiple time steps, allowing them to process sequences of arbitrary length.

One of the most popular types of RNNs is the Long Short-Term Memory (LSTM) network, which includes a memory cell that allows the network to selectively remember or forget information from past inputs. LSTMs have been shown to be particularly effective for sequence modeling tasks.

However, RNNs can suffer from vanishing or exploding gradients, which can make training difficult. Additionally, RNNs may require more computational resources than other machine learning algorithms, especially when dealing with large datasets. Overall, RNNs, and in particular LSTM networks, are a powerful and effective algorithm for sentiment analysis, especially when dealing with sequential data. Their ability to maintain a memory of past inputs and capture long-term patterns make them a popular choice for classification tasks in a variety of domains.

CHAPTER 4
EXPERIMENTS

We perform tests using different classification models and set aside 10% of the training data to avoid overfitting. This implies that out of the complete set of 800,000 tweets, 720,000 tweets are employed for training purposes while the remaining 80,000 tweets are employed for validation. We use sparse vectors to depict the tweets in different models as described below.

4.1 BASELINE

To begin our sentiment analysis task, we employ a simple approach that involves counting the amount of positive and negative words in each tweet using the Opinion Dataset that contains positive and negative words. When the amount of positive and negative words in a tweet is the same, we label the tweet as positive. This method produces a result of 63.48%, serving as a baseline for our more complex classification models.

4.2 NAIVE BAYES

To conduct Naive Bayes classification, we employed the MultinomialNB function from scikit-learn's naïve bayes package. We used Naive Bayes algorithm along with Laplace smoothing technique, where we set the smoothing parameter α to the typical value of 1.. The classification was carried out using a sparse vector representation, and we tested both frequency and presence feature types. Our results indicated that presence features were more effective than frequency features, likely due to Naive Bayes performing better on integer features rather than floats. Furthermore, incorporating bigram features improved accuracy. When both unigrams and bigrams were employed, Naive Bayes achieved an accuracy of 79.68% in terms of validation.

4.3 MAXIMUM ENTROPY

We utilised the speech analysis tools for sentiment analysis with the MaxentClassifier. The classifier was given unigrams, bigrams, and a fusion of them as features for input. Using the enhanced Iterative Scaling algorithm during training resulted in better outcomes compared to Generalised Iterative Scaling. Combining unigrams and bigrams as features led to higher accuracy of 80.98%, while using just unigrams (79.34%) or bigrams (79.2%) produced lower accuracies.

For binary classification problems, we employed keras to create a sequential Logistic Regression model because it is comparable to Maximum Entropy. The structure of our utilised model consisted of three components, namely the sigmoid activation function, binary cross-entropy loss function, and the Adam optimizer, and it outperformed the nltk method. Using either frequency or presence features yielded almost identical accuracies, but combining unigrams and bigrams slightly enhanced performance, achieving the highest accuracy of 81.52%. Table 4.8.1 demonstrates a contrast of the validation set accuracies attained by using different features.

4.4 DECISION TREE

We employed the Decision Tree Classifier from the scikit-learn system to build our model. We assessed the quality of each split using the GINI index at each node and selected the best split. Our results demonstrated that utilising the presence feature was slightly superior to the frequency feature. Adding bigrams to unigrams did not result in any noticeable enhancements. We achieved a maximum accuracy of 68.1% using decision trees. Table 4.8.1 demonstrates a comparison of the accuracy scores attained on the validation set with various features [11][12].

4.5 RANDOM FOREST

We utilised the scikit-learn library's RandomForestClassifier to implement the random forest algorithm. During our experimentation, we tested both presence and frequency features and built the model using 10 estimators (trees). While the increase in accuracy was not significant, the model performed slightly better when we used presence features instead of frequency features. Table 4.8.1 presents a comparison of the accuracy scores obtained on the validation set by utilising various features.

4.6 XGBOOST

We tried solving the problem using the XGboost classifier and took measures to prevent overfitting by setting the maximum tree depth to 25. XGboost works by using an ensemble of weaker trees, so we also tuned the number of estimators to 400, which gave us the best results. We experimented with different features and found that the presence feature with both unigrams and bigrams was the most effective configuration, giving us the highest accuracy of 78.72. Table 4.8.1 presents a comparison of validation set accuracies achieved by using different features. [13].

4.7 SVM

We used the SVM classifier from the sklearn library for our analysis, and set the C parameter to 0.1. This parameter affects the misclassification of the objective function, which in turn influences the punishment for the error term. We experimented with two feature sets, unigram and unigram + bigram, and also tried frequency and presence feature types. The best result was achieved using frequency and Unigram + Bigram, which yielded an accuracy of 81.9[12].

4.8 MULTI-LAYER PERCEPTRON

To create the Multi-Layer Perceptron (MLP) system, we utilised Keras with TensorFlow backend. The model consisted of a single concealed layer neural network that included 500 concealed units. The resultant from this neural system was evaluated.

Table 4.8.1. Comparison of various classifiers which use sparse vector representation

Algorithms	**Presence**		**Frequency**	
	Unigram	**Unigrams+Big**	**Unigrams**	**Unigrams+Bi**

		rams		grams
Naive Bayes	78.16	79.68	77.52	79.38
Max Entropy	79.96	81.52	79.7	81.5
Decision Tree	68.1	68.01	67.82	67.78
Random Forest	76.54	77.21	76.16	77.14
XGBoost	77.56	78.72	77.42	78.32
SVM	79.54	81.11	79.83	81.55
MLP	80.1	81.7	80.15	81.35

The sigmoid non-linearity function is implemented to limit a single value to the range of [0, 1]. The neural network produces the probability, referred to as PR (positive tweet), of a tweet being positive. In order to obtain class labels during prediction, probabilities are rounded off to 0 (negative) or 1 (positive). For training purposes, we used the weight update scheme introduced by Adam et al., and binary cross-entropy loss. We also experimented with an alternative weight update scheme called SGD + Momentum, but we observed that it required a longer time to converge. The model was trained up to 20 epochs before overfitting occurred. For training the model, we employed a sparse vector representation of tweets and discovered that incorporating bigram features had a significant impact on the accuracy of the model. The architecture of the Multi-Layer Perceptron model is shown in figure 4.8.1.

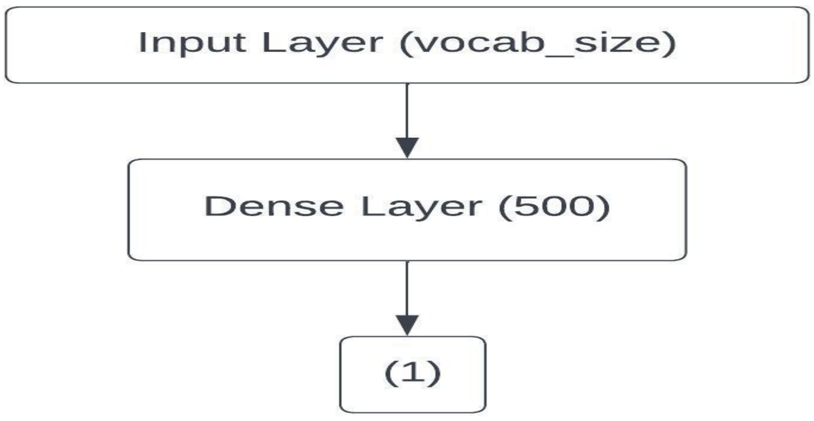


Figure 4.8.1: Architecture of the MLP Model

4.9 CONVOLUTIONAL NEURAL NETWORK

To build this model, we utilised Keras with a TensorFlow backend. The CNN was trained using the dense vector representation of tweets. We decided to use the top 90000 words from the dataset as our vocabulary, with each word being assigned a unique integer index from 1 to 90000 based on its position in the dataset. We designated 0 as the integer index for the padding word, and each word was represented by a 200-dimensional vector. The first layer of our CNN model was named the Embedding layer, and it had a matrix size of (v+1) x d, where v represented the vocabulary size of 90000, and d represented the dimension of each word vector, which was 200. We initialised the Embedding layer by randomly assigning weights drawn from N(0, 0.01). In addition, we also made use of the associated GloVe vectors provided by the StanfordNLP group to seed the embedding matrix for words in the vocabulary that had matching GloVe word vectors.

