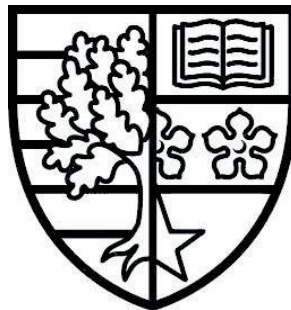


SENTIMENT ANALYSIS OF COVID-19 TWEETS

Author: Muhammad Khan
BSc (Hons) Computer Science
Deliverable 1: Final Year Dissertation

Supervised by Prof. Smitha Kumar



HERIOT-WATT UNIVERSITY

School of Mathematical and Computer Sciences
Department of Computer Science

November 2021

Student Declaration of Authorship

Course code and name:	F20PA Research Methods and Requirements Engineering
Type of assessment:	Individual
Coursework Title:	Dissertation
Student Name:	Muhammad Shayaan Aslam Khan
Student ID Number:	H00304770

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.
- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the [University's website](#), and that I am aware of the penalties that I will face should I not adhere to the University Regulations.
- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](#)

Student Signature: *Muhammad Khan*

Date: 21/04/2022

Abstract

The coronavirus SARS-CoV-2 was responsible for the COVID-19 pandemic. Its beginnings can be traced back to December of 2019 in Wuhan, China. Because of its rapid growth and dissemination, it has had an impact on people's daily lives all over the world, resulting in lockdowns, social distancing, and other restrictions and regulations in citizens' daily lives.

The public has faced new problems and experiences as a result of the efforts taken to control the spread of the virus over the last two years. People across the world have openly expressed their opinions and perspectives on social media sites like Facebook, Twitter, and Instagram in response to these new experiences and situations.

This enabled data analysts to obtain vast volumes of data for analyzing people's opinions on matters such as lockdowns and vaccines, which aided in the fight against the coronavirus pandemic.

The goal of this project is to analyze tweets posted by people about the pandemic and allow readers to understand how public opinion has altered throughout the pandemic. Algorithms such as LSTM and RNN will be used to do this. Finally, the implemented model's performance will be compared to other baseline models.

Keywords: *COVID-19, Sentiment analysis, Coronavirus, Twitter*

Table of contents

Declaration of Authorship	2
Abstract	3
Table of contents	4
1 Chapter 1: Introduction	8
1.1 Aim	8
1.2 Objectives	8
1.3 Manuscript Organization	9
2 Chapter 2: Literature Review	10
2.1 The role of Twitter in the COVID-19 pandemic	10
2.2 Introduction to sentiment analysis and social media	10
2.3 How is sentiment analysis helpful during events such as COVID-19 pandemic	11
2.4 Available Datasets	12
2.5 Data processing	13
2.5.1 Data pre-processing	13
2.5.2 Feature Extraction	14
2.5.3 Sentiment analysis classifiers	15
2.6 Sentiment analysis techniques	15

2.7	Critical Review	18
2.7.1	Dataset collection	18
2.7.2	Data pre-processing	18
2.7.3	Evaluation Strategies and results	19
2.7.4	Conclusion	19
3	Chapter 3: Project Implementation	21
3.1	Hardware and environment	21
3.2	Dataset choices	21
3.3	Initial process	21
3.3.1	Data exploration	22
3.3.2	Data pre-processing	26
3.4	Feature extraction	28
3.5	Model building	31
3.5.1	LSTM	31
3.5.2	Simple RNN	32
3.5.3	Baseline models	33
4	Chapter 4: Critical review of implemented models	35
4.1	Performance evaluation	35
4.1.1	VADER Sentiment Intensity Analyzer	36

4.1.2	Deep Learning models	37
4.1.3	Baseline models	39
4.2	Limitations	43
5	Chapter 6: Conclusion and future work	44
5.1	Requirements Validation	44
5.2	Conclusion	44
5.3	Project source code	45
5.4	Challenges	45
5.5	Future work	45
6	Appendix A: Requirements Analysis	47
A.1	Functional requirements	47
A.2	Non-Functional requirements	48
7	Appendix B: Project management and development methodology	49
B.1	Development methodology	49
B.2	Project management	50
B.2.1	Risk management	50
B.2.2	Project implementation and analysis	52
B.3	Project plan	54
8	Appendix C: Discussion of professional, legal, ethical, and social issues	55

C.1	Professional and Legal issues	55
C.2	Ethical and Social Issues	55
9	Chapter 6: References	56

Chapter 1

Introduction

Sentiment analysis has enabled the automatic examination of information available on social media sites to identify the polarity of people's thoughts. Sentiment analysis technology has progressed over the years to analyze various characteristics such as a person's views or emotions toward a specific issue. [1].

Sentiment analysis has enabled the automatic examination of information available on social media sites to identify the polarity of people's thoughts. Sentiment analysis technology has progressed over the years to analyze various characteristics such as a person's views or emotions toward a specific issue.

1.1 Aim

The aim of this project is to analyze how the sentiments of the people has changed and shifted about the pandemic as well as performing an analytical comparison of the performance of various ML approaches. This will allow us insight about how the opinions of the people shifted over the course of the pandemic in reference to the important dates as well as analyze different approaches to sentiment analysis.

1.2 Objectives

The objective of this project is primarily to assess the sentiment that the tweets portray and to provide an analysis for the readers of this project so that they may understand how the public opinions have shifted throughout the pandemic.

The primary objectives of this project are:

- Pre-processing the dataset using python libraries so that machine learning models can be applied to them.
- Application of different deep learning models to analyze the sentiments of the tweets
- Comparative analysis of the different ML models.

1.3 Manuscript Organization

The project presented has been organized to maintain a flowing structure where the Literature Review covers previously implemented machine learning models relevant in this area of study as well as background regarding sentiment analysis of twitter data. Following this, we have the Requirements Analysis which provides user requirements and MoSCoW analysis of the requirements. Next, we have the development methodology and the Project Management which includes the development methodology, analysis of the risks and appropriate mitigation plans as well as a timetable for the deliverables and deadlines. Following that we have the discussion of professional, legal, ethical, and social issues and finally all the references used are mentioned.

Chapter 2

Literature review

In this chapter we shall look over the different techniques and machine learning models that allow us to perform sentiment analysis over Twitter data.

2.1 The role of Twitter in the COVID-19 pandemic

Twitter is one of the most widely used social media platforms nowadays. As a result of the pandemic, social media usage has increased as more individuals stay indoors and participate in virtual media. Tweets on Twitter have been impacted by the pandemic because there was a spike in people sharing their personal thoughts and stories regarding COVID-19, such as the general population, doctors, and famous figures around the world. During the crisis, Twitter enabled the rapid dissemination of information to the public. However, the spread of correct and incorrect information has also been a source of concern. [2].

Twitter data has been used to study and analyze behavior, and because people have distributed a great amount of information and thought on the platform, researchers have been able to obtain reports on people's behaviors and perspectives. As a result, Twitter has played a significant role in allowing people to talk about their experiences and communicate, as well as allowing researchers and organizations to understand the sentiment of the people affected by the pandemic and form policies and decisions, accordingly, emphasizing how important Twitter is during COVID-19.

2.2 Introduction to Sentiment analysis and social media

Sentiment analysis is the methodical recognition, extraction, evaluation, and examination of emotional states and subjective information using natural language processing, text mining, computational linguistics, and biometry. This is often referred to as opinion mining [3]. Several methods have been used in recent studies for sentiment analysis in the field related to the topic of this study to extract the emotions behind tweets.

In today's environment, social media platforms are a tremendous tool. It has developed throughout time, from emails and chat rooms in its early days to WhatsApp, Facebook, Instagram, Twitter, and so on. People are constantly sharing their views, knowledge, and other material such as photographs due to the variety of platforms available and the easy accessibility of social networking sites. Because everyone utilizes social media, it is a key driving factor in defining and shaping public opinion. It is one of the most used means of exchanging information. Because of its popularity, social media can be seen as a mainstream outlet for big data.

When people publish a post, they express a wide range of emotions such as rage, happiness, grief, joy, fear, and so on. This presentation of human emotion allows business leaders, legislators, and healthcare organizations to better comprehend public sentiment and make judgments or design policies accordingly [4].

Sentiment analysis has been used in a wide range of economic areas, including tourism, education, health, and safety, and so on. This has enabled corporations to comprehend and analyze public opinion and change accordingly, implying that sentiment analysis is a powerful and vital tool for understanding trends and occurrences around the world.

2.3 How is sentiment analysis helpful during events such as the COVID-19 pandemic?

The pandemic has had a negative influence on the world, resulting in increased unemployment, mental health difficulties, and the deaths of thousands of individuals. As previously noted, people voiced their ideas on Twitter, which led to sentiment analysis research employing NLP and ML algorithms. Sentiment analysis studies gave information and results about the pandemic.

A study conducted by [5] employed sentiment analysis on tweets linked to the covid-19 pandemic and discovered that the World Health Organization's tweets failed to provide individuals with guidance so that they might better deal with the corona virus pandemic. Another study [6] conducted sentiment analysis on covid-related tweets in India and found that people responded positively and were supportive of the government's decision to implement the initial nationwide shutdown.

According to such research, sentiment analysis plays a vital role in helping corporations and governments comprehend the public in an event like the present pandemic, and it allows for future planning and decisions.

2.4 Available datasets

1) COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes

The dataset comprises of over 198 million tweets from over 25 million unique users from all around the world. The tweets were collected from the 28th of January 2020 till the 1st of September 2021 using the keywords: “corona”, “Wuhan”, “nCov”, and “covid”. Each tweet was labelled with 17 semantic attributes which include the tweets relevance, quantitative emotional attributes, and qualitative sentiment categories. [7].

2)CORONAVIRUS (COVID-19) GEO-TAGGED TWEETS DATASET

This dataset was collected using over 90 keywords relating to the covid-19 pandemic. It contains over 426 thousand tweets and growing. [8]

3)COVID-19 Twitter Dataset

This is a massive dataset with around 237 million tweets from March 2020 till the July of 2020 that contain the word “COVID”. It was collected by sending hourly requests to the Twitter search API. [9]

4)COVID19 Tweets

This dataset was compiled by Gabriel Preda collecting tweets with the hashtag #covid19. The tweets were collected daily from July 2020 till the end of August 2020 with over 178 thousand tweets [10].

2.5 Data processing

The processing of data is a key stage in sentiment analysis. The goal of data processing is to fine-tune the raw data provided such that only the information needed for the investigation is presented. It also aids in improving the execution time of the algorithms as well as the quality of the results obtained.

2.5.1 Data Pre-processing

Data pre-processing is a vital step that allows the quality of data to be enhanced which results in the extraction of meaningful insights of the data [11]. Discussed below are the pre-processing techniques that have been commonly used in the sentiment analysis of twitter data:

- **Removal of irrelevant text data from Tweets:** The initial step in the dataset's pre-processing stage is to remove texts that are not useful in determining the sentiment of the tweet. The tweets were cleaned in the study by [5] to eliminate redundant and irrelevant symbols and content such as @, RT, #, URLs, numeric values, and punctuation marks.
- **Capital Conversion:** Words with capital/uppercase letters are rewritten in lower case letters.

As the dictionary only contains the lower-case version of any term, this helps to increase context similarities while also lowering the number of words that must be held. [12]

- **Stop word removal:** Some studies, such as [5] and [13], remove stop words because they were deemed unneeded. Stop words are terms that are extensively used in the English language, such as "a," "are," "the," and "is." Stop words have limited meaning in text analysis and categorization and add little value to the text to be analyzed. This strategy aids in reducing dataset size as well as model training time, resulting in improved performance. [14].
- **Tokenization:** It is the process of breaking down words into 'tokens,' which are typically words, characters, or n-grams. Unigram tokenization is the most often used type of tokenization. The text is divided into individual words, which form the tokens. Whitespaces are used to separate words [15].
- **Word normalization:** Stemming and lemmatization are forms of word normalization. The practice of reducing all words with identical 'stems' to a common form is known as stemming. The basic form of any given word is characterized as a stem. Words like "Fishing," "Fisher," and "Fished" will all be reduced to the term "Fish" because of this process [16]. Lemmatization is the process of grouping together similar words expressed in

different formats [17]. As an example of lemmatization, the words "Begging," "Beggar," and "Beginning" will be substituted with "Beg," "Beg," and "Begin" [16].

- Parts of Speech: POS is a method of categorizing components of speech such as nouns, verbs, and adjectives. POS supports automatic part-of-speech annotation for each word in your document. [18].

2.5.2 Feature Extraction

Feature extraction is the process for extracting features that aid in the sentiment analysis of text. The quality of the features selected for model training overall determine the quality of the model developed itself. Following are some methods of feature extraction that researchers have commonly used:

- Bag of words: It is a representation that defines how words appear in a document. Only the quantity of words is recorded, while the grammar and word order are ignored. This is referred to as a "bag" of words since it discards all information about the order and structure of words in the input textual data. [19].
- TF-IDF: It is calculated by multiplying TF and IDF. Essentially, it is the metric that calculates the relevance of a word in relation to a given text. The number of times a word appears in the text is represented by term frequency (TF), while the rarity of a word in the text is represented by Inverse Document Frequency (IDF). When IDF is calculated, a low value for a term indicates that the word is more prevalent. [20]. Mathematically, this is represented as follows:

$$tf\ idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

where

$$tf(t, d) = \log(1 + freq(t, d))$$

$$idf(t, D) = \log(N / count(d \in D: t \in d))$$

- Word embedding: It is accomplished using techniques that map words or sentences from the vocabulary to real number vectors. This approach typically necessitates the mathematical embedding of a huge corpus with many dimensions per word in a vector space of a lower dimension. This is important for deep learning models or machine learning problems involving text classification. [20].

2.5.3 Sentiment Analysis classifiers

- **LSTM:** It is a type of recurrent neural network (RNN) that can learn long-term dependencies, particularly in sequence prediction issues. It features a feedback connection, which means it can process the complete data sequence while ignoring individual data points. [21].
- **Decision Tree:** They are a type of supervised learning known as non-parametric supervised learning, and they are used for classification and regression. Its goal is to anticipate the value of a variable by learning decision rules from data features used as inputs [22].
- **Logistic regression:** It is a different kind of supervised learning algorithm. It forecasts whether a variable will be successful or unsuccessful. It comes in a variety of forms, including Binary, which predicts only two possible outcomes (1 and 0), Multinomial, which gives the result of the dependent variable as one of three or more unordered types, and Ordinal, which is similar to Multinomial but gives the result of the dependent variable as one of three or more ordered types. [23].
- **BERT:** It is a Google NLP model that employs Transformer, an attention mechanism that learns contextual relationships between words in a text. It employs Transformer's encoder technology to read the complete sequence of words at once. This enables the model to learn a word's context based on its surroundings. To function, BERT employs two training strategies: Masked ML and Next sentence prediction. [24].

2.6 Sentiment analysis techniques

Numerous studies have used sentiment analysis techniques on Twitter data to examine a wide range of issues. Most of them employed approaches that were quite similar, such as utilizing the Twitter API to collect tweets over a specific time period. Some were dissatisfied, though, because the Twitter API has severe constraints on the amount of data that can be taken. These restrictions include only being able to access tweets from a maximum of 7 days ago and only being able to scrape 18,000 tweets in a 15-minute span [25]. This is frequently difficult since researchers may demand a bigger quantity of tweets or may want to access tweets from a certain time range that surpasses the 7-day history restriction. Other tools, like as Twint, have no such constraints, and are frequently used. After collecting their datasets, the researchers use multiple algorithms to analyze the sentiment of the Twitter data. The methodologies used to model various sentiment analysis programs from various research articles are discussed below.

To begin, we will look at the work of Chakraborty et al. [5], who suggested a Fuzzy Rule-based model. Fuzzy logic is a type of multi-valued logic in which the truth value falls between 0 and 1, and the values may even contain 0 and 1. To analyze the sentiment of the text, they employed a deep learning approach. Between December 2019 and May 2020, two distinct datasets were collected. The model is distinct in that it characterizes fuzzy inputs using Gaussian membership functions (LOW, MEDIUM, HIGH) and fuzzy outputs using the functions NEGATIVE, NEUTRAL, and POSITIVE.

To identify the output, the Mamdani Fuzzy inference mechanism had been used to characterize the model, which was then defuzzied using the Centroid Defuzzification Method. They evaluated their model's performance to that of the Triangular membership function-based fuzzy rule system and concluded that their proposed model surpassed the other, achieving an accuracy of 79%.

They were able to draw significant inferences from their model, such as the need for governments to strive to ensure the quality of information distributed on social media. They also highlight how the people reacted positively to the sickness, although the imposition of lockdowns was met with skepticism and animosity. The study also examined tweets from the World Health Organization and concluded that they, too, failed to provide useful advice in dealing with the disease.

Deep learning models, such as LSTM, BD-LSTM, and BERT, are used in the works of [20]. Their goal was to conduct an experimental study comparing multi-label classification using various models. They used the Senwave COVID-19 dataset, which is a hand-labelled sentiment dataset encompassing 11 different sentiments and was labelled by 50 professionals. They used GLoVe embedding vectorization to embed each word into a vector with 300 dimensions. Their methodology, which is based on BCE (Binary Cross-entropy) loss and Hamming loss, was developed for multi-label classification.

BCE is a mixture of SoftMax activation and cross-entropy loss [26], whereas Hamming loss is a calculation of the loss generated in the bit string of class labels using XOR between the actual and predicted labels, and the average of all instances is obtained after this procedure [27].

The hyper-parameters of the LSTM model were determined depending on how the model performed in trial tests. For the LSTM and Bidirectional LSTM models, they utilized a dropout regularization probability of 0.65. They include 300 input features, two layers with 128 and 64 hidden units, and an output layer with 11 units for sentiment classification. They used the default hyper-parameters for the base model in the BERT model and only changed the learning rate. They made a critical addition by implementing a dropout layer at the end of the BERT architecture and a linear activation layer with 11 outputs, where the outputs correspond to the 11 sentiments labelled in the dataset. After configuring the models, they fed the training data (Senwave COVID-19 dataset) into the models. After training and testing, they discovered that BERT performed best since it

seemed to capture more expressed thoughts than the LSTM model.

Now we will have a look at the study reported by [13]. They also employed the BERT Deep Learning model. To create their proposed model, they used a combination of manual and automatic procedures. Their pre-processing included manually collecting tweets using a Twitter scraper and tweepy APIs. In the final phases of pre-processing, they had obtained two datasets, one having tweets from all around the world and the other containing solely tweets from India. They used the Vader Sentiment Analyzer's features to generate a polarity score for the entire text. The hugging face with pytorch library for emotion classification was used to construct the mask and encoder representation for the model. They translated the training set into their respective torch tensors and defined batch size to construct the tensors and iterators needed to fine-tune the model. To assess the model's performance, they used metrics such as average likes over period, average retweets over period, intensity analysis, polarity, and subjectivity, and wordcloud. In the study they conducted, the accuracy of their model was 93.89 percent.

Finally, we describe the work of [28], in which they used various machine learning classifiers to find the best classifier among those. The best classifier was chosen based on accuracy and F score. They employed logistic regression, the Nave Bayes classifier, the decision tree classifier, and the random forest classifier as well/ Their initial requirement was the establishment of a SparkContext that enables live data stream processing. They created their model with a six-stage pipeline that included tokenization, N-gram creation, Count vectorizer, IDF, vector assembler, and logistic regression.

They used real-time tweets with the tag COVID-19 that were extracted using the Tweepy API and then cleaned, removing URLs, mentions, punctuation, and so on, so that they could be put into the pipeline. The one-of-a-kind feature they introduced was that they did not simply analyze text, but also texts from individual input movies and photographs. They took frames from videos at regular intervals and retrieved text from those frames using the Tesseract tool. According to their findings from the study, the logistic regression model produced the greatest outcomes in terms of accuracy and F score.

2.7 Critical review

In this section, the various approaches in sentiment analysis such as pre-processing techniques and methodology are summarized.

2.7.1 Dataset Collection

Most of the researchers manually collected their tweets using web-scraping tools and the Tweepy API for specific keywords like #COVID19. Though tweets were acquired using Tweepy in the study [28], they worked with real-time Twitter data that was applied using PySpark. [20] used a pre-existing dataset called the Senwave Covid-19 dataset, which is a hand-labelled dataset annotated by 50 experts.

Because of the difference in how a dataset is obtained, processing of the dataset may be necessary to assign sentiment labels to words and phrases. This is especially common in studies when the tweets are gathered by the researchers themselves, such as [28]. The other method is to employ a pre-labelled dataset, as was done in the instance of [20].

2.7.2 Data Pre-Processing

Because of the nature of how social media users express themselves on Twitter, methods like stop word removal, URL removal, mentions, punctuation, and hashtag removal are common in twitter sentiment analysis. Tokenization is also extensively utilized since it streamlines and simplifies the process of feeding the model's input. They used POS tagging in the study [5] and then used the SentiWordNet lexical database to provide sentiment scores to words. This type of data pre-processing has the advantage of making it relatively simple to assign sentiment scores to text from an unlabeled dataset. Other writers who gathered tweets independently used similar methods. For example, [13] utilized the Vader sentiment analyzer to generate the polarity score, which works similarly to the methods developed by [5], but it requires less pre-processing and so takes less time.

[20] uses a pre-labelled dataset to skip the stages of assigning sentiment scores. The benefit of this technique is that the models work with valid and verified datasets that typically contain only the relevant information, minimizing the amount of pre-processing required and, as a result, the overall run time of the model is also reduced.

2.7.3 Evaluation Strategies and Results

Researchers frequently compare their models to baseline models in order to assess and analyze their performance. This was the case in [5,] when they compared their Fuzzy rule-based model against others including Linear SVC, AdaBoostClassifier, Random Forest Classifier, Logistic Regression, and Multinomial Nave Bayes. They compared their Gaussian membership model to the triangle membership model, and their proposed model outperformed the other. For such comparisons, metrics such as Accuracy, Precision, and Recall are employed.

On a pre-labelled dataset, [20] compared the performances of LSTM, BD-LSTM, and BERT. They used a mix of F1 macro and F1 micro scores to assess the results of various models. Based on the parameters indicated earlier, their research concluded that BERT outperformed both other models.

The study [28] followed the same procedure with a small difference: instead of constructing and comparing a model, they compared Nave Bayes, Decision Trees, Random Forest, and logistic regression to each other for a critical examination. They analyzed their results based on accuracy, precision, recall, and F score, and then used the model with the greatest performance, which turned out to be Logistic Regression, for the remainder of their research.