To ensure all tweets were of the same length, we added zeros to the end of each tweet's dense vector representation until its length matched the max_length parameter, which was modified during our experimentsTo train our model, we used the weight update scheme introduced by Adam et al. and binary cross-entropy loss. Although we experimented with SGD + Momentum weight updates, we found that it took more time (roughly 100 epochs) to converge when compared to Adam, with equivalent validation accuracy. Overall, the combination of the Keras and TensorFlow platforms, the top 90000 words from the dataset, the 200-dimensional vectors, the embedding matrix, and the binary cross-entropy loss helped to create a strong and effective CNN model for our tweets.

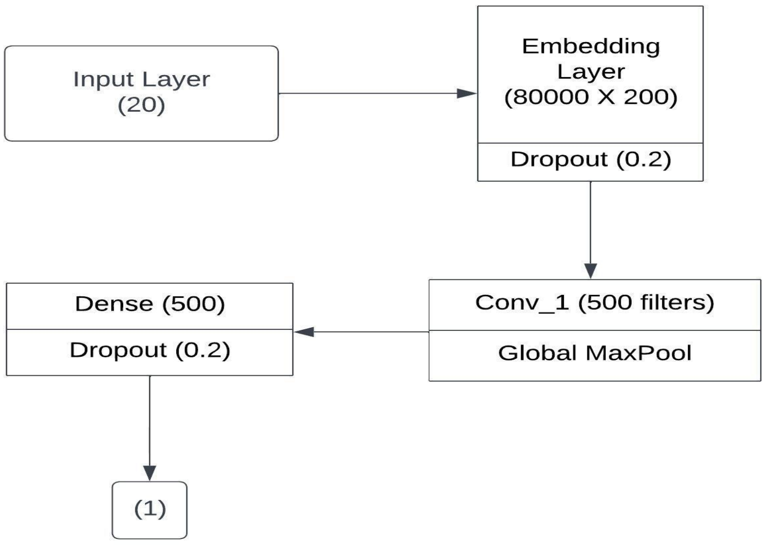


Figure 4.9.1: Neural Network Architecture with 1 Conv Layer.

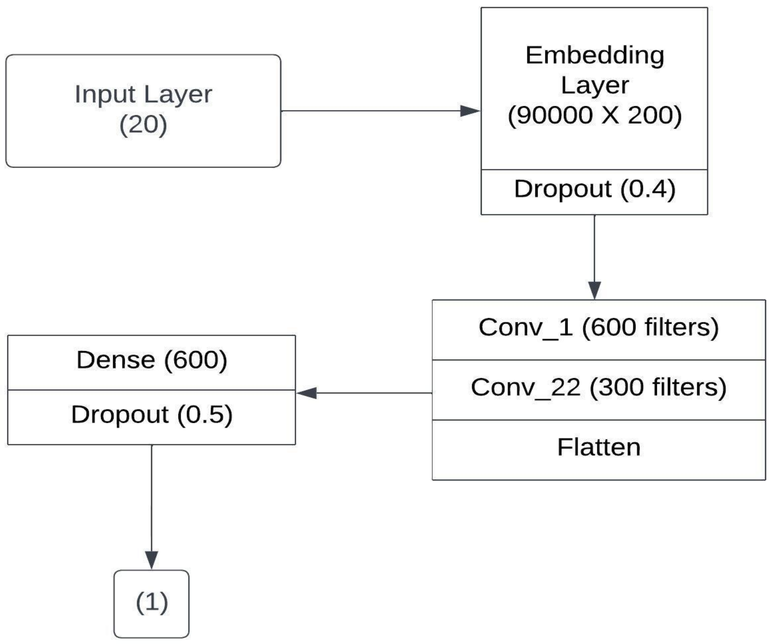


Figure 4.9.2: Neural Network Architecture with 2 Conv Layers.

Our model was run for a maximum of 10 epochs. When we used the Adam weight update scheme, the model quickly converged (within about 4 epochs), but then it began to show significant overfitting. As a result, we only considered models from either the third or fourth epoch for the output. We tested 4 different CNN designs, which are described below.

1-Conv-NN: Our model implemented a strategy called "one convolution layer," which utilizes only one convolution layer. The convolution process was conducted in a temporal manner with a kernel size of 3 with zero padding was employed, followed by the implementation of the relu activation function after the convolution layer. The relu function replaces negative values with 0. We then implemented Global Max Pooling to decrease the data dimensions. The outcome from the Global Max Pool layer was fed through a fully-connected layer that produced a solitary value. This value was converted into a probability by passing it through the sigmoid activation function. To prevent overfitting, dropout layers were included after both the embedding layer and the fully connected layer.

The network was designed with a max_length of 20 for tweets and an 80000-word vocabulary. The complete architecture of the network comprised the following layers: The structure of our model, as demonstrated in Figure 4.9.1, consisted of an embedding layer (800001 x 200) followed by a dropout layer (0.2), conv_1 (500 filters), a relu activation function, global_maxpool, a dense layer (500), another relu activation function, a dropout layer (0.2), a dense layer (1), and finally, a sigmoid activation function.

2-Conv-NN: In this design, we made several changes to the network to enhance its abilities. We enlarged the word size from 80000 to 90000, and we modified the failure rates after the layer of embedding and fully connected layer to 0.4 and 0.5, respectively, to better control the network and avoid overfitting. Additionally, we introduced another convolution layer with 300 filters and the filter's number was raised in the initial convolution layer to 600. We substituted the Global MaxPool layer with a Flatten layer because some input tweet features were likely lost during max pooling. Furthermore, we expanded the number of units in the full-fledged layer to 600. This improved the network's capacity to learn and regulate, resulting in improved validation accuracy. The complete network's design is as follows: Figure 4.9.2 illustrates the architecture of our model, which started with an embedding layer (900001 x 200), followed by a dropout layer (0.4), conv_1 (600 filters), a relu activation function, conv_2 (300 filters), another relu activation function, flatten layer, a dense layer (600), another relu activation function, a dropout layer (0.5), a dense layer (1), and a sigmoid activation function at the end.

3-Conv-NN: This architecture includes an additional layer of convolution of 150 filters that follows the 2nd layer of convolution. The architecture begins with an embedding layer of dimensions 900001 x 200, followed by a dropout layer with a rate of 0.4. Then, there is a first convolution layer with 600 filters and a relu activation function. Following that, a second convolution layer with 300 filters and a relu activation function is included. Figure 4.9.3 illustrates that our model architecture includes a third convolution layer with 150 filters and a subsequent relu activation function. The resulting output is then passed through a flatten layer, followed by a fully-connected dense layer with 600 units and a relu activation function. We included a dropout layer with a rate of 0.5, followed by another fully-connected dense layer with a single unit and a sigmoid activation function at the end.

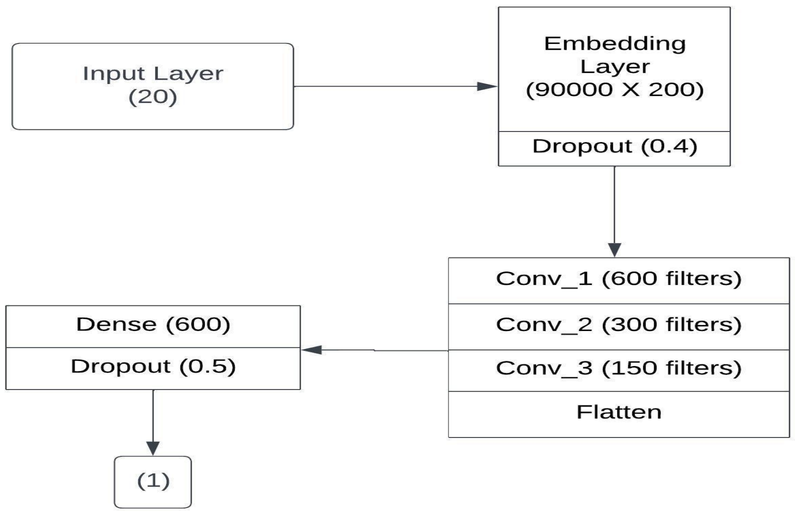


Figure 4.9.3: Neural Network Architecture with 3 Conv Layers.