Instead of comparing their model's performance to other baseline models, the study [13] simply evaluated the performance of BERT over a dataset and used 5 matrices, Average Likes over the period, Average Re-tweets over the period, Intensity Analysis, Polarity & Subjectivity, and Wordcloud, for the analysis of their model's performance, which achieved an accuracy of 93.89 percent.

2.7.4 Conclusion

Different studies have adopted different approaches to sentiment analysis. Their goals ranged from comparing different baseline models to each other, to comparing their models to baseline models, to just implementing their model to assess twitter sentiments. They used a variety of strategies for data pre-processing, feature extraction, and model construction.

We can conclude by stating that to build a successful model, we need to ensure that the context of each sentence has been understood and grasped appropriately in their vector representations such that they may be processed in an efficient and error-free manner by sentence modelling techniques and give us the identified sentiment of sentences that are present in the tweets.

The proposed system for this study would perform data pre-processing on the dataset. This will be

done using stopwords removal, removing duplicate tweets, performing word normalization and vectorization. After the pre-processing has been completed, the data would be split in an 8:2 ratio for the training and testing split. The ML algorithms would be trained on the training data and then tested on the unseen testing data to predict the sentiment of the tweets. From the performance results, the better deep learning model will be chosen for further experimentation and performance comparison with other baseline models. Comparison would be performed using matrices such as accuracy and F measure.

Chapter 3

Project implementation

3.1 Hardware and environment

The code was implemented in Python 3.9.0. Multiple libraries were used and implemented for various functions. Sklearn and Keras were used for machine learning; matplotlib, seaborn and wordcloud were used for the visualization of results and data; NLTK and vaderSentiment were used for majority of the preprocessing performed.

Sklearn is machine learning library for python offering various algorithms such as SVM, regression, naïve bayes, random forest, clustering etc.

Keras is an API developed on Tensorflow which focuses on deep learning with python. It offers machine learning models such as LSTM and RNN.

For the hardware upon which the program was to run had to be able to support the processing power that would be needed. The system had to be completely dedicated to running the program to allow execution at efficient speeds. A system built with a dedicated GPU of 6GB paired with a Quad core, 3.1GHz CPU and 16GB of RAM was used.

3.2 Dataset choices

Our dataset was acquired from Kaggle (<https://kaggle.com>). The website features a diverse collection of datasets to be explored by users. Users can also publish their own datasets, collaborate with other users, and use the web-based environment to build data science models.

From Kaggle, a dataset elected for this project was a set of tweets compiled by Gabriel Preda where tweets containing the hashtag #covid19 were chosen. The tweets feature over 178 thousand tweets at the time of development of the project and the tweets were dated from July 2020 till the end of August 2020 [10]

3.3 Initial processes

The implemented feature extraction and preprocessing methods were adapted and implemented from the works of other researchers with variations to suit the needs of the implemented model.

3.3.1 Data Exploration

In data science and machine learning, it is important for any researcher or developer to have a clear understanding and knowledge of the data that they will be using. Exploratory data analysis allows us to understand what the most common words are, where the majority of tweets originate from, what is the sentiment of those tweets, how they have shifted over a certain duration of time and more.



FIGURE 3.1

Figure 3.1 is a word cloud of all the tweets prior to any preprocessing. Upon observing, it is seen that the most common words include covid, new case, pandemic, children and more.

It is also important to attain the number of values that are missing, such that they can be accounted for and corrected before the data is fed into the model. This can be seen in figure 3.2

```
1 # Missing values in the form of percentages
2
3 for var in twt.columns:
4     if twt[var].isna().sum() > 0:
5         miss = np.round(twt[var].isna().sum() / twt.shape[0] * 100 , 3)
6         print(var, "has {} % of missing values".format(miss))

user_location has 20.53 % of missing values
user_description has 5.743 % of missing values
hashtags has 28.661 % of missing values
source has 0.043 % of missing values
```

FIGURE 3.2

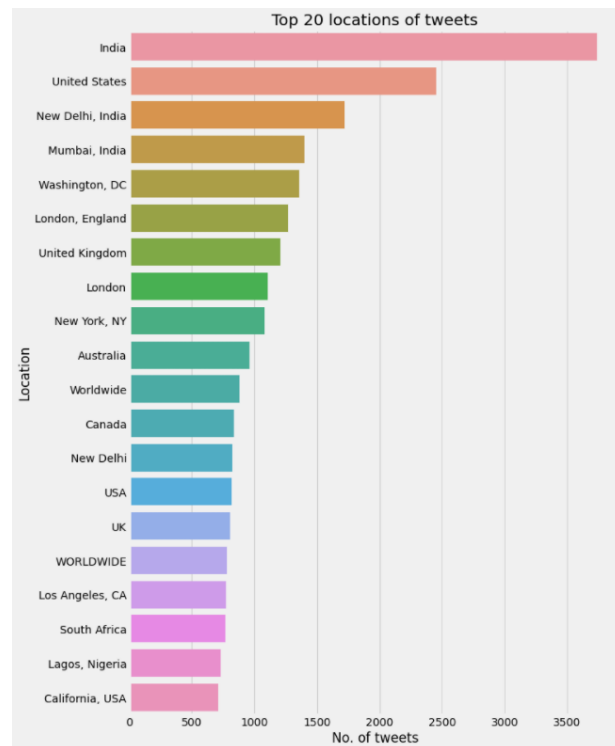


FIGURE 3.3

Figure 3.3 shows us what the top locations of the tweets are. The primary issue with location tagging of twitter is that it can be highly inconsistent as seen here. Users can also add their own custom locations which can skew the data under some cases. In the figure above, it can be seen that there are 2 instances of New Delhi which highlights redundancy. There is a similar case with the location Worldwide, where there are two instances. Another issue that persists is that worldwide is a location tag rather than a collection of all the tags, this implies that ‘worldwide’ is just another location like the USA, India, UK, etc. rather than ‘worldwide’ being a collection of all the tweets from all over the world.

After the implementation of sentiment scoring and assigning sentiments to each of the tweets, they have been categorized as positive, negative, and neutral. This allows for further data exploration for additional information.

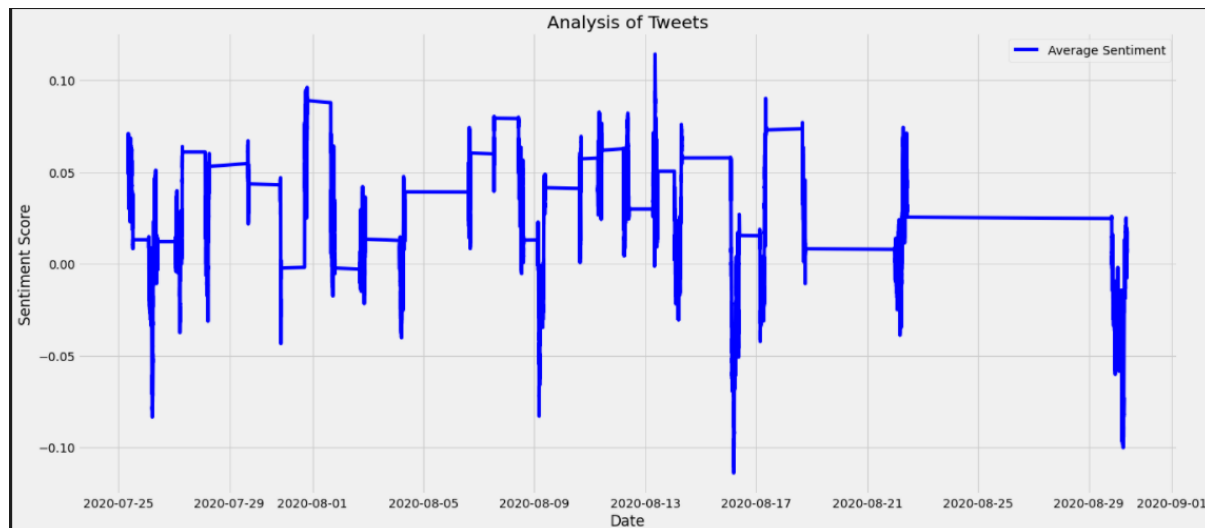


FIGURE 3.4

The figure above allows us to see how the sentiment of the people shifted over the course of July and August 2020. It can be seen that the average sentiment varied by a significant margin over on a fairly regular basis, and towards the end of August and the start of September, the sentiment has turned negative.

Once the sentiments have been acquired of the tweets and been categorized as positive, neutral, and negative, a graph to represent them and the number of tweets in each category can be plotted.

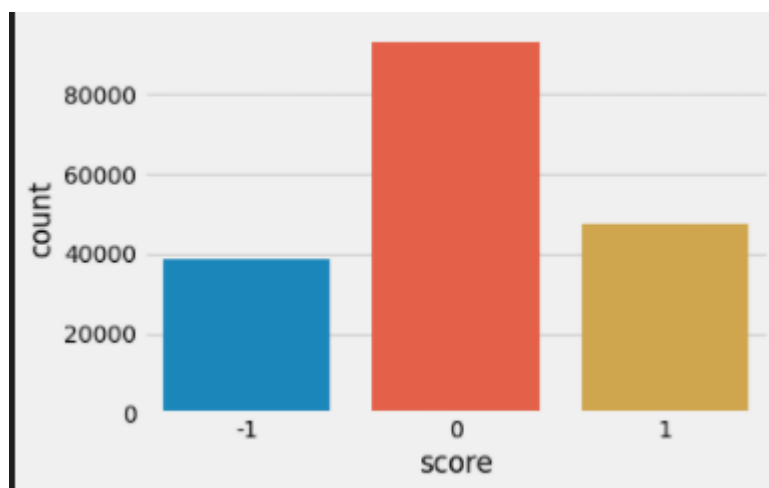


FIGURE 3.5

Looking at figure 3.5, it can be seen that the number of negative and number of positive tweets is nearly the same and quite balanced. However, it can be seen the number of tweets that have a neutral sentiment is more than the combined number of negative and positive tweets.

It is now also possible to formulate wordclouds for the positive, neutral, and negative sentiment tweets.

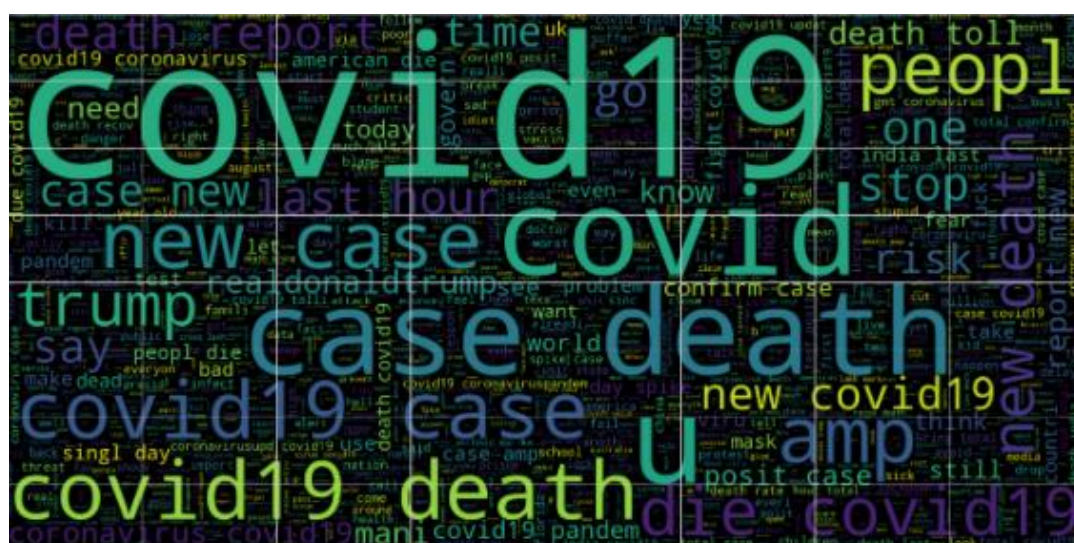
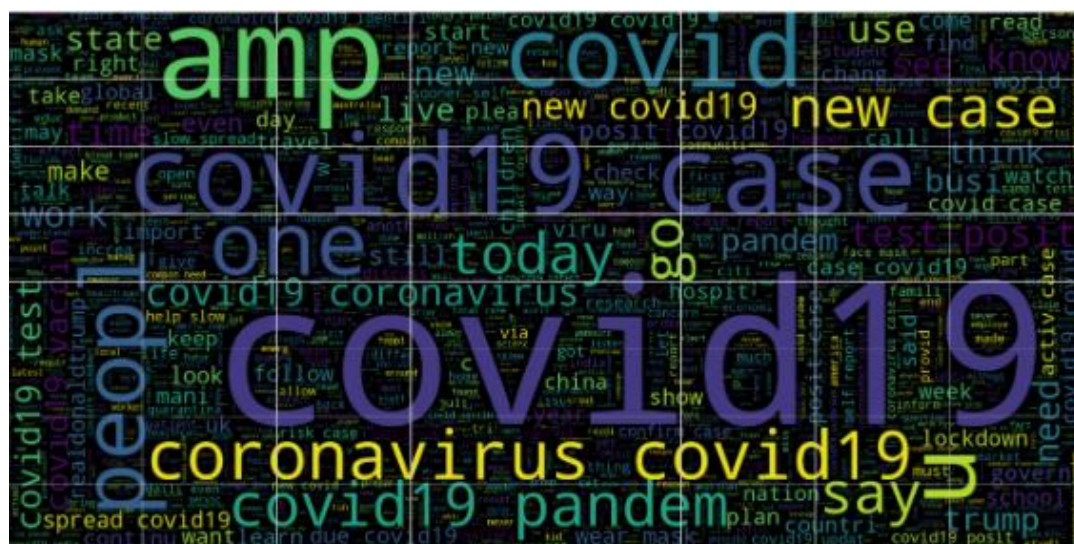
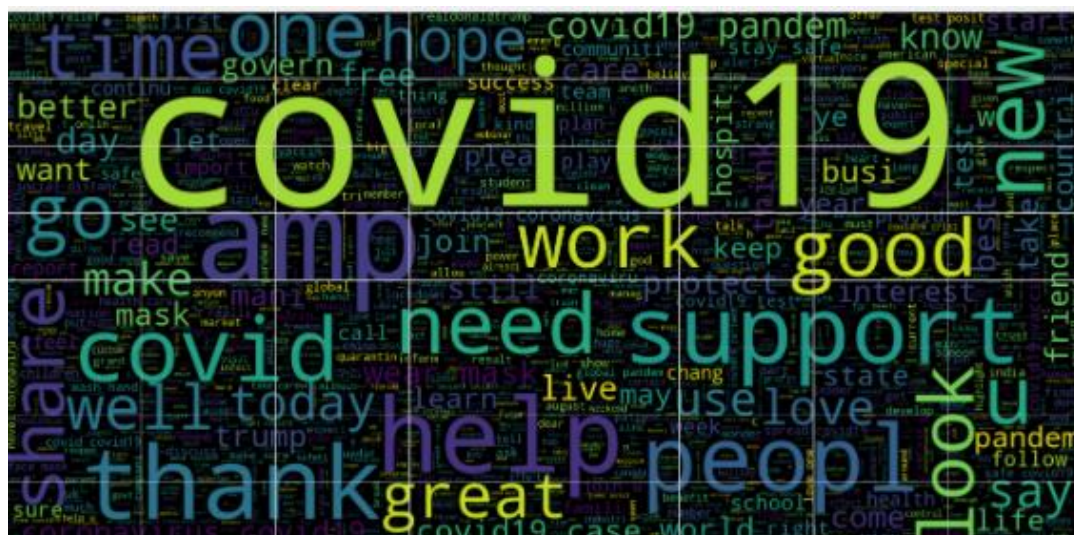


Figure 3.6 displays the most common words from tweets that have been classified as positive in their sentiment, figure 3.7 shows tweets that are neutral in sentiment and figure 3.8 shows the common words from tweets that are negative in sentiment.

It can be seen that in the figure 3.6, the wordcloud includes words such as good, thank, help, support, work, and hope.

In figure 3.7, the most common words in the wordcloud are covid19, but very few words that are conventionally positive or negative.

In figure 3.8, words such as death, death toll, Trump, death report and more.

The wordcloud highlights the clear differences between the negative, neutral, and positive tweets.

3.3.2 Data preprocessing

Data pre-processing is an integral part of data science and machine learning. This is due to the fact that without a dataset that has been prepared according to its intended implementation, any models that have been developed or will be developed may fail to run efficiently.

Pre-processing is also done with the purpose of discarding any information that is irrelevant to the study or analysis conducted.

Pre-processing encapsulates multiple functions in the context of this study, which will be discussed below.

Stop word removal: It has been observed that stop word removal had been implemented by various research papers where natural language processing was performed as seen in the literature review section of this study. Hence this functionality had been implemented.

Text cleaning: The objective of text cleaning is to remove certain text such as hyperlinks, hashtags, '@' mentions, exclamation marks and other punctuation marks. This has been done as they are considered irrelevant to sentiment analysis and removing them allows the number of words to be processed to be smaller, thereby making the model more efficient.

Lowercasing text: All the text of each tweet had been converted to lower case as words are case sensitive. Converting all the text to the same case significantly improves the feature selection efficiency. This was also seen to be performed in other research papers, thus leading to its implementation in this study.

```

1 stop_words = stopwords.words('english')
2 stemmer = SnowballStemmer('english')
3
4 text_cleaning_re = "@\S+|https?:\S+|http?:\S|[\^A-Za-z0-9]+"

# Function to pre-process the tweets
# Pre-processing involves removal of links, punctuations, words containing numbers, text in square brackets as well as convert
def preprocess(text, stem=True):
    text = re.sub(text_cleaning_re, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            if stem:
                tokens.append(stemmer.stem(token))
            else:
                tokens.append(token)
    return " ".join(tokens)

```

FIGURE 3.9

Executing the function shows above cleans and converts the text from figure 3.10 to figure 3.11

	user_name	user_location	date	text
0	astroworld	astroworld	2020-07-25 12:27:21	If I smelled the scent of hand sanitizers toda...
1	Tom Basile us	New York, NY	2020-07-25 12:27:17	Hey @Yankees @YankeesPR and @MLB - wouldn't it...
2	Time4fisticuffs	Pewee Valley, KY	2020-07-25 12:27:14	@diane3443 @wdunlap @realDonaldTrump Trump nev...
3	ethel mertz	Stuck in the Middle	2020-07-25 12:27:10	@brookbanktv The one gift #COVID19 has give me...
4	DIPR-J&K	Jammu and Kashmir	2020-07-25 12:27:08	25 July : Media Bulletin on Novel #CoronaVirus...

FIGURE 3.10

	user_name	user_location	date	text
0	astroworld	astroworld	2020-07-25 12:27:21	smell scent hand sanit today someone past would...
1	Tom Basile us	New York, NY	2020-07-25 12:27:17	hey yanke yankeespr mlb made sens player pay r...
2	Time4fisticuffs	Pewee Valley, KY	2020-07-25 12:27:14	wdunlap realdonaldtrump trump never claim covi...
3	ethel mertz	Stuck in the Middle	2020-07-25 12:27:10	one gift covid19 give appreci simpl thing alwa...
4	DIPR-J&K	Jammu and Kashmir	2020-07-25 12:27:08	25 juli media bulletin novel coronavirusupd co...

FIGURE 3.11

Sentiment Scoring: Since the dataset that had been chosen does not have any sentiments assigned to the tweets, it falls under data pre-processing where sentiment scoring must be performed to train the models later. This has been done using the Sentiment Intensity analyzer from the Vader library.

VADER implements a human-centric approach where it combines qualitative analysis and empirical validation with the help of human raters. It depends upon a dictionary where lexical features are mapped to emotion intensities called sentiment scored. Each word has its individual score between -4 and +4, and the sentiment score of the tweet is acquired by the sum of the individual sentiment intensities of the words which lies between -1 and +1 [29].

Though each word individually has a score between -4 and +4, the reason why the sentence score is

between -1 and +1 is that the results are normalized using the following formula:

$$\frac{x}{\sqrt{x^2 + \alpha}}$$

x is the total sum of the sentiment scored from the individual words and alpha is the normalization parameter.

```
1 # Sentiment is assigned to each tweet using SentimentIntensityAnalyzer from VADER
2
3 sia = SentimentIntensityAnalyzer()
4
5 # This function assigns sentiments
6 def scorer(text):
7     dict_res = sia.polarity_scores(text)
8     return dict_res["compound"]
9
10 # This function changes the score values such that tweets are sorted into only 3 categories, 1 if positive, 0 if neutral
11 # and -1 if negative
12 def sentiment_sorter(score):
13     if score > 0.25:
14         score = 1
15     else:
16         if score < -0.25:
17             score = -1
18         else:
19             score = 0
20     return score
21
22
```

Python

FIGURE 3.12

3.4 Feature extraction

TF-IDF: The TF-IDF vectorizer had been implemented as it has been seen to be used in multiple research papers which show that it is an efficient method of feature extraction in natural language processing. TF-IDF assigns each word a float value which signifies its importance in the entire corpus of words. Figure 3.13 and figure 3.14 shows its implementation.

```
1 max_features = 2000
2 embed_dim = 128
3 lstm_out = 196
4 batch_size = 128
5 maxlen = 80
6 nb_classes = 5
✓ 0.5s Python

1 # TTS
2
3 train, test = train_test_split(twt, test_size = 0.2)
4
5
6 # tfidf
7
8 vectorizer = TfidfVectorizer( min_df=2, max_df=0.95, max_features = 200000, ngram_range = ( 1, 4 ),sublinear_tf = True )
9
10 vectorizer = vectorizer.fit(train['text'])
11 train_features = vectorizer.transform(train['text'])
12 test_features = vectorizer.transform(test['text'])
13
14
15 # tfidf
16
17
18 fselect = SelectKBest(chi2 , k=10000)
19 train_features = fselect.fit_transform(train_features, train["score"])
20 test_features = fselect.transform(test_features)
✓ 32.5s Python

1 X_train = train_features.toarray()
2 X_test = test_features.toarray()
3
4 print('X_train shape:', X_train.shape)
5 print('X_test shape:', X_test.shape)
6 y_train = np.array(train['score']-1)
7 y_test = np.array(test['score']-1)
8
9 Y_train = np_utils.to_categorical(y_train, nb_classes)
10 Y_test = np_utils.to_categorical(y_test, nb_classes)
✓ 1.9s Python

X_train shape: (143286, 10000)
X_test shape: (35822, 10000)
```