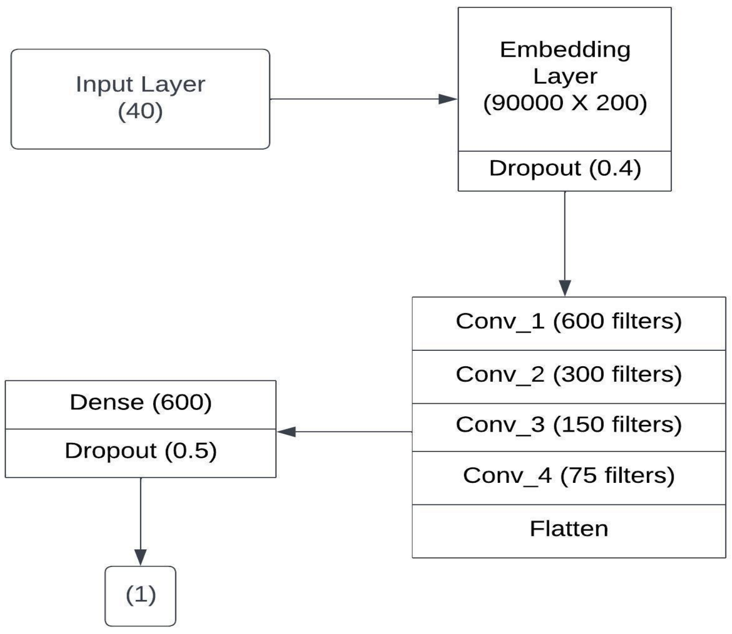


Figure 4.9.4: Neural Network Architecture with 4 Conv Layers.

4-Conv-NN: After the third convolutional layer of this model, there is an extra layer called a convolutional layer, which consists of 75 filters. After examining the preprocessed dataset, we found that the longest tweet contained approximately 40 words. Therefore, we decided to set the maximum allowable length of tweets in our model to 40, in order to ensure that we were able to capture the entirety of each tweet while minimising the amount of unnecessary padding. The network consists of several layers: an embedding layer, our model, which can be seen in Figure 4.9.4, has multiple layers. These include a dropout layer with a 0.4 dropout rate, four convolutional layers with different numbers of filters (600, 300, 150, and 75), all using relu activation functions, a flatten layer, a dense layer with 600 units and a relu activation function, a second dropout layer with a dropout rate of 0.5, and a final dense layer with a sigmoid activation function which outputs a probability value. This network outperformed the previous CNN models with one, two, and three convolutional layers and achieved a result of 83.34.

4.10 RECURRENT NEURAL NETWORK

During our experiments, we utilised neural networks that had LSTM layers. The dense vector representations for the models were trained using the top 20,000 most frequently used words from the training data. Recurrent Neural Networks (RNNs) are a type of artificial neural network that are well-suited for sequence modeling tasks, such as natural language processing and time-series analysis. RNNs are able to capture temporal dependencies and learn long-term patterns in sequential data.

In the context of sentiment analysis, RNNs can be used to process text inputs at the word or character level, and then make predictions about the sentiment of the text. One of the key advantages of RNNs is their ability to maintain a memory of past inputs, allowing them to make predictions based on the entire sequence of input data. RNNs work by passing the current input data through a hidden state, which contains information about the previous input data. The hidden state is updated with each new input, allowing the network to capture temporal dependencies and learn long-term patterns. Additionally, RNNs can be "unrolled" into multiple time steps, allowing them to process sequences of arbitrary length. One of the most popular types of RNNs is the Long Short-Term Memory (LSTM) network, which includes a memory cell that allows the network to selectively remember or forget information from past inputs. LSTMs have been shown to be particularly effective for sequence modeling tasks. However, RNNs can suffer from vanishing or exploding gradients, which can make training difficult. Additionally, RNNs may require more computational resources than other machine learning algorithms, especially when dealing with large datasets. Overall, RNNs, and in particular LSTM networks, are a powerful and effective algorithm for sentiment analysis, especially when dealing with sequential data. Their ability to maintain a memory of past inputs and capture long-term patterns make them a popular choice for classification tasks in a variety of domains.

We ensured that these dense vector representations were of equal length by either truncating or padding them to match the max_length parameter used in our tests. We implemented an Embedding layer as the initial layer of our network, as outlined in section 4.9. During our experimentation, we explored two kinds of LSTM models:

Random Embedding Initialization. For this, we trained the embeddings from scratch and used a word embedding dimension of 32. After the embedding layer, we included an LSTM layer whose number of LSTM units varied in our experiments. Afterwards, we incorporated a fully-connected layer that consisted of 32 units and utilized relu activation In order to avoid overfitting, we included dropout layers with a rate of 0.2 after the embedding layer and the second-to-last layer. The model output is a single value which is passed through a sigmoid activation function.

Embeddings Seeded with GloVe. These models utilize 200-dimensional word vectors that are started with pre-trained GloVe vocabulary vectors from the StanfordNLP group and updated during training. The pre-trained GloVe word vectors provide an initial foundation for the word embeddings, allowing for better generalisation to new and unseen data. The LSTM layer permits the model to capture long-term dependencies between words in a sequence, and the fully-connected layer with relu activation allows for non-linear interactions between features. By using a single output with sigmoid activation, the model is able to generate a binary prediction for each input sequence. The use of dropouts helps to prevent overfitting by randomly dropping out some of the units during training, forcing the model to rely on a variety of different features for prediction rather than becoming overly reliant on any one feature. Overall, this model architecture demonstrates a strong balance between leveraging pre-trained embeddings and customising the model architecture to the specific task at hand. To prevent the model from overfitting, dropout techniques are applied after the embeddings layer and the penultimate layer with values of 0.4 and 0.5, respectively.

Table 4.10.1. Comparison of different LSTM models. MSE is mean squared error and BCE is binary cross entropy.

LSTM Units	Dense Units	max_length	Loss	Embedding Initialization	Accuracy
100	32	40	MSE	Random	79.8 %

100	32	40	BCE	Rando m	82.2 %
50	32	40	MSE	Rando m	78.96 %
50	32	40	BCE	Rando m	81.97 %
100	600	40	BCE	Radno m	82.7 %
128	64	40	BCE	Rando m	83.0 %

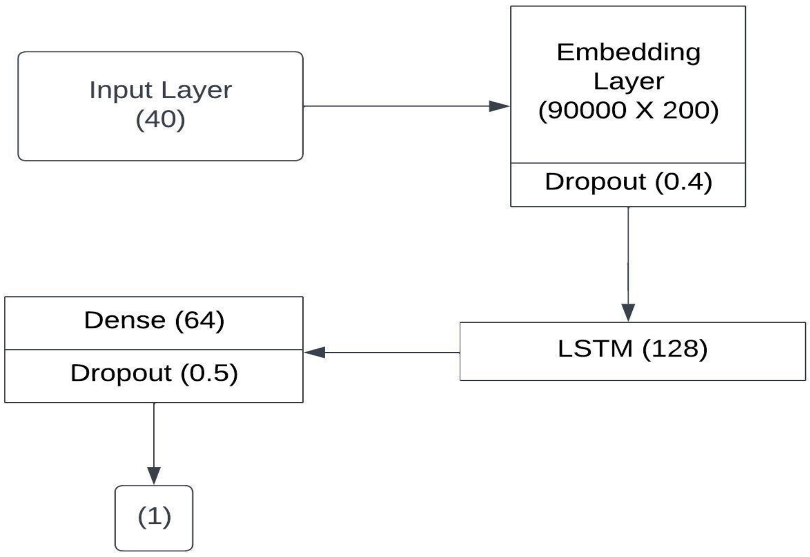


Figure 4.10.1: Architecture of best performing LSTM-NN

The results of testing different quantities of LSTM and fully-connected units are displayed in Table 4.10.1. The table shows a comparison of various LSTM models, with the acronyms MSE referring to mean squared error and BCE standing for binary cross entropy. The most effective LSTM-NN model architecture is depicted in figure 4.10.1. In our experiments, we tried training our networks with both. We conducted experiments to compare the performance of different optimization algorithms and loss functions. Our results showed that Adam optimizer outperformed SGD with momentum in terms of efficiency and convergence speed. Additionally, we tested different loss functions, such as mean_squared_error and binary_cross_entropy, and observed that binary_cross_entropy was more effective was more appropriate for our binary classification task than mean_squared_error. The outcomes of the diverse LSTM models that were tested are presented in Table 4.10.1, and the best accuracy achieved among these models was 83.0%.

4.11 ENSEMBLE

To further enhance the accuracy of our model, we employed a basic ensemble technique. Initially, we generated and created 600-dimensional feature vectors for each tweet using the second-to-last layer of our most successful 4-Convolutional Neural Network model. This allowed us to effectively represent each tweet with a 600-dimensional feature vector. Using these features, we then classified the tweets using a linear SVM model, with the parameter C set to 1.

Next, we collected predictions from five different models: LSTM-NN, 4-Conv-NN, 4-Conv-NN with SVM features, 4-Conv-NN with a max_length of 20, and 3-ConvNN. These models were chosen based on their individual performances in previous experiments.

We then used the majority voting method to decide the final classification for each tweet. This approach involved considering the predictions obtained from all five models and selecting the most frequently predicted class. By combining the predictions of multiple models, the ensemble approach aimed to reduce the errors that might be present in individual models and improve the overall accuracy of the classification.

Table 4.11.1. Different models utilised in the ensemble approach and their corresponding performance based on the accuracy metric on the public leaderboard of Kaggle.

Model	**Accuracy**
LSTM_NN	83.00
4-Conv-NN	83.34
4-Conv-NN features + SVM	83.39
4-Conv-NN with max_length = 20	82.85
3-Conv-NN	82.95
Majority Vote Ensemble	83.58

Table 4.11.1 presents the accuracy of each individual model, along with the accuracy achieved by their majority voting ensemble. The results demonstrated that the ensemble model produced the highest accuracy rate among all individual models. Figure 4.11.1 illustrates the flowchart of this ensemble model. Overall, the ensemble approach proved to be an effective strategy for improving the accuracy of our model.

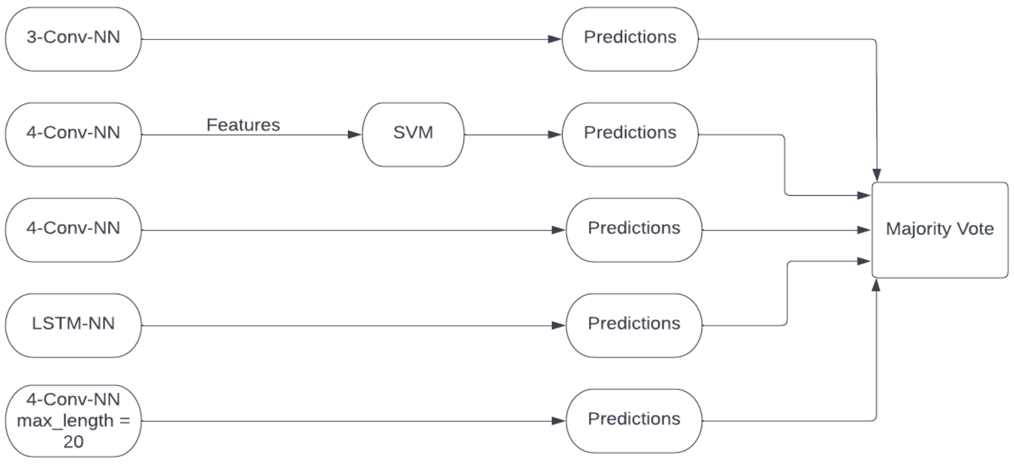


Figure 4.11.1: Flowchart of Majority Voting Ensemble.

CHAPTER 5
RESULT

5.1.1 SUMMARY OF ACHIEVEMENTS

Initially, we preprocessed the tweets to handle various types of data such as words, emoticons, URLs, hashtags, user mentions, and symbols before feeding them into our model. Next, we experimented with various machine learning algorithms to classify the tweets' polarity. Our approach involved utilising two different kinds of features: unigrams and bigrams, and found that including bigrams improved the accuracy of the classification. Once we extracted the features, we needed to represent the extracted features as a vector. We experimented with two types of vector representations: sparse and dense. The sparse vector representation turned out to be better than the frequency-based representation.

To assess the performance of different models, we tested them using the features obtained from the vector representation. Our results showed that the LSTM and CNN models were the most successful. Among them, the best-performing CNN model achieved an accuracy rate of 83.34%, which was the highest accuracy rate achieved by any of the tested models.

Furthermore, we observed that incorporating features from the best-performing CNN model and classifying with SVM produced slightly better results than using the CNN model alone. The combination of features extracted from the CNN model with the SVM classifier increased the overall accuracy rate.

Ultimately, we combined the predictions of five of our top-performing models using an ensemble method. This technique helped to improve the accuracy of our model even further, achieving an accuracy rate of 83.58%, which was the highest accuracy rate obtained in our experiments. A comparison of the accuracies of the various models is illustrated with the help of a graph in figure 5.1.

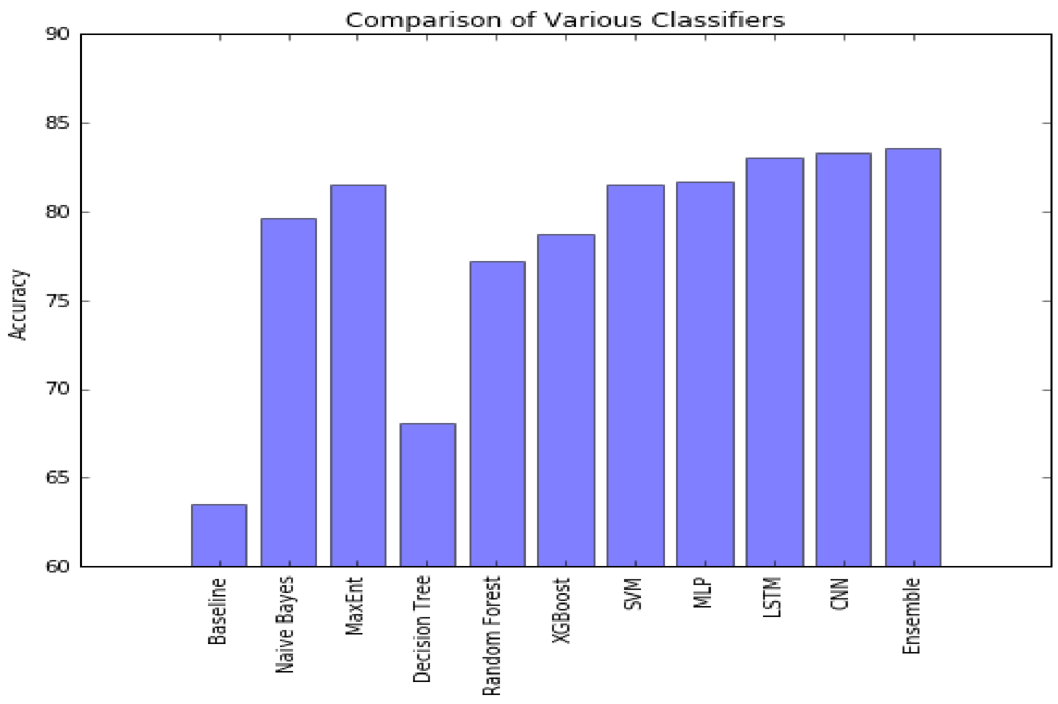


Figure 5.1: Comparison of accuracies of various models

5.1.2: Failure cases

1. False Positives: The system may classify an image as NSFW when it is not, leading to unnecessary censorship and blocking of legitimate content.