FIGURE 3.13

```
1 # tfidf
2
3
4 # pre-processing: divide by max and subtract mean
5 scale = np.max(X_train)
6 X_train /= scale
7 X_test /= scale
8
9
10 mean = np.mean(X_train)
11 X_train -= mean
12 X_test -= mean
✓ 2m 46.6s Python
```

```
1 # tfidf
2
3 # convert into a matrix of tfidf vectors
4 tokenizer = Tokenizer(nb_words=max_features)
5 tokenizer.fit_on_texts(train['text'])
6 sequences_train = tokenizer.texts_to_sequences(train['text'])
7 sequences_test = tokenizer.texts_to_sequences(test['text'])
✓ 4m 27.5s Python
```

```
1 # tfidf
2
3 print('Pad sequences (samples x time)')
4 X_train = sequence.pad_sequences(sequences_train, maxlen=maxlen)
5 X_test = sequence.pad_sequences(sequences_test, maxlen=maxlen)
6
7 Y_train = np_utils.to_categorical(y_train, nb_classes)
8 Y_test = np_utils.to_categorical(y_test, nb_classes)
9
10
11 print('X_train shape:', X_train.shape)
12 print('X_test shape:', X_test.shape)
✓ 7.9s Python
```

Pad sequences (samples x time)
X_train shape: (143286, 80)
X_test shape: (35822, 80)

FIGURE 3.14

3.5 Model building

Both LSTM and Simple RNN were constructed as sequential models as we have singular input and outputs and do not require non-linear topography/

3.5.1 LSTM

A sequential LSTM model is defined with multiple layers to it.

Embedding layer: This is the first layer which represents words using a dense vector representation where the position of a word within the vector space is based on the surround words.

Spatial dropout layer: The spatial dropout layer is similar to the default dropout layer, which creates a random tensor of 0's and 1's where random coordinates in the tensor are dropped, but it applies the noise_shape property automatically. This essentially drops the same feature for all time steps but treats each sample individually.

LSTM layer: The next layer present is the LSTM layer with 196 neurons and the activation function that has been used is the 'softmax' activation. Softmax converts a vector of values to a probability distribution where the elements of the output vector fall within a range of 0 and 1 with their sum being equivalent to 1. The softmax is computed as: $\exp(x) / \text{tf.reduce_sum}(\exp(x))$ for each vector 'x' [30].

Dense layer: This is the final layer in the implemented model for LSTM. It is a deeply connected neural network layer that implements the following operation on the input to return the output [31]:

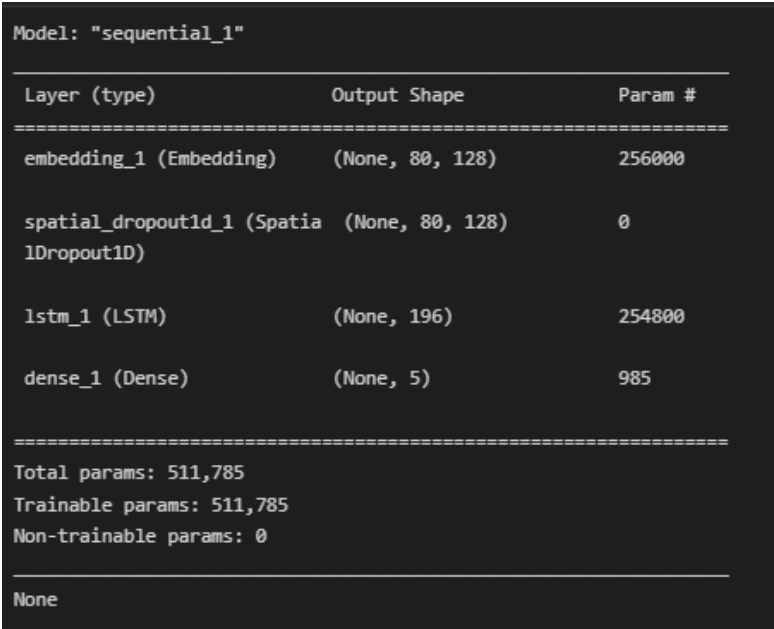
```
output = activation(dot(input, kernel) + bias)
```

where,

- **input** represent the input data
- **kernel** represent the weight data
- **dot** represent numpy dot product of all input and its corresponding weights
- **bias** represent a biased value used in machine learning to optimize the model
- **activation** represent the activation function.

FIGURE 3.15

The following figure shows the LSTM model that was constructed.



```
Model: "sequential_1"
Layer (type)                Output Shape                Param #
=====
embedding_1 (Embedding)     (None, 80, 128)            256000
spatial_dropout1d_1 (Spatial (None, 80, 128)            0
dropout1d)
lstm_1 (LSTM)                (None, 196)                254800
dense_1 (Dense)              (None, 5)                   985
=====
Total params: 511,785
Trainable params: 511,785
Non-trainable params: 0
None
```

FIGURE 3.16

3.5.2 Simple RNN

In the sequential RNN model implemented, there are primarily 3 layers.

Embedding layer: As mentioned above, the embedding layer is the first layer which represents words using a dense vector representation where the position of a word within the vector space is based on the surround words.

Simple RNN layer: The Simple RNN layer us a fully connected RNN. In this RNN, the output from the previous timestep has been fed to the following timestep. It has been implemented using ‘relu’ as the activation function and its graph is show in Figure 3.19 [32].

Dense Layer: This is the final layer of the Simple RNN that was made. As briefly discussed above, it is a deeply connected neural network layer that implements the following operation on the input to return the output [31].

Relu: This is represented with the formula $g(x) = \max(0, x)$

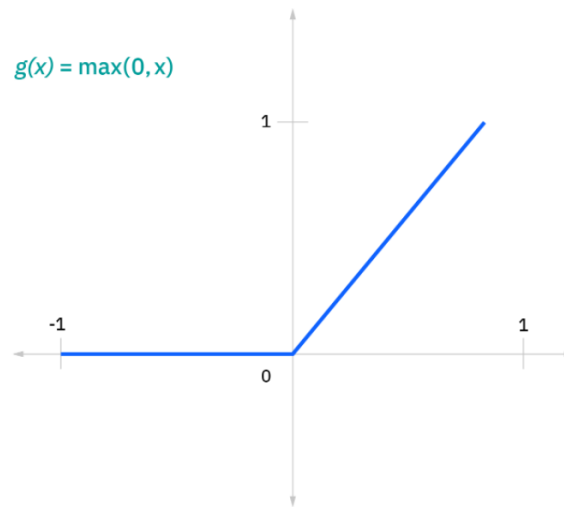


FIGURE 3.17

The following figure shows the Simple RNN that was constructed:

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 80, 128)	256000
simple_rnn (SimpleRNN)	(None, 16)	2320
dense_3 (Dense)	(None, 5)	85

```
=====  
Total params: 258,405  
Trainable params: 258,405  
Non-trainable params: 0  
=====  
None
```

FIGURE 3.18

3.5.3 Baseline models

For the baseline models, Naïve bayes, Random Forest and Logistic Regression were elected as they had been chosen in other research papers for a comparative analysis between the performances of different classifiers. This can be seen in [28], where they used these models for comparison of performance. Their implementation can be seen in the following figures:

Naive Bayes

```
[67] 1 # Initializing and fitting the Multinomial Naive Bayes model
      2
      3 model_nb = MultinomialNB(alpha = 0.001)
      4 model_nb.fit( train_features, train["score"])
Python

... MultinomialNB(alpha=0.001)

[68] 1 # Acquiring and storing the predictions
      2
      3 nb_pred = model_nb.predict( test_features.toarray() )
Python
```

FIGURE 3.19

Random Forest

```
[71] 1 # Initializing and fitting the Random forest model
      2
      3 model_rf = RandomForestClassifier()
      4 model_rf.fit( train_features, train["score"] )
Python

Outputs are collapsed ...

[72] 1 # Acquiring and storing the predictions
      2
      3 rf_pred = model_rf.predict( test_features.toarray() )
Python
```

FIGURE 3.20

Logistic Regression

```
[74] 1 # Initializing and fitting the Logistic regression model
      2
      3 model_lr = LogisticRegression(solver='liblinear', random_state=0)
      4 model_lr.fit(train_features, train["score"])
Python

... LogisticRegression(random_state=0, solver='liblinear')

[75] 1 # Acquiring and storing the predictions
      2
      3 lr_pred = model_lr.predict( test_features.toarray() )
Python
```

FIGURE 3.21

Chapter 4

Critical review of implemented models

In the following chapter, the performance of the two deep learning algorithms, LSTM and RNN, will be analyzed and further compared to Naïve bayes, Random Forest and Logistic regression with a critical evaluation of the findings acquired. The matrices to evaluate the performance of the algorithms are:

1. Time taken to train the model and predict the sentiments
2. Accuracy

4.1 Performance evaluation

In order to understand how the performances of the algorithms are, we must first begin by understanding and highlighting the input dataset.

The input dataset contains cleaned tweets with a sentiment score assigned to them, where -1 correlates to negative sentiment, 0 correlates to neutral sentiment and +1 correlates to positive sentiment. However, it is important to note the basis upon which the tweets were set to one of the three values. If the sentiment was below -0.25, they were considered to possess negative sentiment, if they were above 0.25, they were assigned as positive sentiment, and if they didn't meet either of the criteria, they were assigned the value of 0 indicating that they were neutral (Figure 4.1 and Figure 4.2).

```
1 # Sentiment is assigned to each tweet using SentimentIntensityAnalyzer from VADER
2
3 sia = SentimentIntensityAnalyzer()
4
5 # This function assigns sentiments
6 def scorer(text):
7     dict_res = sia.polarity_scores(text)
8     return dict_res["compound"]
9
10 # This function changes the score values such that tweets are sorted into only 3 categories, 1 if positive, 0 if neutral
11 # and -1 if negative
12 def sentiment_sorter(score):
13     if score > 0.25:
14         score = 1
15     else:
16         if score < -0.25:
17             score = -1
18         else:
19             score = 0
20     return score
21
22
```

✓ 0.2s Python

FIGURE 4.1

```
1 # The sentiment scores are now sorted into 1, 0 and -1 corresponding to Positive, neutral and negative respectively
2
3 twt['score'] = twt['score'].apply(lambda x: sentiment_sorter(x))
```

✓ 0.1s Python

FIGURE 4.2

This was done to simplify the training process with only 3 sentiments possible, instead of the sentiments possibly being one of every float value between -1 and +1.

Lastly, the training and testing sets were creating in a manner where 80% of the data was training data and 20% was the testing data (Figure 4.3).

```
1 # TTS
2
3 train, test = train_test_split(twt, test_size = 0.2)
4
```

FIGURE 4.3: Train-Test splitting

4.1.1 VADER Sentiment Intensity Analyzer

In order to assess the performance of the Vader Sentiment Intensity analyzer, few tweets from the dataset will be analyzed by a human reader and their classification compared to the classification executed by the Sentiment analyzer.