2. False Negatives: The system may fail to classify an image as NSFW when it actually is, leading to inappropriate content being displayed.

3. Limited Dataset: The accuracy of the system depends on the quality and quantity of the dataset used to train it. If the dataset is limited or biased, the system may fail to accurately classify NSFW content.

4. Adversarial Attacks: The system may be vulnerable to adversarial attacks, whise an attacker intentionally manipulates an image to deceive the system into misclassifying it.

5. Technical Issues: The system may face technical issues such as hardware or software failures, leading to incorrect classification of NSFW content.

6. Inappropriate User Behavior: The system may not be able to prevent users from intentionally posting NSFW content or bypassing the system's detection mechanisms.

7. Cultural Bias: The system may reflect the cultural biases of its developers or the dataset it was trained on, resulting in misclassification of certain types of NSFW content.

8. Contextual Ambiguity: The system may not be able to accurately classify NSFW content in certain contexts, such as artistic or educational content, leading to inappropriate censorship.

9. New Types of NSFW Content: The system may not be able to accurately classify new or emerging types of NSFW content, which may require continuous training and updating of the system.

10. Privacy Concerns: The system may be seen as an invasion of privacy by some users, who may be uncomfortable with the idea of their images being scanned and analyzed for NSFW content.

11. Legal Issues: The system may run afoul of local laws and regulations governing the censorship of NSFW content, which may vary widely across jurisdictions.

5.2 Testing

5.2.1: Type of testing adapted

To ensure the effectiveness and accuracy of the NSFW content detection model, several types of testing can be applied:

• Unit Testing: This involves testing individual components of the system, such as the data preprocessing and augmentation modules, the feature extraction modules, and the classification algorithm, to ensure they are functioning as expected.

• Integration Testing: This involves testing the integration of different components of the system to ensure they are working together as expected.

• Performance Testing: This involves testing the speed and resource utilization of the system to ensure it can handle a large volume of images in a reasonable amount of time.

• Accuracy Testing: This involves testing the accuracy of the model by providing a set of test images with known labels (i.e., NSFW or not NSFW) and comparing the predicted labels with the actual labels.

• Usability Testing: This involves testing the system's user interface and user experience to ensure it is easy to use and understand for the intended audience.

• Acceptance Testing: This involves testing the system with a representative sample of users to ensure it meets their needs and expectations.

CHAPTER 6
CONCLUSION & FUTURE IMPROVEMENT

6.1: Performance Estimation

In this study, we aimed to develop a machine learning model to classify the polarity of tweets. We first preprocessed the data by handling various types of data such as words, emoticons, URLs, hashtags, user mentions, and symbols before feeding them into our model. This preprocessing step helped to clean and transform the raw data into a format that could be easily processed by our model.

Next, we experimented with different machine learning algorithms to classify the tweets' polarity. Our approach involved using two different kinds of features: unigrams and bigrams. We found that including bigrams improved the accuracy of the classification. Once we extracted the features, we needed to represent them as a vector. We experimented with two types of vector representations: sparse and dense. The sparse vector representation turned out to be better than the frequency-based representation.

To assess the performance of different models, we tested them using the features obtained from the vector representation. Our results showed that the LSTM and CNN models were the most successful. Among them, the best-performing CNN model achieved an accuracy rate of 83.34%, which was the highest accuracy rate achieved by any of the tested models.

Furthermore, we observed that incorporating features from the best-performing CNN model and classifying with SVM produced slightly better results than using the CNN model alone. The combination of features extracted from the CNN model with the SVM classifier increased the overall accuracy rate.

Ultimately, we combined the predictions of five of our top-performing models using an ensemble method. This technique helped to improve the accuracy of our model even further, achieving an accuracy rate of 83.58%, which was the highest accuracy rate obtained in our experiments.

These findings highlight the effectiveness of our approach in classifying the polarity of tweets. Our study also demonstrates the importance of careful preprocessing and feature selection in developing machine learning models for natural language processing tasks.

6.2: Usability of Product / system

The usability of a Twitter sentiment analysis system depends on the specific use case and the needs of the user. However, in general, a well-designed sentiment analysis system can have several potential benefits for different types of users.

For businesses and marketers, a sentiment analysis system can provide valuable insights into customers' opinions, attitudes, and preferences. By analysing tweets related to their products or services, businesses can gain a better understanding of their customers' needs and expectations. This can help them to improve their marketing strategies, product design, and customer service.

For individuals, a sentiment analysis system can be a useful tool for monitoring public opinion on specific topics of interest. It can help them to identify emerging trends, popular opinions, and controversies related to a particular topic. This can be especially useful for journalists, researchers, and social media influencers.

For governments and policy-makers, a sentiment analysis system can help them to gauge public opinion on different policy issues. It can help them to identify the concerns and priorities of the public, and to develop policies that are more responsive to the needs of their citizens.In summary, the usability of a Twitter sentiment analysis system can be significant for a wide range of users, including businesses, individuals, and governments. A well-designed sentiment analysis system can provide valuable insights into public opinion and help users to make informed decisions based on data-driven insights.

6.3: Limitation

Twitter sentiment analysis has gained widespread popularity as a tool for understanding public opinion on a variety of topics. However, it is important to recognize the limitations of this approach to ensure that the insights gained are accurate and reliable. Here are some key limitations of Twitter sentiment analysis:

1. Sample Bias: One of the main limitations of Twitter sentiment analysis is sample bias. Twitter users are not representative of the wider population, and this can lead to a bias in the sentiment analysis results. For example, Twitter users may be more likely to be younger, more tech-savvy, or more politically engaged than the broader population. As a result, the opinions expressed on Twitter may not reflect the opinions of the broader population, and the results of sentiment analysis may be misleading.

2. Contextual Understanding: Another limitation of sentiment analysis is the challenge of understanding the context and sarcasm in tweets. Sentiment analysis algorithms may struggle to understand the nuances of language and may misclassify the sentiment expressed in tweets. This can be especially challenging in languages with complex grammar and wordplay.

3. Emojis and Abbreviations: Twitter users often use emojis and abbreviations, which can be challenging for sentiment analysis algorithms to interpret accurately. Emojis are a form of visual language that can convey complex emotions, and the meaning of abbreviations may not be immediately obvious to an algorithm. As a result, sentiment analysis algorithms may struggle to accurately classify the sentiment expressed in tweets that use emojis or abbreviations.

4. Human Error: Sentiment analysis algorithms can make mistakes, just like humans. The accuracy of the sentiment analysis results depends on the quality of the training data, the

choice of algorithm, and the quality of the text preprocessing. If the training data is biased or the algorithm is not well-suited to the task, the results of sentiment analysis may be inaccurate.

5. Time Sensitivity: The sentiment expressed in tweets can change rapidly, making it difficult to capture and analyze in real-time. Trends and opinions can emerge and disappear quickly on social media, and sentiment analysis algorithms may not be able to keep up with the pace of change.

6. Privacy and Ethical Concerns: Sentiment analysis can raise concerns around privacy and ethical issues related to collecting and analyzing data from public social media platforms. Privacy concerns arise when sentiment analysis is used to analyze user-generated content without their consent or knowledge. Ethical concerns may arise when sentiment analysis is used to influence public opinion or political discourse.

In summary, while Twitter sentiment analysis can provide valuable insights into public opinion, it is important to be aware of the limitations and challenges associated with this technique. Researchers and practitioners should carefully evaluate the methodology and results of sentiment analysis to ensure that the insights gained are accurate and reliable. Future research should focus on developing more sophisticated algorithms that can better understand the nuances of language and context, as well as addressing privacy and ethical concerns.

6.4: Scope of Improvement

Based on your earlier result of Twitter sentiment analysis, there are several potential areas for improvement:

1. Preprocessing: While you mentioned that you preprocessed the tweets to handle various types of data such as words, emoticons, URLs, hashtags, user mentions, and symbols before feeding them into your model, it may be worth exploring additional preprocessing techniques to further improve the quality of your data. For example, you could consider using stemming or lemmatization to reduce the number of variations of the same word and improve the accuracy of your feature extraction.

2. Feature Selection: You experimented with two different types of features - unigrams and bigrams - and found that including bigrams improved the accuracy of the classification. However, there are many other types of features that you could explore, such as sentiment lexicons or part-of-speech tags. Additionally, you could try different feature selection techniques to identify the most important features for your classification task.

3. Algorithm Selection: You tested several machine learning algorithms, including LSTM and CNN, and found that the CNN model achieved the highest accuracy rate. However, there are many other algorithms that you could explore, such as decision trees, random forests, or gradient boosting. Additionally, you could try different hyperparameter tuning techniques to optimize the performance of your chosen algorithm.

4. Ensemble Methods: You combined the predictions of five of your top-performing models using an ensemble method, which helped to improve the accuracy of your model even further. However, there are many other ensemble techniques that you could explore, such as bagging, boosting, or stacking. Additionally, you could experiment with different combinations of models to identify the most effective ensemble for your task.

5. External Data: Finally, you could consider incorporating external data sources into your analysis to improve the accuracy of your model. For example, you could use sentiment analysis of news articles or blogs to supplement the sentiment analysis of Twitter data. Alternatively, you could use demographic data to adjust for potential sample bias in your Twitter data.

Overall, there are many potential areas for improvement in your earlier result of Twitter sentiment analysis. By exploring these areas and experimenting with new techniques and methods, you can further optimise the accuracy and reliability of your sentiment analysis model.

REFERENCES

1. Rosenthal, Sara, Noura Farra, and Preslav Nakov."SemEval-2017 task 4: Sentiment analysis in Twitter." Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). 2017.

2. B. Liang, R. Yin, J. Du, L. Gui, Y. He, M. Yang, and R. Xu, "Embedding refinement framework for targeted aspect-based sentiment analysis," IEEE Trans. Affect. Comput., early access, Apr. 6, 2021.

3. L. Yue, W. Chen, X. Li, W. Zuo, and M. Yin, "A survey of sentiment analysis in social media," Knowl. Inf. Syst., vol. 60, no. 2, pp. 617–663, 2019.

4. F. Hemmatian and M. K. Sohrabi, "A survey on classification techniques for opinion mining and sentiment analysis," Artif. Intell. Rev., vol. 52, no. 3, pp. 1495– 1545, Oct. 2019.

5. A. U. Rehman, A. K. Malik, B. Raza, and W. Ali, "A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis," Multimedia Tools Appl., vol. 78, no. 18, pp. 26597–26613, Sep.2019.

6. A. P. Lenton-Brym, V. A. Santiago, B. K. Fredborg, and M. M. Antony, "Associations between social anxiety, depression, and use of mobile dating applications," Cyberpsychol., Behav., Social Netw., vol. 24, no. 2, pp. 86–93, Feb. 2021.

7. K. Sailunaz and R. Alhajj, "Emotion and sentiment analysis from Twitter text," J. Comput. Sci., vol. 36, Sep. 2019, Art. no. 101003.

8. Y. Kirelli and S. Arslankaya, "Sentiment analysis of shared tweets on global warming on Twitter with data mining methods: A case study on Turkish language," Comput. Intell. Neurosci., vol. 2020, pp. 1904172:1–1904172:9, Sep.

9. A. Kumar and A. Jaiswal, "Systematic literature review of sentiment analysis on Twitter using soft computing techniques," Concurrency Comput., Pract. Exp., vol. 32, no. 1, Jan. 2020, Art. no. e5107.

10. F. Z. Kermani, F. Sadeghi, and E. Eslami, "Solving the Twitter sentiment analysis problem based on a machine learning-based approach," Evol. Intell., vol. 13, no. 3, pp. 381–398, 2020

11. Pontiki, Maria, et al. "SemEval-2016 task 5: Aspect based sentiment analysis." ProWorkshop on Semantic Evaluation (SemEval-2016). Association for Computational Linguistics, 2016.

12. Ahmed, Khaled, Neamat El Tazi, and Ahmad Hany Hossny. "Sentiment Analysis over Social Networks: An Overview." Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on. IEEE, 2015.

13. Fang, Xing, and Justin Zhan. "Sentiment analysis using product review data." Journal of Big Data 2.1 (2015).

14. Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12), 2009.

15. Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10), 1320-1326.

16. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment analysis of Twitter data. Proceedings of the Workshop on Languages in Social Media, 30-38.

17. Smailović, J., Grčar, M., Lavrač, N., & Žnidaršič, M. (2013). Sentiment analysis on Twitter. Journal of Information and Organizational Sciences, 37(1), 25-40.

18. Wang, G., & Xie, Y. (2016). A comparative study of sentiment analysis methods on Twitter. Journal of Information Science, 42(5), 699-714.

19. Shrivastava, S., & Singh, S. (2018). Hybrid model for Twitter sentiment analysis. Proceedings of the International Conference on Computing, Communication and Automation (ICCCA), 1-6.

20. Chen, Y., & Skiena, S. (2014). Building sentiment classifiers for all languages. Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'14), 4236-4242.

21. Agarwal, D., & Mittal, N. (2019). Twitter sentiment analysis: A review of techniques and applications. IEEE Access, 7, 107467-107479.

22. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

23. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment in Twitter events. Journal of the American Society for Information Science and Technology, 61(12), 2521-2530.

24. Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010). Detecting sentiment change in Twitter streaming data. Proceedings of the 4th International Workshop on Knowledge Discovery from Sensor Data, 1-7.

25. Mohammad, S. M., & Turney, P. D. (2010). Emotions evoked by common words and phrases: Using mechanical Turk to create an emotion lexicon. Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, 26-34.

26. Wang, B., & Ittycheriah, A. (2013). Twitter sentiment analysis with recursive neural networks. Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 7083-7087.

27. Aggarwal, C. C., & Zhai, C. (2012). Mining text data. Springer Science & Business Media.

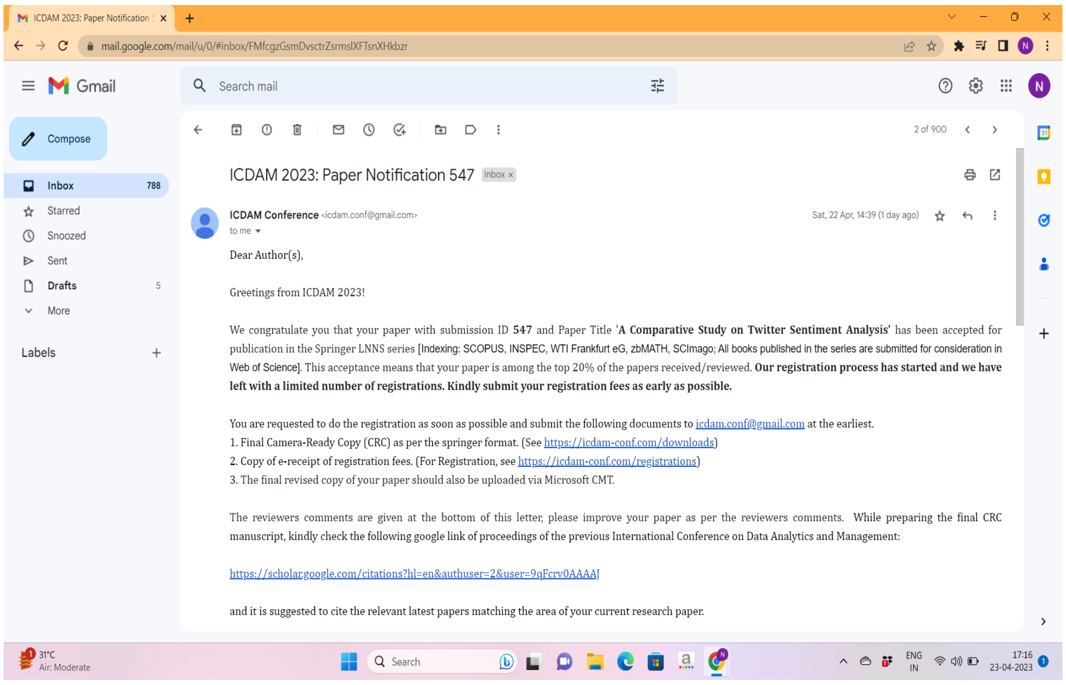
28. Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, 5(4), 1093-1113.