Tweet: @brookbanktv The one gift #COVID19 has give me is an appreciation for the simple things that were always around me

Sentiment assigned by VADER: Positive

Sentiment assigned by Human: Positive

Tweet: You now have to wear face coverings when out shopping - this includes a visit to your local Community Pharmacy

Sentiment assigned by VADER: Neutral

Sentiment assigned by Human: Neutral

Tweet: #coronavirus #covid19 deaths continue to rise. It's almost as bad as it ever was. Politicians and businesses want your money. But it is not safe. #STAYatHOME

Sentiment assigned by VADER: Negative

Sentiment assigned by Human: Negative

Tweet: It is during our darkest moments that we must focus to see the #light.~Aristotle

Sentiment assigned by VADER: Negative

Sentiment assigned by Human: Positive

It can be seen that the sentiment intensity analyzer is clearly not completely accurate, this is an issue that persists with any sentiments that are annotated automatically and not manually by humans.

4.1.2 Deep Learning models

LSTM

They key detail to be taken note of was the duration it took for LSTM to complete the training process. For 5 epochs, it took a total time of 4317 seconds, which translates to approximately 1 hour and 12 minutes (Figure 4.4).

```
4 lstm.fit(X_train, Y_train, epochs = 5, batch_size=batch_size)

Train...
Epoch 1/5
1120/1120 [=====] - 884s 776ms/step - loss: 1.1702 - accuracy: 0.5298
Epoch 2/5
1120/1120 [=====] - 863s 771ms/step - loss: 0.7537 - accuracy: 0.6802
Epoch 3/5
1120/1120 [=====] - 860s 768ms/step - loss: 0.6306 - accuracy: 0.7468
Epoch 4/5
1120/1120 [=====] - 856s 764ms/step - loss: 0.4621 - accuracy: 0.9143
Epoch 5/5
1120/1120 [=====] - 854s 762ms/step - loss: 0.3997 - accuracy: 0.9161
<keras.callbacks.History at 0x27c1bae5790>
```

FIGURE 4.4: LSTM training

It took another additional 43 seconds to evaluate the score and accuracy of the algorithm (Figure 4.5).

```
1 # Model results
2
3 score, acc = lstm.evaluate(X_test, Y_test, batch_size=batch_size)
4 print('Test score:', score)
5 print('Test accuracy:', acc)

280/280 [=====] - 43s 150ms/step - loss: 0.3760 - accuracy: 0.9199
Test score: 0.3760015368461609
Test accuracy: 0.9199374914169312
```

FIGURE 4.5: LSTM Results

The perfect LSTM model would have an accuracy of 100%, and a loss of 0. In this case, the loss achieved is only 0.376 and the resulting accuracy of the algorithm was exactly 91.99374914169312%, which can be rounded off to 92%, finally the resulting score of LSTM was calculated to be 0.37600

RNN

RNN, being another Deep Learning algorithm, is significantly simpler than LSTM. The difference is that RNN's do not possess a cell state, and only contain hidden states, while LSTM possesses the hidden states as well as the cell states. The affect in the performance in significant as RNN showed to be slightly over 30 times quicker than LSTM, while providing a higher accuracy and a lower loss and score (Figure 4.7).

The training time for the RNN only took 141 seconds, which translates to 2 minutes and 21 seconds in the same number of epochs as LSTM (Figure 4.6).

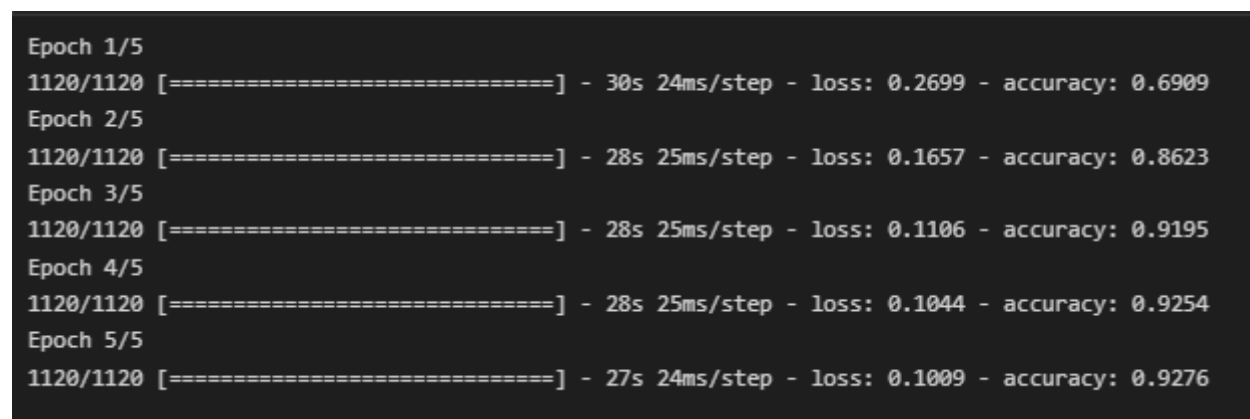


FIGURE 4.6

The resulting accuracy was 92.79772043228149%, or approximately 93%, this is already slightly better than the results achieved by LSTM. Looking at the loss and score, the loss is significantly lower at only 0.1031 and the score is approximately 0.10306 (Figure 4.7).

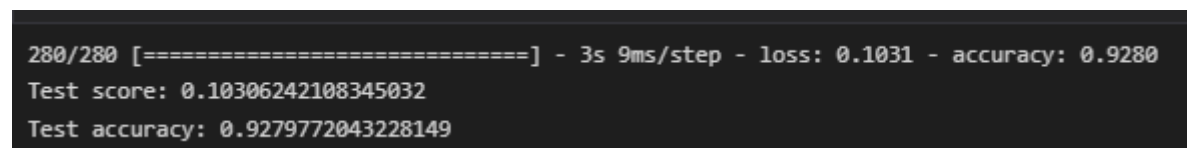


FIGURE 4.7

The acquired results by RNN are better. But that can vary based on the size of the dataset. LSTM also has many benefits which lead to the user to have more control over the implementation itself. Based on these results which correspond to the dataset chosen, RNN has outperformed LSTM. This however can change depending on the intended purpose of a model as well as the type and size of dataset as LSTM offers more control which could lead to better results at the cost of complexity and time.

4.1.2 Baseline models

Naïve Bayes

The Naïve Bayes algorithm is fairly simple and efficient in terms of the amount of time it takes to train and provide results as well as utilized a fairly low amount of system resources. However, it has shown to perform well on a task such as sentiment analysis on a dataset containing over 175 thousand tweets.

In this model, Multinomial Naïve Bayes has been implemented which uses the Bayes theory to predict the sentiment of a given tweet.

The Bayes Theorem, which employed by the Naïve Bayes algorithm, is as follows:

$$P(A|B) = (P(B|A) \times P(A)) / P(B)$$

Looking at the accuracy of Naïve bayes, it achieved an approximate accuracy of 76% as seen in Figure 4.8. In the total 5 algorithms tested, Naïve bayes has had the worst performance. However it was also the second quickest, taking only around 6 seconds to complete training, and another 6 seconds to predict and store the predictions (Figure 4.9).

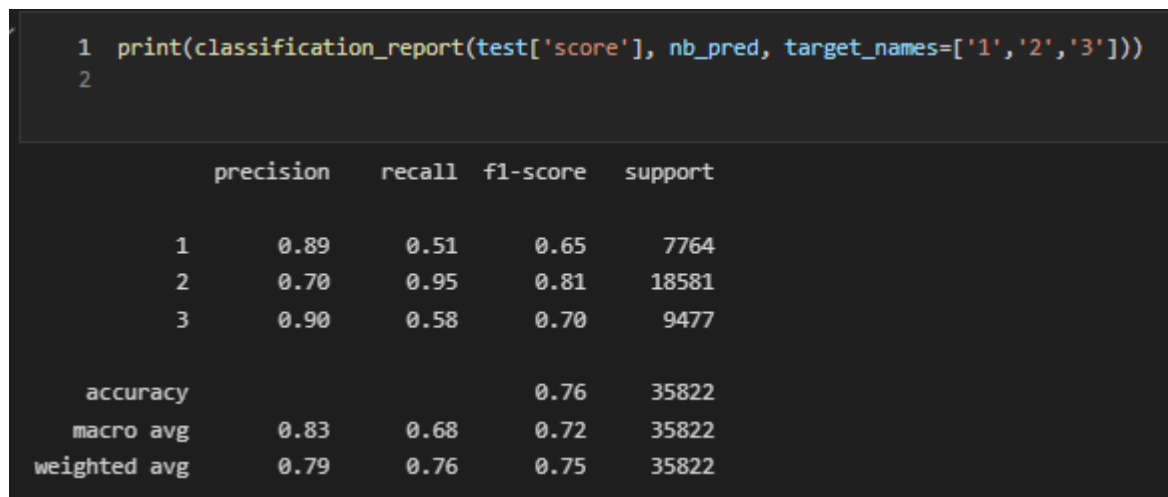


FIGURE 4.8

```
1 # Initializing and fitting the Multinomial Naive Bayes model
2
3 model_nb = MultinomialNB(alpha = 0.001)
4 model_nb.fit( train_features, train["score"])
✓ 6.3s

MultinomialNB(alpha=0.001)

1 # Acquiring and storing the predictions
2
3 nb_pred = model_nb.predict( test_features.toarray() )
✓ 6.1s
```

FIGURE 4.9

Random Forest

Among the baseline models, Random Forest was the most time and resource intensive to run. It took a total of 6 minutes and 41 seconds to complete model fitting and predicting. However, it provides results that justify its longer runtime by providing the highest accuracy among all the algorithms tested achieving almost 94% accuracy. From this analysis, it can be seen that Random Forest can outperform simple deep learning models for natural language processing and sentiment analysis on small scale datasets. However due to the nature of Random Forest and it essentially being multiple decision trees represented as one algorithm, it would be extremely slow and inefficient for NLP on datasets which may not be in tabular form and where there are millions of tweets to process. Its performance can be seen in Figure 4.10.


```
1 # Initializing and fitting the Random forest model
2
3 model_rf = RandomForestClassifier()
4 model_rf.fit( train_features, train["score"] )
✓ 6m 19.7s

RandomForestClassifier()

1 # Acquiring and storing the predictions
2
3 rf_pred = model_rf.predict( test_features.toarray() )
✓ 22.2s

1 # Showing the resulting accuracy of the Random forest model
2
3 print('accuracy: ', accuracy_score(test['score'], rf_pred))
✓ 0.1s

accuracy: 0.9379710792250572
```

FIGURE 4.10

Logistic Regression

Logistic regression is used in a variety of NLP processes. Its primarily used in cases where binary classification is required, however in this case, there are 3 classifications possible, Positive sentiment, Neutral sentiment and Negative sentiment. The core function that Logistic regression implements is the logistic function which is as follows: $1 / (1 + e^{-\text{value}})$ where e is the natural logarithms base.