ANNEXURE I

Paper for the said project has been accepted in 4th International Conference on Data Analytics & Management (ICDAM-2023) ICDAM-2023 Theme: Data Analytics with Computer Networks Organized By: London Metropolitan University, London, UK

Paper Title: "A Comparative Study on Twitter Sentiment Analysis"

Authors: Nitesh Mishra, Mousam Panthi, Aditi Patial, Dr. Jabir Ali



ICDAM 2023: Paper Notification 547

ICDAM Conference <icdam.conf@gmail.com>
to me
Sat, 22 Apr, 14:39 (1 day ago)

Dear Author(s),

Greetings from ICDAM 2023!

We congratulate you that your paper with submission ID **547** and Paper Title '**A Comparative Study on Twitter Sentiment Analysis**' has been accepted for publication in the Springer LNNS series [Indexing: SCOPUS, INSPEC, WTI Frankfurt eG, zbMATH, SCImago; All books published in the series are submitted for consideration in Web of Science]. This acceptance means that your paper is among the top 20% of the papers received/reviewed. **Our registration process has started and we have left with a limited number of registrations. Kindly submit your registration fees as early as possible.**

You are requested to do the registration as soon as possible and submit the following documents to icdam.conf@gmail.com at the earliest.
1. Final Camera-Ready Copy (CRC) as per the springer format. (See https://icdam-conf.com/downloads)
2. Copy of e-receipt of registration fees. (For Registration, see https://icdam-conf.com/registrations)
3. The final revised copy of your paper should also be uploaded via Microsoft CMT.

The reviewers comments are given at the bottom of this letter, please improve your paper as per the reviewers comments. While preparing the final CRC manuscript, kindly check the following google link of proceedings of the previous International Conference on Data Analytics and Management:

https://scholar.google.com/citations?hl=en&authuser=2&user=9qFcrv0AAAAJ

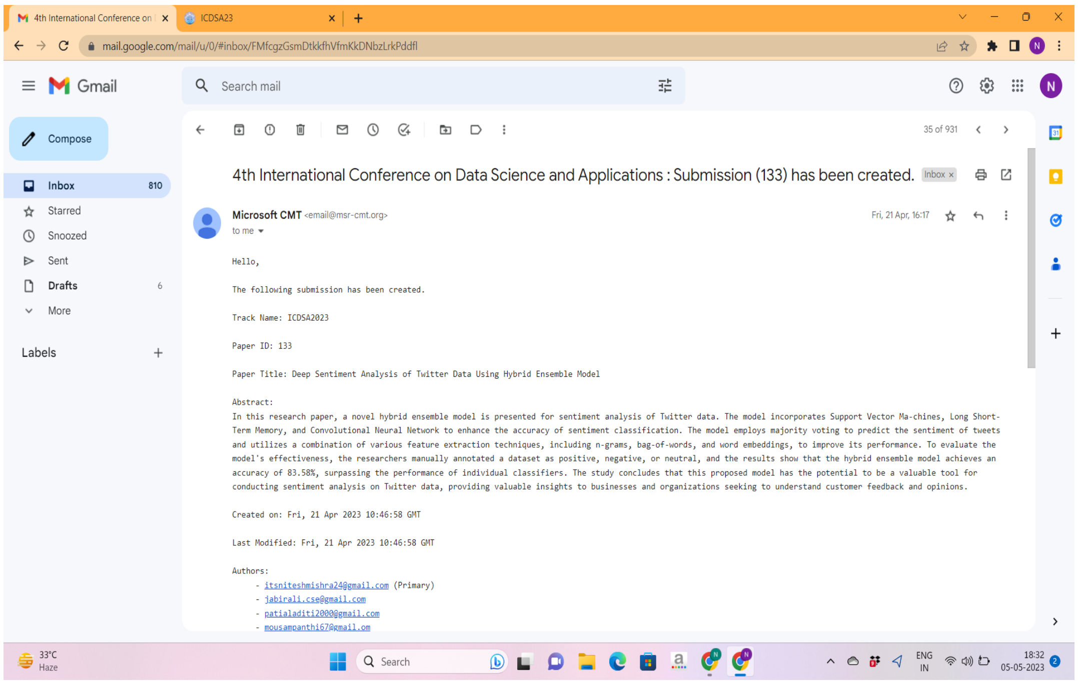
and it is suggested to cite the relevant latest papers matching the area of your current research paper.

ANNEXURE 2

Research Paper for the said project has been communicated at “4th International Conference on Data Science and Applications (ICDSA 2023) Organised by Malaviya National Institute of Technology Jaipur, India”.

Paper Title: Deep Sentiment Analysis of Twitter Data Using Hybrid Ensemble Model

Authors: Nitesh Mishra, Mousam Panthi, Aditi Patial, Dr. Jabir Ali



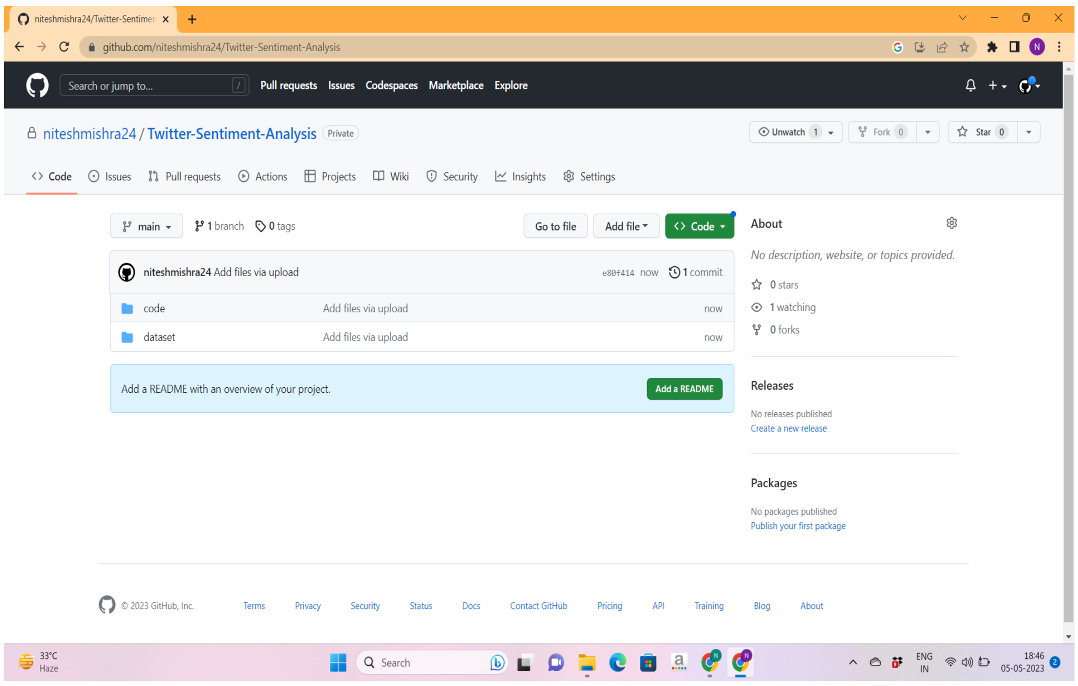
4th International Conference on Data Science and Applications : Submission (133) has been created.

```
Hello,

The following submission has been created.

Track Name: ICDSA2023

Paper ID: 133

Paper Title: Deep Sentiment Analysis of Twitter Data Using Hybrid Ensemble Model

Abstract:
In this research paper, a novel hybrid ensemble model is presented for sentiment analysis of Twitter data. The model incorporates Support Vector Ma-chines, Long Short-
Term Memory, and Convolutional Neural Network to enhance the accuracy of sentiment classification. The model employs majority voting to predict the sentiment of tweets
and utilizes a combination of various feature extraction techniques, including n-grams, bag-of-words, and word embeddings, to improve its performance. To evaluate the
model's effectiveness, the researchers manually annotated a dataset as positive, negative, or neutral, and the results show that the hybrid ensemble model achieves an
accuracy of 83.58%, surpassing the performance of individual classifiers. The study concludes that this proposed model has the potential to be a valuable tool for
conducting sentiment analysis on Twitter data, providing valuable insights to businesses and organizations seeking to understand customer feedback and opinions.

Created on: Fri, 21 Apr 2023 10:46:58 GMT

Last Modified: Fri, 21 Apr 2023 10:46:58 GMT

Authors:
      - itsniteshmishra24@gmail.com (Primary)
      - jabirali.cse@gmail.com
      - patialaditi2000@gmail.com
      - mousampanthi67@gmail.om
```

ANNEXURE 3

Github Repository : Twitter-Sentiment-Analysis

Github Repository Link : https://github.com/niteshmishra24/Twitter-Sentiment-Analysis



github.com/niteshmishra24/Twitter-Sentiment-Analysis

niteshmishra24 / Twitter-Sentiment-Analysis Private

main 1 branch 0 tags

niteshmishra24 Add files via upload e80f414 now 1 commit

code	Add files via upload	now
dataset	Add files via upload	now

Add a README with an overview of your project.

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

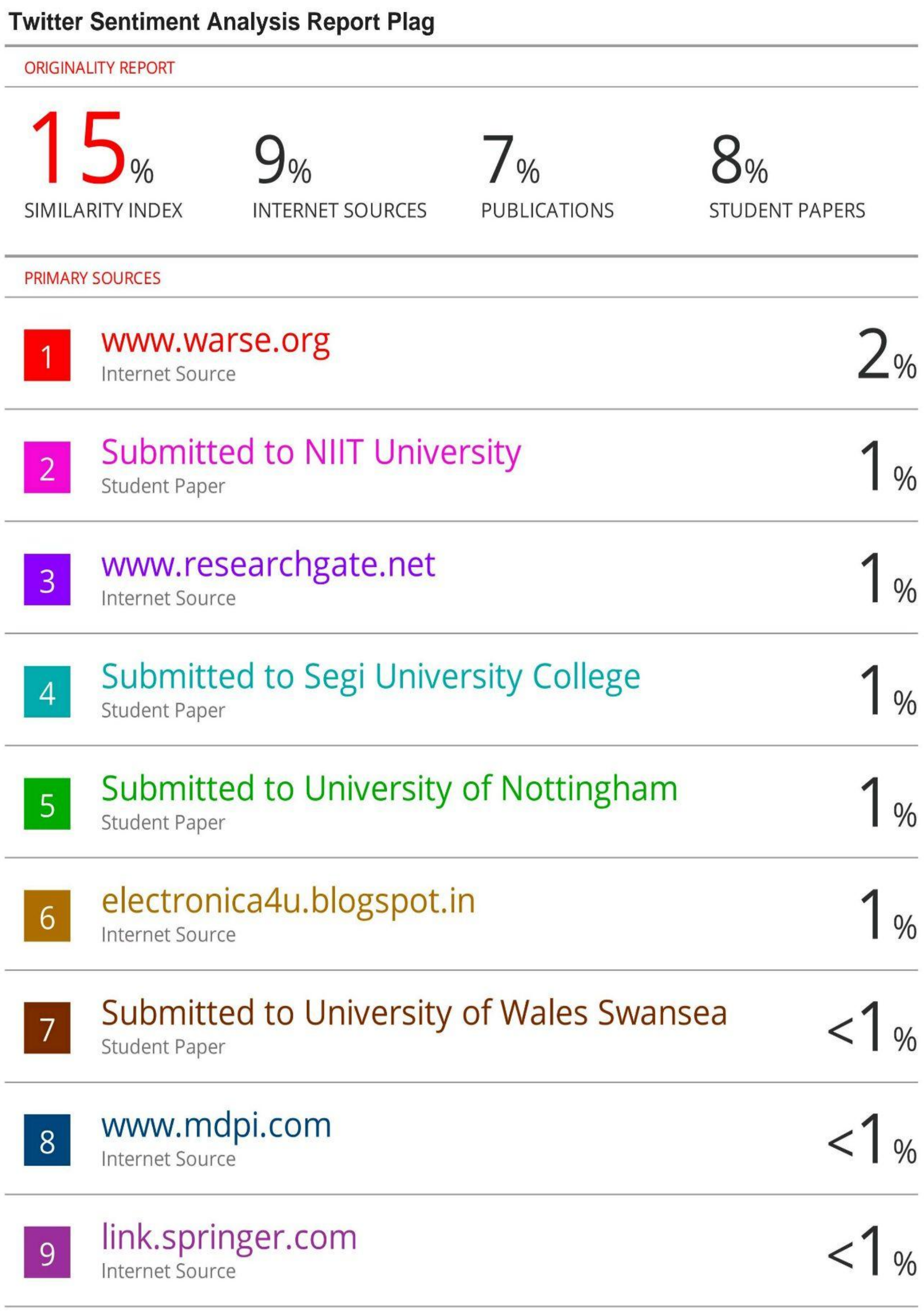
Packages

No packages published

Publish your first package

© 2023 GitHub, Inc.

ANNEXURE 4



Twitter Sentiment Analysis Report Plag

ORIGINALITY REPORT

15%	9%	7%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.warse.org Internet Source	2%
2	Submitted to NIIT University Student Paper	1%
3	www.researchgate.net Internet Source	1%
4	Submitted to Segi University College Student Paper	1%
5	Submitted to University of Nottingham Student Paper	1%
6	electronica4u.blogspot.in Internet Source	1%
7	Submitted to University of Wales Swansea Student Paper	<1%
8	www.mdpi.com Internet Source	<1%
9	link.springer.com Internet Source	<1%

10	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
11	Jarkko Hyysalo, Sandun Dasanayake, Jari Hannu, Christian Schuss, Mikko Rajanen, Teemu Leppänen, David Doermann, Jaakko Sauvola. "Smart Mask – Wearable IoT Solution for Improved Protection and Personal Health", Internet of Things, 2022 Publication	<1 %
12	"Intelligent Communication, Control and Devices", Springer Science and Business Media LLC, 2018 Publication	<1 %
13	Submitted to Liverpool John Moores University Student Paper	<1 %
14	M. Jahangir Ikram. "Air pollution monitoring through an Internet-based network of volunteers", Environment and Urbanization, 04/01/2007 Publication	<1 %
15	Submitted to College of the North Atlantic-Qatar Student Paper	<1 %
16	P Prabhu, Umang ., Jyothi Jayakumar, Ch Phanindra Kumar, . .. "Intelligent Wearable	<1 %

	Device for Coal Miners", International Journal of Engineering & Technology, 2018 Publication	
17	Submitted to Walters State Community College Student Paper	<1 %
18	journals.plos.org Internet Source	<1 %
19	Submitted to University of Ulster Student Paper	<1 %
20	Submitted to University of South Africa Student Paper	<1 %
21	Fabian Medina, Luis Patarroyo, Bryann Gutierrez, Jorge Enrique Espindola, Eduardo Avendano Fernandez. "Wearable Prototype Module for the Measurement of Explosive Gases in Underground Mining", 2021 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT), 2021 Publication	<1 %
22	Submitted to University of Liverpool Student Paper	<1 %
23	Submitted to University of South Australia Student Paper	<1 %
24	dlibrary.univ-boumerdes.dz:8080 Internet Source	