Analyzing the runtime for logistic regression, it was the quickest, completing training and predicting under 10 seconds. Its accuracy was average in comparison to the other models. Overall, it can be said that logistic regression is possibly one of the fastest methods for sentiment analysis offering good, but not the best, accuracy results.

```

Logistic Regression

1 # Initializing and fitting the Logistic regression model
2
3 model_lr = LogisticRegression(solver='liblinear', random_state=0)
4 model_lr.fit(train_features, train["score"])
[47] ✓ 6.6s

... LogisticRegression(random_state=0, solver='liblinear')

1 # Acquiring and storing the predictions
2
3 lr_pred = model_lr.predict( test_features.toarray() )
[48] ✓ 3.1s

1 # Showing the resulting accuracy of the Logistic Regression model
2
3 print('accuracy: ', accuracy_score(test['score'], lr_pred))
[49] ✓ 0.7s

... accuracy: 0.8826698676790798

```

FIGURE 4.11

Constructing tables based on the matrices that have been used for this analysis, mainly Time to run and Accuracy, the results are as follows:

Rank (fastest to slowest)	Algorithm	Time to run in seconds
1	Logistic regression	9.7 seconds
2	Naïve Bayes	12.4 seconds
3	RNN	144 seconds
4	Random Forest	401.9 seconds
5	LSTM	4317 seconds

TABLE 4.1: The time calculates is the sum of training time and prediction time

Rank (most accurate to least)	Algorithm	Accuracy (precise to 0.01%)
1	Random Forest	93.79%
2	RNN	92.79%
3	LSTM	91.99%
4	Logistic Regression	88.26%
5	Naïve Bayes	75.77%

TABLE 4.2

4.2 Limitations

Mentioned below are some of the limitations of the project that has been implemented.

- Due to recent changes with Twitter, a dataset containing tweets from a desired time could not be created. This forced the reliance on a pre-existing dataset which only collected tweets based on the COVID19 hashtag
- Due to the processing limitations, the implemented sequential LSTM had to be kept simple. More sophisticated and efficient designs would yield better performance results.
- Due to unavailability of datasets, tweets with different search criteria could not be acquired. The available datasets only had one or two search criteria for the tweets, thereby limiting the tweet variety.

Chapter 5

Conclusion and future work

Within this section, an overview of the functional requirements that were fulfilled will be presented, following the conclusion of this study, access to the source code of the project as well as the challenges faced and the possibilities of future work.

5.1 Requirements validation

All of the mentioned functional requirements that were established at the beginning of the project have been completed. A table describing this is present below:

Functional Requirements		
S. No	Description	Status
FR-1	Data pre-processing Performing removal of irrelevant data, capital conversion, stop-word removal and word normalization	Completed
FR-2	Extract the sentiment of tweets relating to COVID-19	Completed
FR-3	Applying different ML classifiers for sentiment analysis Apply different ML classifiers on the dataset	Completed
FR-4	Comparative analysis Compare and review the results of different models.	Completed

TABLE 5.1: Functional requirements status

5.2 Conclusion

Different studies have adopted different approaches to sentiment analysis. Their goals ranged from comparing different baseline models to each other, to comparing their models to baseline models, to just implementing their model to assess twitter sentiments. They used a variety of strategies for data pre-processing, feature extraction, and model construction.

The goal of this project was to allow readers to understand how the sentiment of the public has shifted over time and to assess the performance of different algorithms and compare them. Through a graphical representation, it was shown how the sentiment had shifted.

Observing the results of the critical review and analyzing the results of the model based on the matrices taken into consideration, Logistic Regression is the fastest algorithm among the 5, however in terms of

accuracy, it falls behind both of the Deep Learning algorithms. On the basis of accuracy, Random Forest is the most accurate, closely followed by both RNN and LSTM, but it is the second slowest. Taking all factors into account, a simple RNN provides the most optimal results giving a very high accuracy and low runtime within the scope of the dataset used.

5.3 Project source code

The project code was implemented in a Jupyter Notebook using Visual Studio Code. The source code provided is present in the Python Programming Language. The code for this project can be used and modified without requiring permission.

5.4 Challenges

As a whole, the learning process and the experience gained from the project was interesting and enriching. However, challenges presented themselves through the implementation of the project.

Data set availability:

The original idea for the project was to create a database by acquiring tweets from twitter using the Twint tool. This would allow to access tweets from a desired date and time frame. However due to recent changes in the twitter API, the resulting changes prevented any tools, such as Twint, from scraping tweets off of twitter. This caused severe limitations and the desired dataset could no longer be constructed. This eventually resulting in making use of a pre-existing dataset, which only got close to the desired data range.

Existing Models:

Most of the papers that were analyzed and researched on the topic regarding machine learning, sentiment analysis and natural language processing failed to provide the code implementation. This added an extra layer of challenge to develop the implementation from scratch. Some of the papers also failed to provide clear documentation of the exact methods and practices they executed in their code, which only lead to more research being done as well as following more general standards of implementation.

5.5 Future work

The performances of the different machine learning algorithms have been compared in the context of sentiment analysis of Tweets. However, there is plenty of scope for addition of unique features as well as optimization and improvement of the code.

1. Dataset variety:

Currently, only datasets that have the entire tweet stored on the dataset without annotation can be optimally used for sentiment analysis. Possible features such as checking if the tweets are annotated with sentiments and allowing input from datasets that require tweet hydration could be implemented to allow a larger variety of datasets to be compatible with the code as well as improving the efficiency of the program.

2. Dataset optimization:

As seen earlier, the tweets had many redundant location tags, removing the redundancies and duplicate location tags could vastly provide more accurate information about the location of tweets as well as reduce the total number of individual location tags present.

Appendix A

Requirement Analysis

A.1 Functional requirements

Functional requirements define the behavior of the system. The definition of Functional requirements is “Any requirement which specifies what a system should do” [33].

Functional Requirements		
S. No	Description	Priority
FR-1	Data Pre-processing Performing removal of irrelevant data, capital conversion, stop-word removal and word normalization.	M
FR-2	Extract the sentiment of tweets relating to COVID-19	M
FR-3	Applying different ML classifiers for sentiment analysis Apply different ML classifiers on the dataset	M
FR-4	Comparative analysis Compare and review the results of different models.	S

A.2 Non-Functional requirements

Non-Functional requirements specify how a system behaves rather than what the system does. The definition of non-functional requirements is “Any Requirement That Specifies How the System Performs a Certain Function.” [33]

Non-Functional Requirements		
S. No	Description	Priority
NFR-1	Scalability The code should be designed in a manner that it allows for future work and developments.	S
NFR-2	Readability The code must be easy to read with comments explaining different sections appropriately	M
NFR-3	The code must be able to be ported to different operating systems and be easily accessible. A jupyter Notebook script should be published.	M
NFR-4	Accessibility The resulting code should be open sourced and be published on GitHub so that others may view and make improvements.	M

Appendix B

Project management and development methodology

B.1 Development methodology

The project will be developed in primarily 3 stages:

Stage 1: Data Pre-processing

Stage 2: Model Development and Testing

Stage 3: Model evaluation and analysis

Stage 1 involves methods such as stopwords removal, tokenization, word normalization and feature extraction. Following this, the dataset will be split into training and testing data. This will be done to optimize the dataset for its use in the model during the later stages of the project.

Stage 2 will be focusing on the implementation of the model. The model will focus on the implementation of different deep learning algorithms, LSTM and RNN, which will be trained and tested on the dataset to predict the tweet sentiments. Other baseline models such as Naïve bayes will also be trained and tested on the dataset.

Stage 3 aims to analyze the results of the different deep learning algorithms and further comparing them to other models using matrices such as Confusion Matrix, Accuracy, Precision, F measure, etc.

B.2 Project management

B.2.1 Risk management

Risk management is an integral part of any project development. It helps identify risks and allows the progress to follow the planned timeline during the implementation of the project.

The main steps involved when it comes to risk management are as follows:

1. Risk Identification: Risks are present in distinct types such as tools, people, time, and requirements. This stage ensures that the risks are understood and categorized.

2. Risk analysis: Once the risks have been identified, the next stage is to determine the likelihood of the risk to occur along with how severely it will impact the project's development. The levels of likelihood and severity are color coded as follows:

Color	Likelihood	Severity
Red	High	Catastrophic
Yellow	Medium	Serious
Green	Low	Tolerable

3. Risk planning: It is vital to document the possible strategies that will assist in mitigating risk and help in solving the circumstances that may present themselves.

Risk Management		
Risk, Type, Likelihood, Impact	Planning and monitoring	
	Monitoring and Avoidance	Contingency and Minimization
Data Loss Type: Tools Likelihood Impact	Monitoring: File save times are large and some parts may be lost. Avoidance: Perform regular backups and maintain appropriate version control	Contingency: Restart the project Minimize: Use the documentation prepared while project development to recover from the setback
Data-processing Type: Tools Likelihood Impact	Monitoring: Processing may take a long time which can delay the project. Avoidance: Ensure that the tool has been used before for our intent.	Contingency: Ensure continuation on alternative device. Minimize: Use more time efficient tools for processing
Running behind project plan Type: Time Likelihood Impact	Monitoring: Required time was not dedicated to the tasks.Avoidance: Work in advanceand attempt to stay ahead of schedule	Contingency: Begin the implementation of the following task after one is over before the start date to save time.
		Minimize: Make a draft version of plan along with processes and other details and compile it later.
Health and Deadlines	Monitoring: Deadlines overlapping with project plan.	Contingency: Plan completion dates while keeping possible

Type: People	Physical and mental health	deadlines in mind.
Likelihood	experiencing shifts.	Minimize: Maintain a healthy
Impact	Avoidance: Adjust time to accommodate deadlines and project plan. Maintain a stable and healthy environment.	schedule and diet and re-organize timeline according to other deadlines.

B.2.2 Project implementation and analysis

This section is the list of ideations of the model to be implemented, data processing, software to use and evaluation strategy.

Implementation and evaluation

1. Data collection

As seen from the literature review, most authors used web-scraping tools to collect tweets and compile their own datasets. Meanwhile, twitter no longer allows web-scraping via Third-party methods and only allows the acquisition of tweets through their own API. Since this does not allow us to collect tweets from the time frame where COVID was still in its developing stages with the vaccines being developed and severe lockdowns in place, a pre-existing dataset was chosen for this study.

2. Data processing

Tokenization, cleaning of data and lowercasing of text will be used to process the tweets.

Testing: The processing of the tweets should be testing to make sure that there are no conflicts or issues when providing input to the model.

Evaluation: Sentence modelling technique will be performed to make sure that the processing that will be conducted is valid.

3. Sentence Modelling

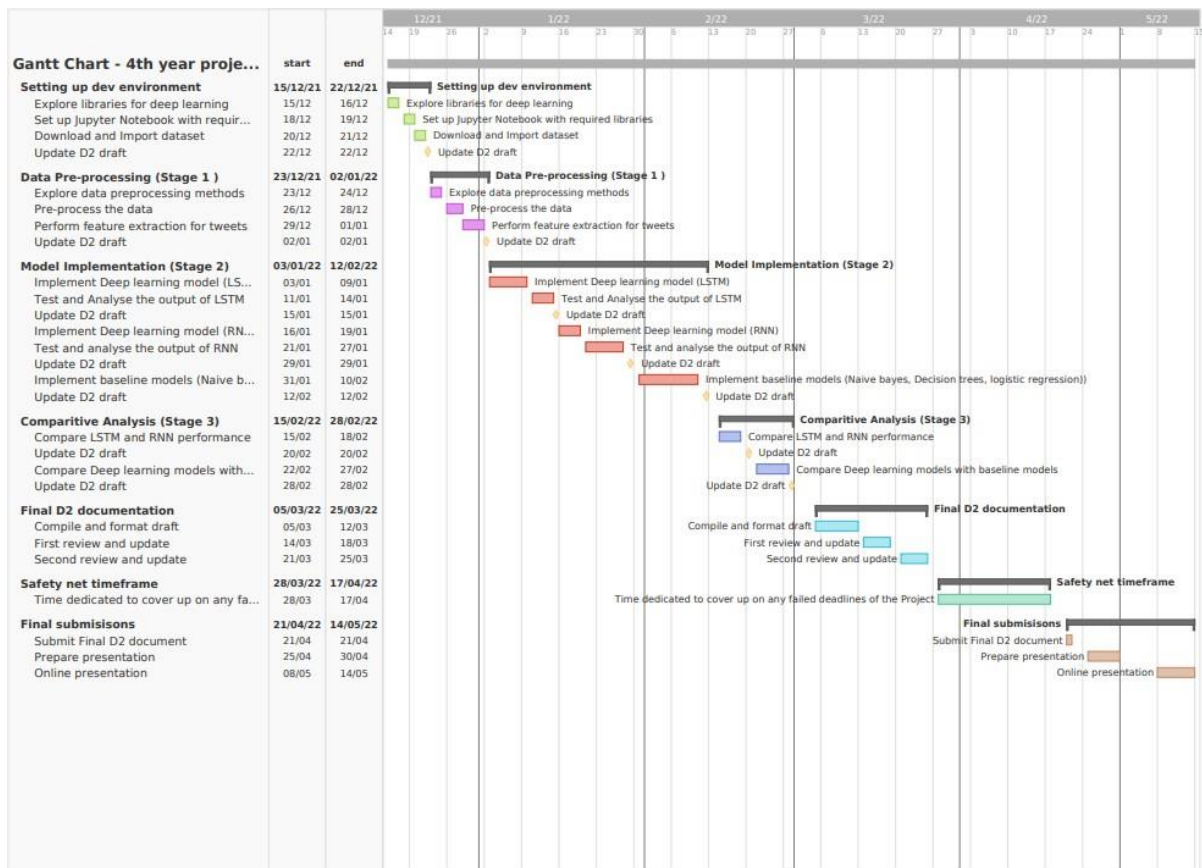
The processed tweets from the dataset will be converted to their vector representation that carry the context of the tweets.

Testing: Deep learning algorithms will be applied to the dataset, later on baseline models would also be applied to the dataset.

Evaluation: The performance of the model will be evaluated through the comparison of their measurement matrices such as accuracy, time taken to train the model and make predictions. The performance of the model will also be compared to the performance of the baseline models using the same matrices.

B.3 Project plan

The following represents the project plan. It outlines the project deliverables with their respective deadlines. This chart shall serve as a timetable and assist us in keeping track of our goals and deliverables.



There has also been allocated a time frame (marked in green in the chart) that will allow us to coverup for any lost time due to unseen circumstances. It will help the project to be completed on time.

Appendix C

Discussion of professional, legal, ethical, and social issues

C.1 Professional and Legal issues

Research papers that have been used will be referenced properly. If any code or part thereof has been used and/or implemented, it shall be documented and acknowledged.

The dataset used is freely available to all public and does not inflict any rights that have been or maybe reserved.

The project will be written in python, making use of public libraries. The program will be developed in Jupyter Notebook and the source code will be published under a GNU General Public License as follows:

The program is a free software. It can be redistributed and/or modified under the terms specified by the GNU General Public License as published by the Free Software Foundation, version 3.

This program will be distributed such that it may be useful, but without any warranty, without even the implied warranty of merchantability or fitness for a particular purpose. See GNU General Public License for more information

C.2 Ethical and Social Issues

This study does not involve any testing with human subjects or confidential data in any shape or form. The dataset that will be used only contains information that is publicly available. It consists of tweets that were made publicly and therefore does not require permission for use. No risk is involved as all the data that will be used is freely available online.

Chapter 6

References

- [1] C. A. Iglesias and A. Moreno, "Sentiment Analysis for Social Media," *Applied Sciences*, vol. 9, no. 23, p. 5037, Nov. 2019.

<https://www.mdpi.com/2076-3417/9/23/5037>
- [2] M. D. Shoaie and M. Dastani, "The Role of Twitter During the COVID-19 Crisis: A Systematic Literature Review," *Acta Informatica Pragensia*, vol. 9, pp. 154-69, 2020.

<http://doi.org/10.18267/j.aip.138>
- [3] A. Alsaeedi and M. Z. Khan, "A Study on Sentiment Analysis Techniques of Twitter Data," *International Journal of Advanced Computer Science and Applications(Ijacs)*, 10(2), 2019.

<http://dx.doi.org/10.14569/IJACSA.2019.0100248>
- [4] W. Chung, S. He and D. D. Zeng, "eMood: Modeling Emotion for Social Media Analytics on Ebola Disease Outbreak Research-in-Progress." *International Conference on Information Systems*, 2015

https://www.researchgate.net/publication/329876555_eMood_Modeling_Emotion_for_Social_Media_Analytics_on_Ebola_Disease_Outbreak_Research-in-Progress
- [5] K. Chakraborty, S. Bhatia, S. Bhattacharyya, J. Platos, R. Bag and A. Ella Hassanien, "Sentiment Analysis of COVID-19 tweets by Deep Learning Classifiers—A study to show how popularity is affecting accuracy in social media", *Applied Soft Computing*, Volume 97, Part A, 2020.

<https://doi.org/10.1016/j.asoc.2020.106754>
- [6] G. Barkur, Vibha, G. B. Kamath. "Sentiment analysis of nationwide lockdown due to COVID 19 outbreaks: Evidence from India". *Asian journal of psychiatry*. 2020

<https://doi.org/10.1016/j.ajp.2020.102089>

- [7] R. Gupta, A. Vishwanath, Y. Yang, “COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes,” arXiv e-prints, 2020

<https://doi.org/10.3886/E120321V11>

- [8] R. Lamsam, “Coronavirus (COVID-19) Geo-tagged Tweets Dataset,” IEEE Dataport, 2020

<https://dx.doi.org/10.21227/fpsb-jz61>

- [9] A. Gruzd and P. Mai, “COVID-19 Twitter Dataset,” Scholars Postal Dataverse, 2020

<https://doi.org/10.5683/SP2/PXF2CU>

- [10] G. Preda, “COVID19 Tweets,” Kaggle, 2020

<https://www.kaggle.com/gpreda/covid19-tweets>

- [11] K. Goyal, “Data Pre-processing in Machine Learning: 7 Easy Steps to Follow”, upGrad, 2021

<https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/>

- [12] L. Batista and L. A. Alexandre, “Text Pre-processing for Lossless Compression”. Data Compression Conference (dcc 2008), 2008, pp. 506-506

<https://ieeexplore.ieee.org/document/4483333>

- [13] M. Singh, A.K. Jakhar, and S. Pandey, “Sentiment analysis on the impact of coronavirus in social life using the BERT model.” Social Network Analysis Mining 2021

<https://doi.org/10.1007/s13278-021-00737-z>

- [14] S. Singh, “” NLP Essentials: Removing Stop words and Performing Text Normalization using NLTK and spaCy in Python,” Analytics Vidhya, 2019

<https://www.analyticsvidhya.com/blog/2019/08/how-to-remove-stopwords-text-normalization-nltk-spacy-gensim-python/>

- [15] S. Saxena, "Tokenization and Text Normalization," Analytics Vidhya, 2021

<https://www.analyticsvidhya.com/blog/2021/03/tokenization-and-text-normalization/>

- [16] M. Javed, S. Kamal, "Normalization of Unstructured and Informal Text in Sentiment Analysis," International Journal of Advanced Computer Science and Applications, Vol. 9, No. 10, 2018

<https://dx.doi.org/10.14569/IJACSA.2018.091011>

- [17] TechTarget Contributor, "Definition: Lemmatization," TechTarget, 2018

<https://searchenterpriseai.techtarget.com/definition/lemmatization>

- [18] A. Dinakaramani, F. Rashel, A. Luthfi and R. Manurung, "Designing an Indonesian part of speech tagset and manually tagged Indonesian corpus," 2014 International Conference on Asian Language Processing (IALP), pp. 66-69, 2014

<https://doi.org/10.1109/IALP.2014.6973519>

- [19] H. Mujtaba, "An introduction to Bag of words (BoW) | What is Bag of Words?" Great Learning, 2020

<https://www.mygreatlearning.com/blog/bag-of-words/>

- [20] R. Chandra, A. Krishna, "COVID-19 sentiment analysis via deep learning during the rise of novel cases." PLOS ONE 16(8): e0255615, 2021

<https://doi.org/10.1371/journal.pone.0255615>

- [21] "What is LSTM? Introduction to Long Short-Term Memory," Intellipaat, 2021

<https://intellipaat.com/blog/what-is-lstm/>

[22] 1.10 Decision Trees

<https://scikit-learn.org/stable/modules/tree.html>

[23] “Machine Learning – Logistic Regression”, tutorialspoint

https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.htm#:~:text=Logistic%20regression%20is%20a%20supervised,probability%20of%20a%20target%20variable.&text=It%20is%20one%20of%20the,Diabetes%20prediction%2C%20cancer%20detection%20etc.

[24] R. Horev, “BERT Explained: State of the art language model for NLP”, towardsDatascience 2018

<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

[25] M. Beck, “How to Scrape Tweets from Twitter”, towardsDatascience, 2020

<https://towardsdatascience.com/how-to-scrape-tweets-from-twitter-59287e20f0f1>

[26] Z. Zhang , M.R Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” NIPS, pp: 8792-8802, 2018

<https://dl.acm.org/doi/epdf/10.5555/3327546.3327555>

[27] K. Dembczyński, W. Waegeman, W. Cheng and E. Hüllermeier, “On label dependence and loss minimization in multi-label classification.” Mach Learn 88, 2012.

<https://doi.org/10.1007/s10994-012-5285-8>

[28] H. Krishnan, P. G, A. Poosari, A. Jayaraj, C. Thomas, and G. M. Joy, "Machine Learning based Sentiment Analysis of Coronavirus Disease Related Twitter Data," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), 2021

<https://doi.org/10.1109/ICSCCC51823.2021.9478145>

- [29] Pio Claderon, “VADER Sentiment Analysis Explained,” medium, 2017
<https://medium.com/@piocalderon/vader-sentiment-analysis-explained-f1c4f9101cd9#:~:text=Quantifying%20the%20Emotion%20of%20a,each%20word%20in%20the%20text.>
- [30] “Layer activation functions,” Keras
<https://keras.io/api/layers/activations/>
- [31] “Keras – Dense layer,” tutorialspoint
[https://www.tutorialspoint.com/keras/keras_dense_layer.htm#:~:text=Dense%20layer%20is%20the%20regular,input%2C%20kernel\)%20%2B%20bias](https://www.tutorialspoint.com/keras/keras_dense_layer.htm#:~:text=Dense%20layer%20is%20the%20regular,input%2C%20kernel)%20%2B%20bias)
- [32] IBM Cloud Education, “Recurrent Neural Networks,” IBM, 2020
<https://www.ibm.com/cloud/learn/recurrent-neural-networks>
- [33] Reqtest, “Why is the difference between functional and Non-functional requirements important?,” ReQtest, 2012
<https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements>