Twitter Emotion Analysis – An LSTM Approach

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering
Spl. in Information Security

by

ROHAN ALLEN 18BCI0247

Under the guidance of

Dr. MANIKANDAN N

SCOPE

VIT, Vellore.

June, 2022

DECLARATION

I hereby declare that the thesis entitled **“Twitter Emotion Analysis – An LSTM Approach”** submitted by me, for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (Spl. in Information Security)** to VIT is a record of bonafide work carried out by me under the supervision of **Dr. Manikandan N**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidate
Rohan Allen(18BCI0247)

CERTIFICATE

This is to certify that the thesis entitled **Twitter Emotion Analysis – An LSTM Approach**" submitted by **Rohan Allen (18BCI0247)**, **SCOPE**, VIT University, for the award of the degree of ***Bachelor of Technology in Computer Science and Engineering (Spl. in Information Security)***, is a record of bonafide work carried out by him under my supervision during the period, 12. 03. 2022 to 14. 06. 2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

Information Security

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank **Dr. G.Viswanathan**, Chancellor, **and Dr. Rambabu Kodali**, Vice Chancellor, for extending the facilities of both the School and University towards this project. I would also like to thank the University of Plymouth for providing me with all the support required to complete the project successfully.

I am also grateful to **Dr. RAMESH BABU K**, dean of scope school and **Dr. SATHYARAJ R**, HOD of Information security for their constant support and guidance.

I would like to extend my sincere gratitude towards **Dr. Manikandan N**, my guide and Associate Professor under SCOPE school and **Dr. Marco Palomino**, Associate Professor of Big Data and Information Systems under the School of Engineering, Computing and Mathematics (Faculty of Science and Engineering) at the University of Plymouth who has been supervising my project and providing me with valuable guidance and insights.

I also wish to thank all my family and friends for their constant encouragement. Lastly, I would like to also thank all the faculties in both VIT Vellore and the University of Plymouth for their unending support throughout the course of the project.

Student Name

Rohan Allen

Executive Summary

The primary goal of this project is to stream real-time COVID-19 related tweets using Twitter and its Python API-Tweepy and then conduct a sentiment and emotion analysis on the same. The results are compared with a corpus of tweets collected two years ago. Initially lexicon-based tools such as Text2Emotion, VADER, TextBlob, etc were considered. To incorporate machine learning concepts, we also utilized some pre-trained emotion detection models from the work published by Colnerič and Demšar. These models were able to predict emotions with regards to some famous psychological models such as Ekman, Plutchik and the Profile of Mood States. However, we decided to build our very own deep learning model due to some deficiencies discovered in the pre-trained models.

A bi-directional RNN LSTM neural network model was trained on a labelled emotion dataset to predict the emotions on our tweet collection. This was successful in generating more realistic and accurate results when compared to the previous models. Furthermore, a comprehensive data analysis and visualization was also performed on the corpus of tweets to give us further insights and clarity on the mood of the public towards COVID-19 and how they have changed.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

RNN	Recurrent Neural Network
NLP	Natural Language Processing
JSON	JavaScript Object Notation
TF-IDF	Term Frequency- Inverse Document Frequency
LSTM	Long Short Term Memory
API	Application Programming Interface

1 INTRODUCTION

Many individuals throughout the globe are increasingly utilising blogs, forums, and social media sites like Twitter and Instagram to express their beliefs to the wider world. Social media has evolved into arguably one of the most powerful communication platforms known to mankind. As a consequence, a significant amount of data is created, referred to as big data, and sentiment analysis was developed to make sense of this huge amount of data.

The method of extracting relevant information and semantics from text utilising natural language processing techniques and detecting the polarity of the text, which could exhibit negative, positive or neutral characteristics, is known as sentiment analysis.

Machine learning-based technique, lexicon-based approach, and hybrid approach are the three main approaches employed by sentiment analysers. The lexicon-based method relies on a third-party dictionary or repository that assigns a polarity score to each phrase. The text's polarity would be the aggregate sum of the individual sentiments of the phrases that make it up. The machine learning approach entails utilising a labelled dataset and proper data vectorization to train a classifier model. The hybrid technique combines a machine learning classifier model trained on the dataset with an external lexical database that will be used as a foundation to label the datasets.

The objective of this project and the strategy to achieve it are stated below:

1.1 OBJECTIVE

To compare and contrast the sentiment and emotions of two corpora of COVID-19 tweets using both lexicon-based approach and deep learning-based approach.

Strategy

•To use Big Data and Data Mining concepts to collect a corpus of 265,000 tweets using Tweepy.

• To explore NLP concepts thoroughly and use the same to pre-process the tweets.

• Performing Sentiment and Emotion Analysis using existing tools.

• To research extensively on deep learning/machine learning and experiment with different frameworks and train a bi-directional LSTM-RNN model using TensorFlow to predict emotions of tweets.

• Delving into the topic of hyper parameter tuning and constantly experimenting with numerous parameters to improve the accuracy and the performance of the model.

• Producing numerous histograms, word clouds, etc using the matplotlib python package to effectively visualize the tweets.

1.2 Motivation

“Data is the new oil,” is a common statement touted by many leading scientists and eminent personalities for the past few years. But on closer inspection, one can argue that the power of data far exceeds that of oil, because it is an infinite commodity that will only increase in quantity. Massive amounts of data enable organizations to analyse trends and patterns, draw accurate and meaningful insights into customer preferences, and seamlessly predict the customer’s very next move. The vast range of applications of data science is, quite simply put, frightening, and is something that makes it an extremely exciting and enticing prospect.

Twitter, a social media and microblogging platform is a storehouse of information and data in today’s world. This is where the news first breaks and where millions of people come to voice their opinion on a variety of fields such as politics, sports, entertainment, etc. On average there are nearly 500 million tweets tweeted daily. This is a staggering number, one which makes Twitter an ideal source of data for our experiment.

Sentiment Analysis is a very popular tool deployed by many organizations to test the popularity of a product and gauge the customer’s preferences. It is used in a wide array of fields such as online shopping, movie reviews, etc. However, the sentiment is described as merely positive or negative which is rather vague. This project aims to delve deeper into the topic of sentiment analysis and classify a sentiment into specific emotions such as love, anger, fear, joy, etc. This would certainly help in providing a more nuanced understanding of the mood of the public towards a certain product/idea.

1.3 Background

There has been no dearth of cutting-edge research conducted on the fields of emotion detection, sentiment analysis and how machine learning and deep learning algorithms can help enhance these fields. Below, is a brief description of some of the research papers I have referred to, which has widened my knowledge on these topics.

Table 1: Literature Survey

Sl. No	Title	Journal/Conference	Brief Description
1.	M. A. Palomino and A. Padmanabhan Varma; "Any Publicity is Good Publicity: Positive, Negative and Neutral Tweets Can All Become Trends"	Proceedings of the 39th International Conference of the Chilean Computer Science Society (SCCC), 2020, pp. 1-8,doi:10.1109/SCCC51225 .2020.9281266.	Sentiment Analysis of tweets using lexicon based tools and aims to establish a corelation between the emergence of a trend and the polarity of a tweet.
2.	N. Colnerič and J. Demšar, "Emotion Recognition on Twitter: Comparative Study and Training a Unison Model"	Proceedings in IEEE Transactions on Affective Computing, vol. 11, no. 3, pp. 433-446, 1 July-Sept.2020,doi:10.1109/TAFFC.2018.2807817	Focuses on emotion detection of twitter data using deep learning models such as RNN's and compare it with lexical models.
3.	Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen; CARER: Contextualized Affect Representations for Emotion Recognition. .	In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.	An unsupervised feature extraction approach using graphs to extract emotion-relevant information was proposed. The word embeddings are added to the contextualised affect representations, which are then used to train numerous deep learning-based emotion identification models. The patterns capture both implicit and explicit language emotional information, improving emotion detection outcomes dramatically.
4.	Acheampong, Francisca & Chen, Wenyu & Nunoo-Mensah, Henry; Text-Based Emotion Detection: Advances, Challenges and Opportunities.	Wiley, July 2020, Engineering Reports. 2. 10.1002/eng2.12189.	This article talks about some of the most famous emotion models, and commonly used approaches to detect and predict emotions. Limitations, challenges and techniques to improve emotion detection are also discussed.

5.	Nandwani, Pansy & Verma, Rupali; A review on sentiment analysis and emotion detection from text.	Springer, December 2021, Social Network Analysis and Mining. 11. 10.1007/s13278-021-00776-6.	This article briefly delineates the history of Artificial intelligence and its role in the emerging fields of Natural Language Processing, Sentiment analysis and affective computing. Common emotion models and the process of emotion detection are also discussed.
6.	Kusal, S.; Patil, S.; Kotecha, K.; Aluvalu, R.; Varadarajan, V. AI Based Emotion Detection for Textual Big Data: Techniques and Contribution.	Big Data and Cognitive Computing. *2021; 5(3):43. https://doi.org/10.3390/bdcc5030043*	Goes into great detail on different emotion models, datasets, algorithms, and application domains of text-based emotion detection.
7.	Guo, Jia. "Deep learning approach to text analysis for human emotion detection from big data"	Journal of Intelligent Systems, vol. 31, no. 1, 2022, pp. 113-126. https://doi.org/10.1515/jisys-2022-0001	Deep learning based semantic text analysis using big data to detect emotions has been proposed. NLP techniques and word embedding are utilized to build a model with an impressive accuracy of 97.22%.
8.	Alla, K.R., Kandibanda, N., Katta, P., Muthavarapu, A., Kuchibhotla, S. (2022). Emotion Detection from Text Using LSTM.	Proceedings of Sixth International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems, vol 216. Springer, Singapore. https://doi.org/10.1007/978-981-16-1781-2_49	Uses LSTM to predict emotions on textual data. Pre-trained words from glove word embeddings are utilized.
9.	M. -H. Su, C. -H. Wu, K. -Y. Huang and Q. -B. Hong, "LSTM-based Text Emotion Recognition Using Semantic and Emotional Word Vectors	2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, 2018, pp. 1-6, doi: 10.1109/ACIIAsia.2018.8470378.*	LSTM in combination with autoencoders are used to predict emotions of textual data. Word2vec is used to get word embeddings which provide contextual meaning to the text data. Autoencoders help in dimensionality reduction of the word vectors and helps improve the efficiency.

10.	E. Batbaatar, M. Li and K. H. Ryu, "Semantic-Emotion Neural Network for Emotion Recognition From Text,"	*IEEE Access*, vol. 7, pp. 111866-111878, 2019, doi: 10.1109/ACCESS.2019.2934529.	A novel neural network architecture called Semantic-Emotion Neural Network is proposed which used both Bi-LSTM and CNNto detect emotions.

2 PROJECT DESCRIPTION AND GOALS

EXPERIMENTAL CORPORA

This section details the tweet collection and experimental corpora used in the project

Twitter Developer Access and TwitterAPI

To retrieve real time tweets from Twitter we had to create a Twitter account and gain developer access to the Twitter API. We were able to gain elevated access to the Twitter API. With elevated access the limit on retrieving tweets was two million tweets per month, which was more than sufficient for our project. Then we had to create an application which then with the help of some python code would enable us to interface with that application and retrieve real time tweets. Valid credentials like consumer keys and access tokens were provided upon successful registration to help establish a connection to the API.

Tweepy

Because of the significant support in libraries available, Python3 was utilised throughout the project. Tweepy is a simple and user-friendly python library for gaining access to the Twitter API. It features various classes and functions that make it simple to use.

The python code we used consists of an authentication handler class which is then instantiated with our valid consumer and access token keys and secrets. This will enable us to successfully establish a connection with the Twitter API to stream real time tweets. Each tweet streamed is in the form of a status object which holds a multitude of information regarding the tweet such as the full text of the tweet, date posted, user name of the person who posted the tweet and other relevant metadata. To listen to the tweets in real time we used the in-built Stream Listener class of the Tweepy library. The class methods however are overridden to only extract tweets to our requirements. Tweets were filtered based on the language (only English tweets were retrieved) and certain covid related hashtags. The full list of hashtags is provided down below. The filtered tweets were then printed out into a json file, with each line representing a tweet json object.

List of Hashtags used to filter the tweets

['#covid19','#longcovid','#coronavirus','#stayhome','#socialdistancing','#covid-19','#covid2019','#coronavirusoutbreak', '#sarscov2', '#virus', '#covidisnotover', '#covidvaccines','#vaccinated', '#longcovid', '#omicron', '#cases', '#covid', '#pandemic', '#coronaviruspandemic', '#mask', '#deltacron', '#covidiots']

Loading JSON File

We execute a python script which reads each line of the file as a json object. Relevant metadata such as Tweet Text, Tweet ID, etc are extracted. The list of all the metadata extracted is provided below. With the help of the pandas library a dataframe is created and all the twitter data is stored here. It is also saved as a csv file. This ensures that the data can be processed with ease at a later time.

Explanation of Corpora used in Project

Palomino and Varma [1] have attempted to gauge the corelation between the emergence of a trend and the overall sentiment it denotes. For the purpose of this experiment, they collected an enormous collection of tweets related to the COVID-19 pandemic. They gathered a corpus of 409,761 tweets on 22nd April,2020. The relevance of that date they argue, was when the UK Foreign Secretary, Dominic Raab, conducted a press conference to address the UK Government’s response to the situation.
Nearly two years have passed and it is not unreasonable to assume that the public sentiment towards the covid pandemic has changed. This forms the basis of our project. We aim to compare and contrast the sentiment of the tweets collected two years ago and as of now and see if there is any change. An emotional analysis, one where sentiments where further classified into specific emotions such as love, fear, joy, anger, happiness, etc was also performed to help us gauge the mood of the public sentiment towards the COVID-19 pandemic more comprehensively.

Corpus 1

This corpus consists entirely of the tweets collected by Palomino and Varma [1]. The tweets collected by them were all related to the COVID-19 pandemic. COVID-19 is an infectious virus strain that adversely impacts respiratory system. It originated in Wuhan, China in late 2019 and by March-April 2022 it had spread all across the globe, with most countries imposing strict restrictions such as social distancing and lockdowns to mitigate the spread of this deadly virus. The tweets were scheduled to be retrieved on 22nd April 2020 from 15:30 to 19:30 (GMT). It was during this time interval that the scheduled press conference highlighting the UK governments measures to mitigate the deadly virus was held. Therefore, anticipating a multitude of tweets by netizens, sharing their opinions and feedbacks, the tweets were collected during this time.

Table 2: Hashtag Distribution in Corpus 1

No.	Hashtag	No. of Tweets
1.	#covid19	238,432
2.	#coronavirus	116,557
3.	#stayhome	31,820
4.	#covid_19	11,068
5.	#socialdistancing	6510
6.	#covid-19	4636
7.	#covid2019	2341
8.	#flattenthecurve	2124
9.	#coronavirusoutbreak	2058
10.	#sarscov2	1861
11.	#virus	1211

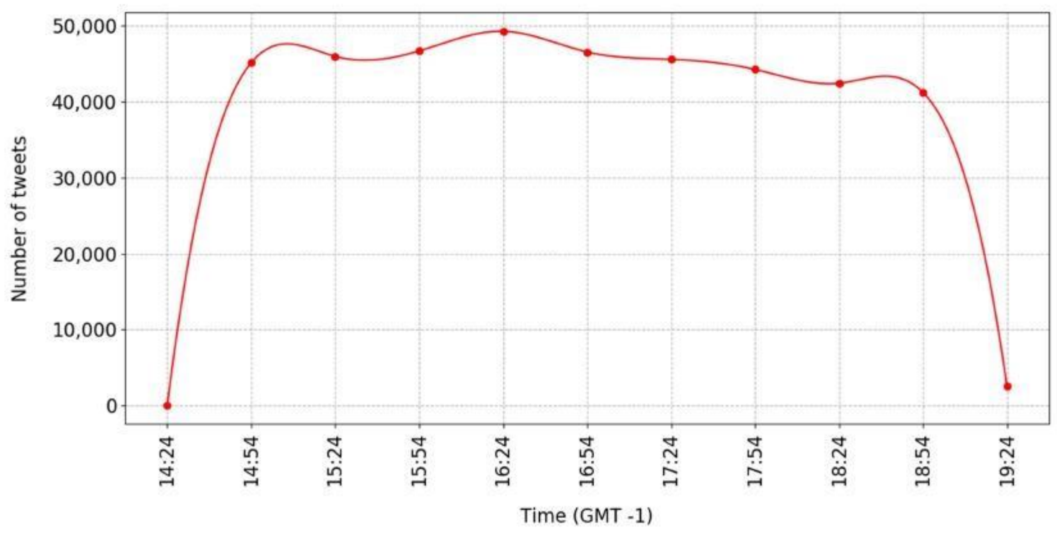


Figure 1: Tweet Distribution over time

The tweets were collected based on the hashtags shown in Table 2. The first tweet was collected at 14:24:39, and the last one was collected at 18:56:27 (GMT - 1), spanning a total duration of 4 hours and 30 minutes as shown in Figure 1. A total of 409761 tweets were collected. The numbers in Table 2 do not add up to the total number of tweets in the corpus as tweet could contain multiple hashtags.

Corpus 2

This entire corpus consists of the tweets collected by us. It was collected using the procedure explained above. The tweet collection commenced on the 24th of March, 2022 at 23:30 GMT and stopped on the 29th of March, 2022 at 00:30 GMT. Factoring in the forwarding of the clock due to daylight savings time during the collection, the total duration of tweet collection was ninety-six hours. In total we were able to collect 265,108 tweets. The tweets were retrieved according to hashtags mentioned below in Table 3. The numbers in Table 3 do not add up to the total number of tweets in the corpus as tweet could contain multiple hashtags.

Table 3: Hashtag Distribution in Corpus 2

No.	Hashtag	No. of Tweets
1.	#covid	182,686
2.	#covid19	125,434
3.	#longcovid	34,565
4.	#covidisnotover	25,010
5.	#omicron	15,608
6.	#coronavirus	9487
7.	#covid-19	8443
8.	#pandemic	8429
9.	#mask	6627
10.	#sarscov2	2950
11.	#stayhome	2118
12.	#virus	2032
13.	#vaccinated	1320
14.	#covidiots	1038
15.	#socialdistancing	502
16.	#deltacron	283
17.	#cases	273
18.	#covidvaccines	206
19.	#covid2019	196
20.	#coronaviruspandemic	188
21.	#coronavirusoutbreak	153

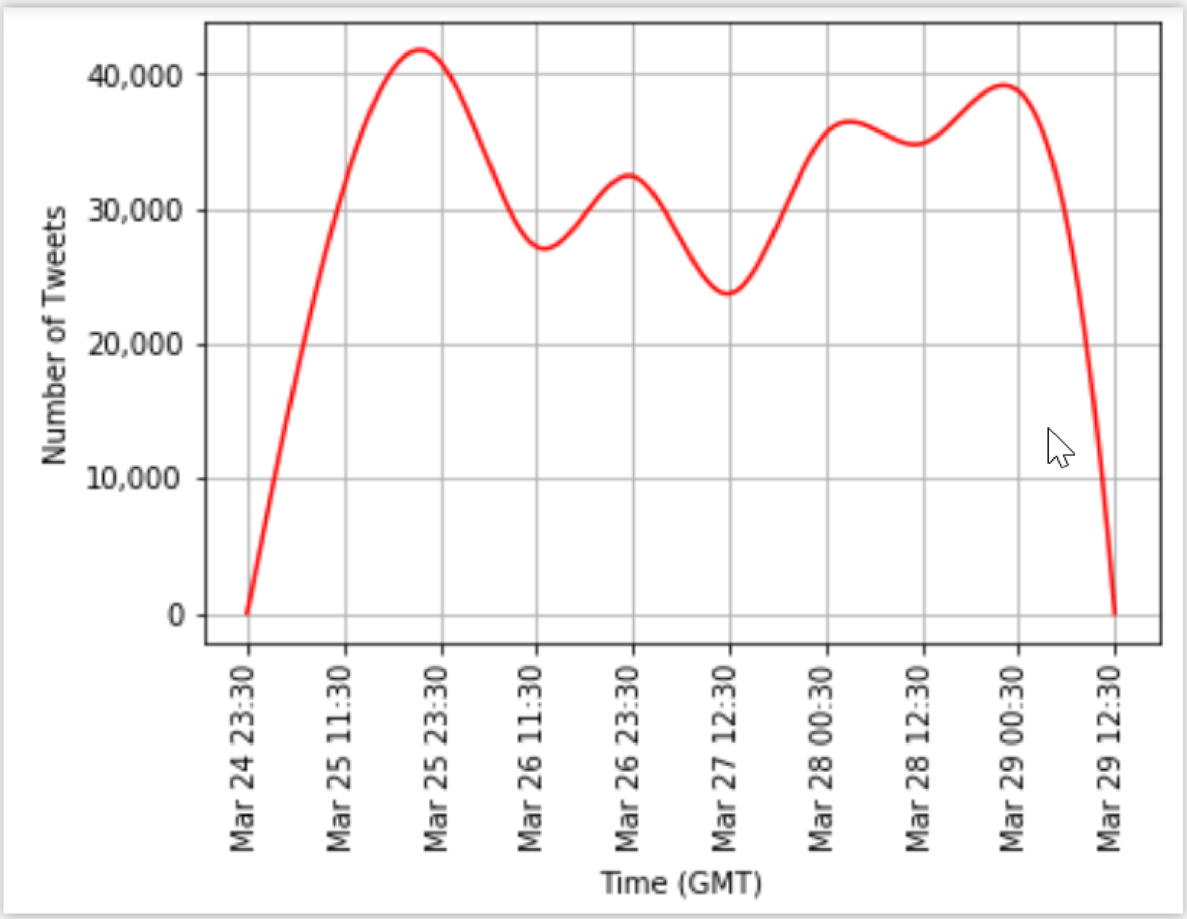


Figure 2: Tweet Distribution over time

Sentiment Analysis Tools

When it comes to tools that may be used to calculate a text's sentiment score, there are several options. This decision is crucial since the entire research is skewed in this direction because to the basic differences in the algorithms' foundations. Due to the lack of agreement and consensus in sentiment analysis tools [7] we experiment with two such tools VADER and TextBlob to compare and contrast the results.

VADER

VADER, or Valence Aware Dictionary and sEntiment Reasoner, is a social media lexical and rule-based evaluator. It's a valence-based tool that tells you how negative, positive,
or neutral a piece of text is. It also gives you a compound score, which is the normalised score of the individual positive, negative and neutral score. It combines lexical and rule-based techniques, emphasising a text's sentiment score in the presence of capitalization, punctuation (such as many exclamations), and the usage of transitional words like "but," to mention a few.

It's easy to use, computationally efficient, and versatile enough to work with both polarity and valence. Another reason for its wide spread popularity is its open-source feature, which permits anyone to modify the rules it follows for sentiment scoring.

TextBlob

Textblob is a python package for processing textual data that is open-source. It conducts a variety of operations on textual data, including sentiment analysis, categorization, and translation, among other things. It produces a polarity score for sentiment analysis that ranges from -1 to 1, with the extremes denoting negative and positive feelings, respectively.

The area of focus in sentiment analysis is polarity, but in emotion detection, the emotional or psychological state or mood is paramount. Emotion identification is much more comprehensive and objective than sentiment analysis, which is very subjective.

Emotion Detection:

Since the advent of Artificial Intelligence in 1950, it has played an immense role in providing solutions to numerous human problems. Natural Language Processing (NLP) is one such field which aims to aims to make machines understand and comprehend human languages. Some of the most commonly researched NLP fields are Sentiment Analysis, Language Translation systems, text summarization systems, text auto complete systems. Emotion detection is a sub-field of Sentiment Analysis which aims to classify specific emotions such as happy, sad, angry, etc instead of attributing overtly general polarities such as positive and neutral.

Emotions play a crucial role in any person’s life. It helps convey their thoughts and feelings to others effectively. Therefore, in this extremely interconnected and technologically advanced social media world, it helps massive corporations and businesses understand the moods of the public towards a certain product. This therefore helps immeasurably in their promotion and marketing strategies ensuring growth and increased revenue.

The bedrock of Emotion Detection (ED) systems are emotion models, which determine how emotions are expressed. The models presume that emotions exist in different states, justifying the need to differentiate between them. When engaging in any ED-related task, it is essential to initially determine the emotional model to be used. Research on emotion detection on text pales in comparison to emotion detection, on facial expressions, body language, etc.

Discrete emotion models (DEMs): A discrete model of emotions includes categorising or classifying emotions distinctly.

Among the most notable are:

The Paul Ekman model groups emotions into six types. According to his theory, there are six basic emotions that arise from discrete neurological systems as a result of how a person interprets a situation, and hence emotions are independent. Happiness, sadness, anger, disgust, surprise, and fear are the basic emotions. However, the combination of these feelings can result in more complicated emotions like guilt, shame, pride, desire, and greed, etc.

The Robert Plutchik model, much like Ekman, proposes that there are just a few primary emotions that occur in opposite pairings and combine to form complicated emotions. Supplementing Ekman's six core emotions, he named two more, including trust and anticipation, making a total of eight essential emotions. Joy vs. sadness, trust vs. disgust, anger vs. fear, and surprise vs. anticipation are the eight opposing emotions. As per Plutchik, there are differing levels of intensities for each emotion determined by how an event is interpreted by an individual.

Profile of Mood States or POMS is a standard validated psychological test formulated by McNair et al [18] to assess the psychological state or mood of a person. It classifies emotions into seven categories; anger, confusion, depression, fatigue, friendliness, tension and vigour.

Building on Colnerič and Demšar[5] work we use their pre-built models to classify the emotions on our tweet collections using Ekman, Pluthick and POMS emotion models. Furthermore, to supplement this we decided to use Text2Emotion, which is lexicon based python library to attain emotions of tweets. The five emotions that are assessed in Text2Emotion are Happy, Angry, Sad, Surprise and Fear.

3 TECHNICAL SPECIFICATIONS

3.1 Hardware Requirements:

System: intel i7

Hard Disk: 1 TB

Monitor: 15.6’’ LCD

Ram: 8GB

3.2 Software Requirements:

Operating System: Windows 10

Coding Language: Python

Coding Environment: Google Collaboratory

4 DESIGN APPROACH AND DETAILS

.

4.1 TEXT PRE-PROCESSING

This chapter describes the pre-processing pipeline that was implemented to pre-process the tweets and remove the ‘noise’ and other unimportant matter so that the crux of the tweet text can be retained for further data analysis.

Tweet structure

Because Twitter is a microblogging service, the amount of characters that may be contained in a single tweet is limited. The initial restriction was 140 characters; however, it was increased to 280 characters in 2017. Other than text, a range of other elements can be included within the predetermined limit.

Emoticons

Emoticons are one of the most often used methods of expressing emotions in tweets. They have been employed in virtually every tweet to convey emotions, thanks to the large number of options present. Emoticons are Unicode characters when obtained in text form, which may require translations to a secondary encoding to be saved because not all platforms accept Unicode characters.

Media

Media is also popularly used in tweets, just behind the usage of emoticons. When paired with the tweet content, they provide a richer context for the user. Any Twitter-compatible image, video, or GIF qualifies as media. The address relating to the media on Twitter appears as a URL when a tweet with media is gathered.

Hashtag

Hashtags are terms that are prefixed by the '#' character, as the name clearly indicates. It was first used on Twitter in 2007, and it quickly grew in prominence as the preferred technique to group/cluster posts or conversations about a specific topic. Since then, hashtags have become extensively used on Twitter and other social media platforms. Additionally, hashtags are used to generate a list of "trending themes" or the most popular hashtags.

URLs

A web address is referred to as a URL (Uniform Resource Locator). Every website or piece of media has a unique URL that may be used to go to it at any time. They have a specific format that begins with 'http://' or 'https://' and ends with their unique IDs. Because of the word constraint on Twitter, people frequently choose to send readers to a specific website or piece of media like YouTube, which helps to gain a more comprehensive understanding of their argument.

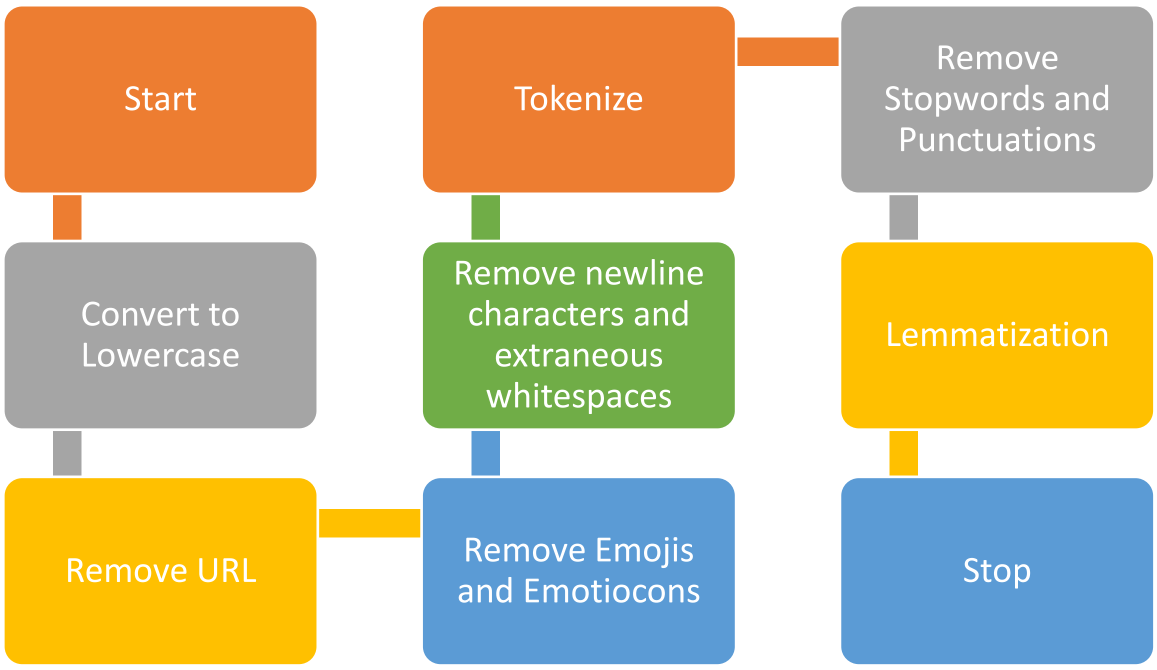

Start
Tokenize
Remove Stopwords and Punctuations
Convert to Lowercase
Remove newline characters and extraneous whitespaces
Lemmatization
Remove URL
Remove Emojis and Emoticons
Stop

Figure 3: Pre-Processing Block

Pre-Processing Techniques

Pre-processing is a technique for cleaning text input before it is fed into analytic algorithms. A lack of pre-processing often leads to more time-consuming calculations and erroneous results. Because the data and needs for each application differ, there are no hard and fast rules for pre-processing.

General Clean-Up

General clean-up describes a set of typical text preparation techniques, such as converting all text to lowercase, deleting extraneous whitespace ('\n' and '\r'), and eliminating media links and emoticons. URLs are deleted with the aid of regular expressions since they follow a definitive format which is easy to exploit. Emoticons aren't particularly useful in a text-only algorithm. The simplest technique to remove emoticons from text was to use their representation as a Unicode character and exploit that.

Tokenisation

Tokenization is a method of separating text into individual words and punctuation marks. This can be accomplished through a variety of methods. The Tweet Tokeniser function in NLTK (Natural Language Tool Kit) was utilised for the project as it was a standardized suite for Natural Language Processing (NLP)Tokenization was performed as an initial step to process the content due to the relatively complicated structure of a tweet.

Stop-words and Punctuations

Stop words include terms like this, is, at which, and on, which are incredibly frequent and semantically non-selective. These phrases do not influence the overall sentiment of the tweet and place a great strain on computational power while processing if not removed. By deleting these phrases, greater emphasis is placed on terms that add more to the meaning/context of a tweet. Punctuations also add very little weightage to a tweet's plain textual content and must be eliminated.

For 16 languages, NLTK maintains an extensive list of stopwords. They are regularly updated to reflect the most recent findings. The list of English stopwords may be swiftly compiled and matched to the tokens in tweets. Similarly, the Python String module keeps a list of all punctuations which can be easily exploited to delete them in the tweets. Individual tokens from the tweet may be compared to both of these lists to confirm that all stop words and punctuation have been deleted. However, we used the stop-word list built by Salton and Buckley for the experimental SMART information retrieval system [2] which contains 571 words and is more comprehensive than the NLTK stopword list.

Lemmatization

It is the process of reducing inflectional and derivational related forms of a word to a common base [3]. Despite stemming being computationally faster that lemmatization, we opted for the use of lemmatization based on the recent results reported by Haynes, et al. [4]. Both stemming and lemmatization aim to reduce a word's inflectional forms and occasionally associated derivational forms to a basic form. The flavours of the two terms vary, though. Stemming is a term that often describes a rudimentary heuristic method that removes derivational affixes from words in the hopes of attaining this aim most of the time. Lemmatization often refers to carrying out tasks correctly using a vocabulary and morphological analysis of words with the goal of removing only inflectional ends and returning the lemma, or dictionary form, of a word[19].

TF-IDF

The scientific metric TF-IDF, or Term Frequency and Inverse Document Frequency, is used to detect and rate the importance of a term in a document or dataset. The TF-IDF value is obtained by multiplying both terms.

Term Frequency

In this case each document refers to each tweet. The Term Frequency (TF) simply calculates the frequency of each term/word in a document. The method in which this is calculated is by dividing the the number of times each term occurs in a document by the total number of terms in that document.

Inverse Document Frequency

The relevance of each phrase in a particular document is determined by looking at the inverse document frequency. The logarithmic ratio of the entire number of documents/tweets in the collection to the total number of documents where that specific tweet appears is used to compute it. On multiplying both the scores, we arrive at the final TF-IDF score. This is a much more holistic method to gauge the importance of each term, rather than just simply calculating the term frequency as it in away give semantic meaning to the text and penalizes terms which occur to frequently in each document but not influence the entire dataset in a huge manner.

WordCloud

A WordCloud is an extremely powerful tool to visually represent data. At a quick glance viewers will be able to understand the most frequently occurring words as they appear in a larger font and are more prominent. This helps to quickly draw insights from a large amount of data.

Thus, we use wordclouds to better visualize our data. A python script is implemented for the same. The words visible on the wordcloud are ranked based on their TF-IDF score. Therefore, the word with a higher TF-IDF score appears more prominently.

4.2 Deep Learning Bi-Directional RNN-LSTM

The results after using the pre-trained Ekman and Plutchik model [5] are quite suspicious, as feelings of joy and trust are some of the dominant emotions, which seems highly unlikely. Therefore, in order to address this, we have trained our very own Bi-Directional RNN-LSTM to predict emotions.

Tools used:

TensorFlow: It is machine learning framework created by google. TensorFlow is a free and open-source dataflow and differentiable computing toolkit that may be used for a variety of purposes. It's a symbolic math package that's also employed in machine learning applications like neural networks.

Keras: It is nothing but an API interface over TensorFlow which makes deep learning programming easier. Keras is a Python-based open-source neural network API. It may be used in conjunction with TensorFlow, Microsoft Cognitive Toolkit, R, Theano, etc. The reason Keras is so widely used is its user friendliness. It is modular, and expandable, with the goal of allowing quick testing with deep neural networks.

Keras is built on a simple framework that makes it simple to build deep learning models using TensorFlow or Theano. It is a deep learning framework that allows you to easily define models and utilise the advantages of deep learning computing, without having to solely rely on TensorFlow, which on its own can be very difficult to program and interface with.

Pandas:

Pandas is an extremely popular open source Python library for data science, data analytics, and machine learning activities. It used to store and manipulate data in an efficient manner. Using a data structure called DataFrame in Pandas, we can store data, do common pre-processing tasks such as cleaning the data, filtering the data,etc. The Pandas library has numerous inbuilt methods for the same.

NumPy: It is short for Numerical Python. It is a library that includes a collection of array processing routines and multidimensional array objects. Using NumPy, we can perform a multitude of mathematical and logical operations on arrays. It makes common mathematical operations such as vector addition, multiplication simple and much more computationally efficient as compared to a for loop. As deep learning tasks generally involve colossal amounts of data it is widely used in this field.

Matplotlib:

Matplotlib is a graphing package for Python with NumPy. For integrating charts into programmes using GUI toolkits like Tkinter, wxPython, etc., it provides an object-oriented API.

A Matplotlib programme called Pyplot imitates MATLAB features. In addition to being completely free, Matplotlib aims to be just as user-friendly as MATLAB. A plot is altered in some manner by each PyPlot method, such as by adding a figure, labels to the axes, legends, plot colours, and so on.

NLP:

Natural language processing (NLP) one could say is at the cross-roads of linguistics, computer science, and artificial intelligence. It is primarily focused on human computer interaction and how can machines can be designed to manipulate and understand large amounts of text. And their underlying context. It is very popularly used in chatbots, and problems such as sentiment analysis, text autocomplete suggestions and language translations.

The NLTK (Natural Language Toolkit) Python Library consists of a numerous NLP tools and methods. It is used in NLP tasks such as Tokenization, Removing punctuations and stopwords and Stemming and Lemmatization.

Google Colboratory:

The RNN model was created with Google Collab, a cloud-based Jupyter notebook environment. The main purpose of using the Google Collab environment was the free access to GPU's. This provides better results when working with computationally intensive deep learning models as compared to a regular CPU.

Deep Learning:

It is a subset of Machine Learning that mimics the human brain and its neurons and neural networks to learn and gain knowledge. The performance of machine learning model plateaus, when dealing with problems which have a high dimensionality of data. The more powerful deep neural network model is thus preferred for computationally heavy tasks such as image recognition and natural language processing.

RNN

Regular artificial neural networks do not depend on the sequence or order of input. For example, in an artificial neural network to detect if a credit card transaction is fraudulent or not, the output does not depend on the order of the input parameters of the neural network. However, for natural language processing tasks such as sentiment analysis, this is not the case. All NLP tasks are sequence modelling problems, where the order of the words play an important role in determining the meaning and context of a sentence, and thus the sentiment. RNN is specially designed for sequential data. A RNN is a looped neural network that gains knowledge from the preceding stage. The data is conveyed from one unit to the next through its output. Hence, the RNN unlike artificial neural networks allow, the result of the output of the previous state/word to influence the current input and output.

LSTM

A common problem faced with RNN's is that it has an extremely short-term memory. Thus, words encountered at the beginning of a sentence are quickly forgotten and have little influence on the later and final outputs of the neural networks. This is due to the vanishing gradient problem. After the forward pass of the neural network, the cost and then the total loss is calculated. During the backpropagation step, the loss is differentiated with respect to the weights of the input parameters so that the weights can be adjusted accordingly and the model will learn and minimize the loss value, maximizing accuracy. In a deep neural network however when adjusting the weights of the inputs of the first layer, the value of the gradient is extremely small, due to the constant multiplication of minute numbers. Hence, the change of their values are negligible, and soon they are forgotten by the RNN.

To combat this LSTM neural network is used, which stores certain words in its memory. Long-range dependencies can be captured via LSTM. Therefore, it has the ability to remember prior inputs for long periods of time. An LSTM cell consists of three gates, Input, Output and Forget gate.

• Input Gate: The input gate adds important data/words to the cell state.

• Output Gate: The output gate adds further data/words considered important to the cell state.

• Forget Gate: The forget gate eliminates data from the cell state that is no longer meaningful/ helpful.

These gates are used in LSTM to manipulate memory. Gates in long-short-term memory regulate the gradient propagation in a recurrent network's memory (LSTM). The network was able to learn the circumstances for when to discard, disregard, or keep data in the memory cell thanks to the LSTM's gating mechanism. The beauty of the LSTM model is that it automatically learns which words are extremely influential in determining the emotion and stores them in memory, while discarding the rest.

LSTM uses the hidden layer to maintain data from inputs that have already gone through it. As it has only been exposed to the previous inputs, the unidirectional LSTM only saves data from the past. Bidirectional LSTM on the other hand, will run the inputs from both directions i.e., from past to future, then from future to past. Because of this, the LSTM that run backwards maintains the data from the future as well. Simply put, because of this words that appear later on will also have an impact/influence on the words that appear before it.

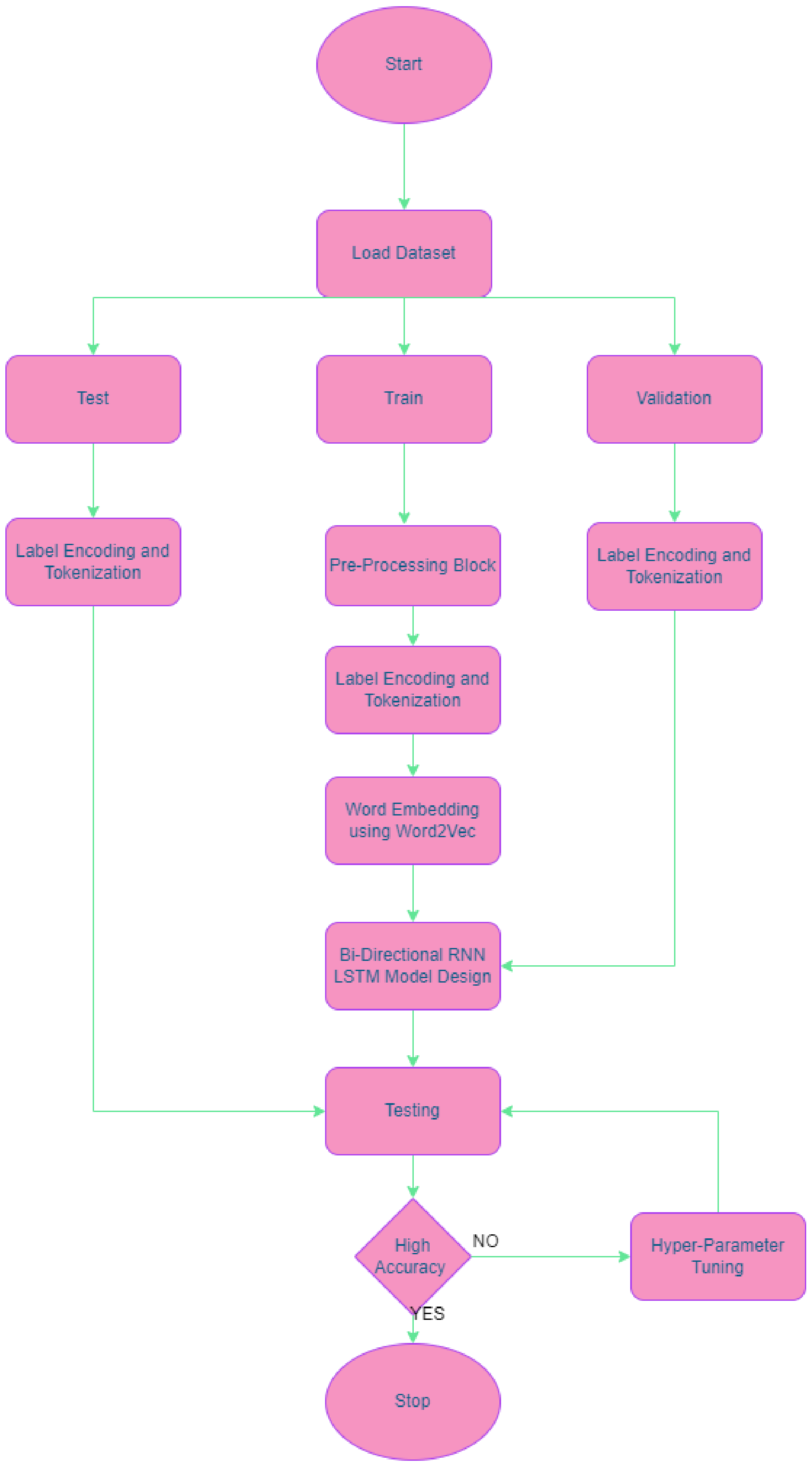

Start
Load Dataset
Test
Train
Validation
Label Encoding and Tokenization
Pre-Processing Block
Label Encoding and Tokenization
Label Encoding and Tokenization
Word Embedding using Word2Vec
Bi-Directional RNN LSTM Model Design
Testing
High Accuracy
NO
YES
Hyper-Parameter Tuning
Stop

Figure4: High Level Model Architecture

Dataset:

The dataset used to train our model was taken from the work published by Saravia, Liu et al [6]. It contains 3 files, namely the training set, test set and validation set. They consist of 16,000 ,2000 and 2000 columns respectively. The six emotions classified are sadness, anger, love, fear, surprise and joy. This dataset was chosen as this was as close to the emotions as that of Ekman and Plutchik. As you can see in the histogram below, the training set data is not balanced as joy and sadness are the most dominant emotions, whereas surprise and love are extremely supressed.

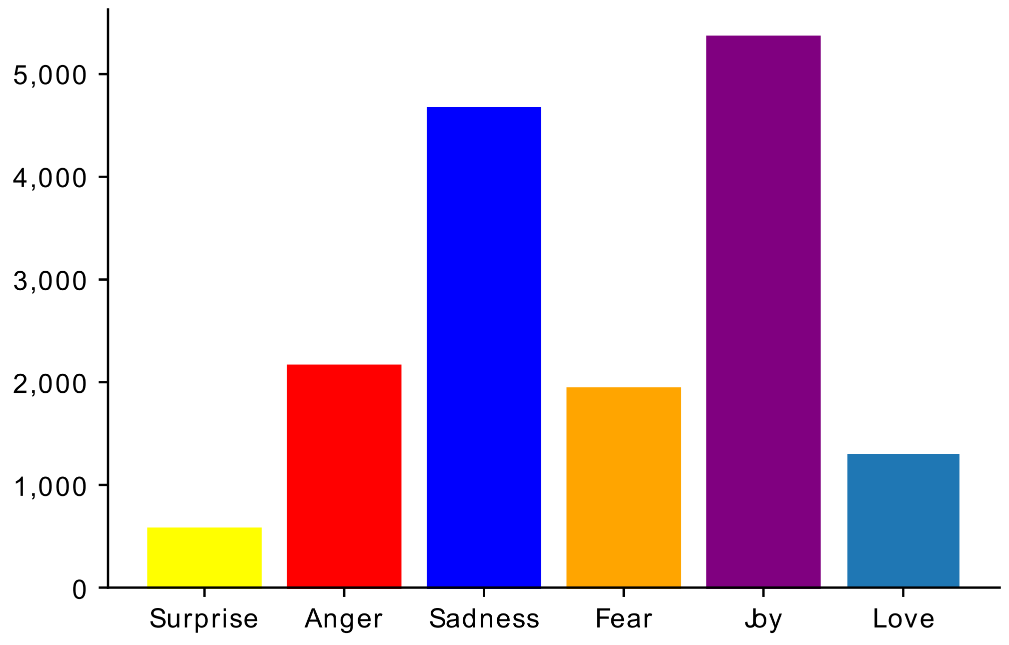


Figure 5: Tweet Distribution by Emotion in Training Set

Text Pre-Processing:

The tweets were pre-processed similar to the procedure mentioned in section 4.1.

Label Encoding:

As our deep learning model will not be able to able to understand text, we will need to convert the emotion values into numeric values. Using a dictionary, we hardcode the emotion values to {‘joy’:0,’anger’:1,’love’:2,’sadness’:3,’fear’:4,’surprise’:5}

Tokenization:

Similarly, the tweet text has to be also converted into numerical values before being passed into the model. The tokenizer class converts each tweet into an array of numbers. First, a dictionary is produced which assigns a number to each word based on the frequency of the words. There is a parameter named ‘num_words’ that can be specified. In this case it has been specified to 10000. Thus, only the top 10000 words will be taken into account i.e., be part of the tokenizer vocabulary. Both the test and train dataset were used to augment the vocabulary of the tokenizer. Then the text_to_sequence method converts each tweet into a sequence of numbers based on the dictionary.

However, the process, does not stop here. As each tweet is of differing length, the length of each tokenized sequence will invariably vary. Hence, in order to pass them into our model, we must ensure that they are all of the exact same length. For our model we chose a maximum length of 300. Therefore, each tokenized sequence will get padded with zeroes at the end so that the length is brought up to 300. If, the maximum length is greater than 300, it will automatically trim from the end to ensure the tokenized sequence length remains at 300. This process is performed on the train, test and validation sets.

Word2Vec:

It is a technique which uses neural networks to train on large datasets and figure out word embeddings as a result. The word embeddings are nothing but numeric vectors based on arbitrary features. This allows for word associations, and similar words can be grouped based on their similarity in meaning. Therefore, by using Word2Vec we can establish relationships between words that are similar in meaning, which otherwise would have been treated separately by our tokenizer.

We employ the glove-wiki-gigaword-100 model, which was developed using Wikipedia data and transforms a word into an array of 100 elements. We utilize the glove vector by using the genism library. Now using the downloaded glove model, we can map the vocabulary learnt by the tokenizer into the word vector of length 100, if the word is indeed present in the glove genism vocabulary.

Model Design:

After all the processing tasks are complete, it is time to create our Bi-directional RNN-LSTM model and train it. We create a sequential model using keras. The first layer of the neural network architecture is the input layer or in our case the embedding layer. As mentioned above, we have already created a word embedding matrix with the help of 10,000 unique words with a embedding(vector) dimension of 100 and sequence length of 300. Then we add three hidden layers in the neural network architecture. This increases the computation and in turn enhances the accuracy of the model. However, in order to avoid overfitting the model, we add a dropout regulazation to each hidden layer. This inactivates twenty percent of all the neurons randomly in each hidden layer. The three hidden layers are bi-directional CuDNNLSTM layers. These can only be run on gpu and is much faster to implement than the regular LSTM Layer. By trial and error, we set the number of neurons in the 3 hidden layers to 100,200 and 100 respectively and return sequence is set to true, which implies that if an output is transmitted to another bidirectional LSTM layer, it is delivered as a sequence rather than a single value of each input so that the next LSTM layer may receive the appropriate input. The default activation function for the CuDNNLSTM layers is tanh which ouputs a value from -1 to 1.

The final layer of the neural network is a Dense layer with six neurons present, for each of the six emotions that have to be classified. The activaton function for the final layer is softmax which returns the probability scores of how likely it is to belong to each of the target classes.

The loss is set to 'sparse categorical crossentropy' since it is used for multi-class classification issues when the classes aren't one-hot encoded (for binary classes). 'adam' was chosen as the optimizer since it is extremely efficient when working with huge datasets. The accuracy training metric is used to calculate how often the predictions match the real labels.

To efficiently train our models, we implemented certain callbacks. EarlyStopping ensures that the training is put on hold, if there is no further improvement in the models accuracy score or if the training loss decreases or validation loss increases beyond a certain threshold. ModelChckpoint saves the model checkpoint.

Training the model:

We then proceed to train the model and call the fit method on the model and pass the processed training inputs and outputs. The number of epochs (passes through the dataset) are set to 25 and the validation data is also passed to validate the training process.

Testing the model:

Now comes the most important part, which is testing. This is vital to gauge how the model performs in the real world on untrained data, which it has not seen previously. We then proceed to pass the test data to the model to predict the emotions and then compare to the actual values. Thus we can get a sense of the accuracy of the model. To better visualize this, we plot a confusion matrix and print the classification report, which has a comprehensive summary of all the metric scores.

5 Legal, Social, Ethical and Professional Issues

Legal

The potential legal issues concerning this project are primarily related to the use of the twitter data that we collected and the privacy and confidentiality of the users. While creating a Twitter Developer account to gain authorised access to the Twitter API to stream real time tweets, we had to sign a service agreement to abide by all of Twitter's rules and regulations [15]. Therefore, all the twitter data that we had collected for the purpose of this project is legal. Furthermore, we will not publicly publish sensitive personal information such as user name, user id, location, etc.

Social

Twitter users do not represent the entirety of human population, nor are they indicative of Internet users. Twitter data is not reflective of users on twitter as well. This is due to the fact that not every Twitter user will tweet about a particular topic. A small subsection of very active twitter handles account for the lion's share of tweets in any dataset [16]. Spam baiting, where twitter handles utilize popular hashtags to promote their content and improve their chances of trending are also extremely prevalent on twitter. There are also umpteen number of bots or fake twitter handles created to increase a person's popularity and following. Therefore, not all twitter data are reliable and their users genuine. These are some of the social problems plaguing the credibility of twitter data.

Ethical

Due to the sheer volume of tweets retrieved it is impossible to obtain the consent of each individual user, whose tweet has been used for the project. The Twitter API Terms of Service prevent the sharing of datasets; however, researchers can share the tweet identification numbers, which can be used by other researchers to get Twitter datasets. If providing tweet IDs is not allowed for any reason, researchers may be able to access a comparable dataset by sharing keywords and data retrieval time [15].

Professional

Any software project which contains code can have an adverse impact on its users if not approached in a professional manner. While developing this project we have followed the BCS code of conduct which ensures due regard for public health, privacy, security and wellbeing of others and the environment [17].

6 SCHEDULE, TASKS AND MILESTONES

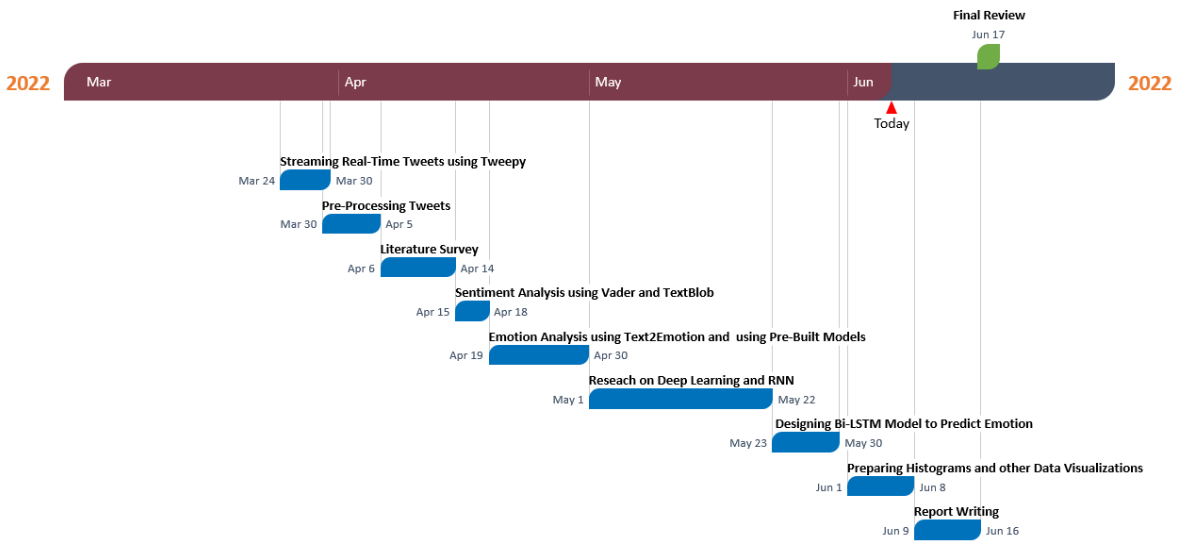


Figure 6: Gantt Chart

Table4: Project Schedule

Task/Milestone	Start Date	End Date	Duration (Days)
Streaming Real-Time Tweets using Tweepy	**24-03-2022**	**30-03-2022**	**6**
Pre-Processing Tweets	**30-03-2022**	**05-04-2022**	**6**
Literature Survey	**06-04-2022**	**14-04-2022**	**8**
Sentiment Analysis using Vader and TextBlob	**15-04-2022**	**18-04-2022**	**3**
Emotion Analysis using Text2Emotion and using Pre-Built Models	**19-04-2022**	**30-04-2022**	**11**
Reseach on Deep Learning and RNN	**01-05-2022**	**22-05-2022**	**21**
Designing Bi-LSTM Model to Predict Emotion	**23-05-2022**	**30-05-2022**	**7**
Preparing Histograms and other Data Visualizations	**01-06-2022**	**08-06-2022**	**7**
Report Writing	**09-06-2022**	**16-06-2022**	**7**
Final Review	**09-07-2022**		**1**

7 RESULTS & DISCUSSION

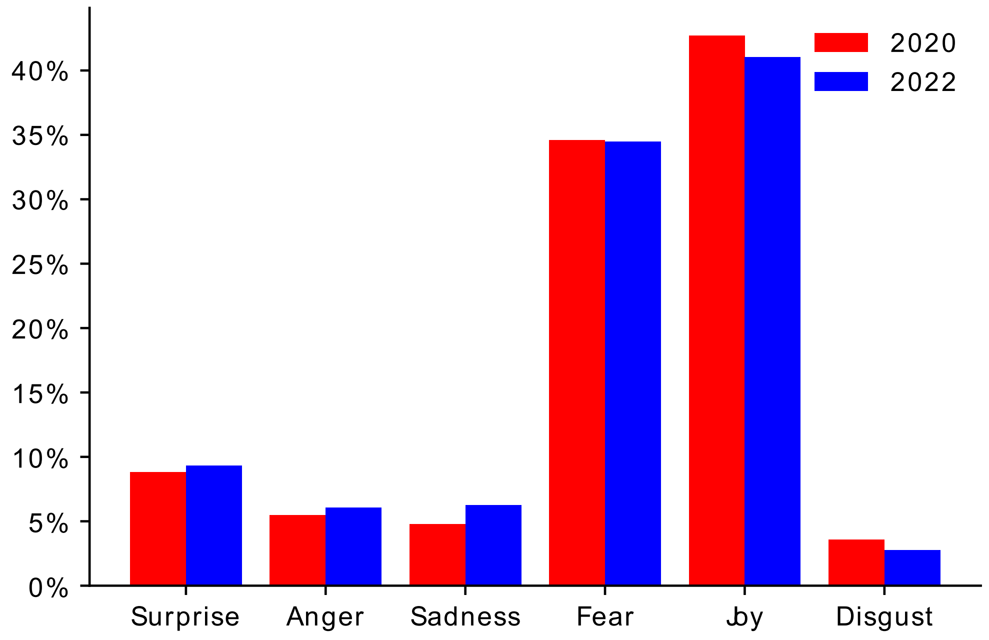


Figure 7: Differences in emotions from 2020 and 2022. The histograms display the cumulative distribution of the probabilities of each of Ekman's emotions.

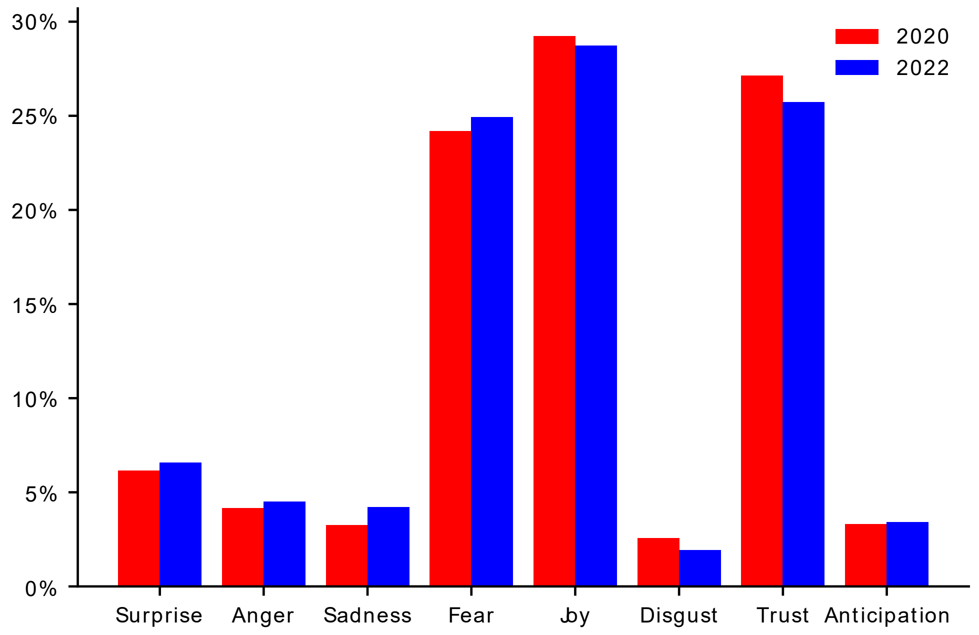


Figure 8: Differences in emotions from 2020 and 2022. The histograms display the cumulative distribution of the probabilities of each of Plutchik's emotions.

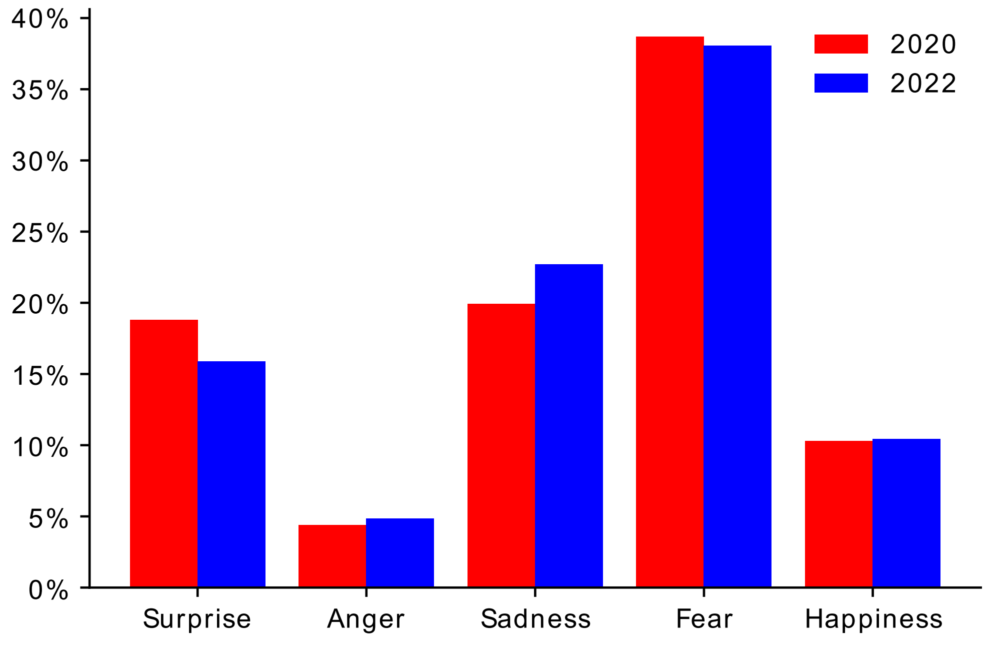


Figure 9: Differences in emotions from 2020 and 2022. The histograms display the cumulative distribution of the probabilities of each of the emotions in Text2Emotion.

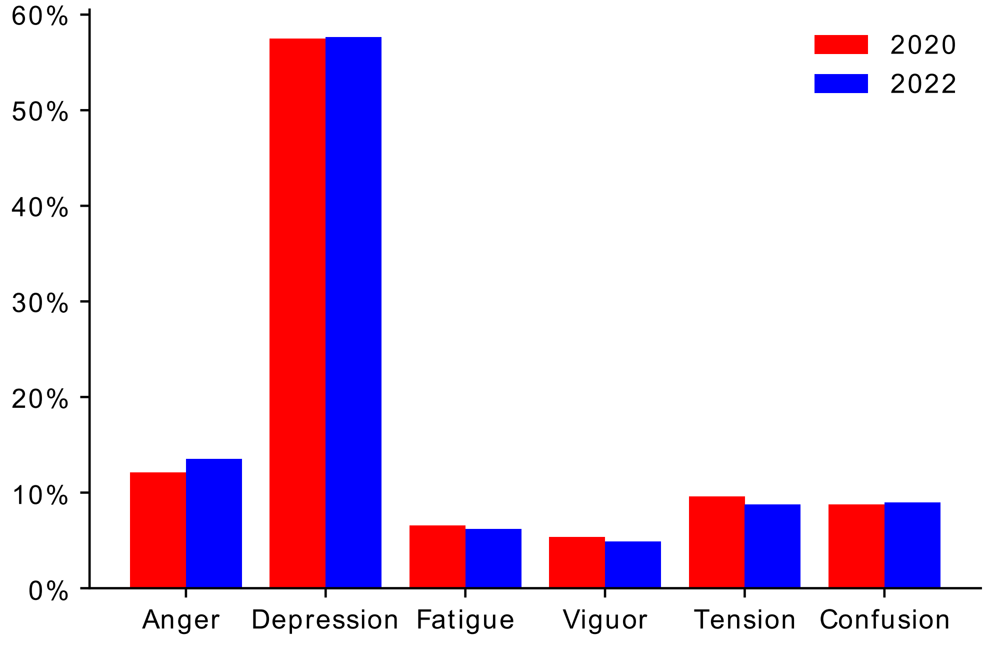


Figure 10: Difference in emotions from 2020 and 2022. The histograms display the cumulative distribution of the probabilities of each of the POMS emotions.

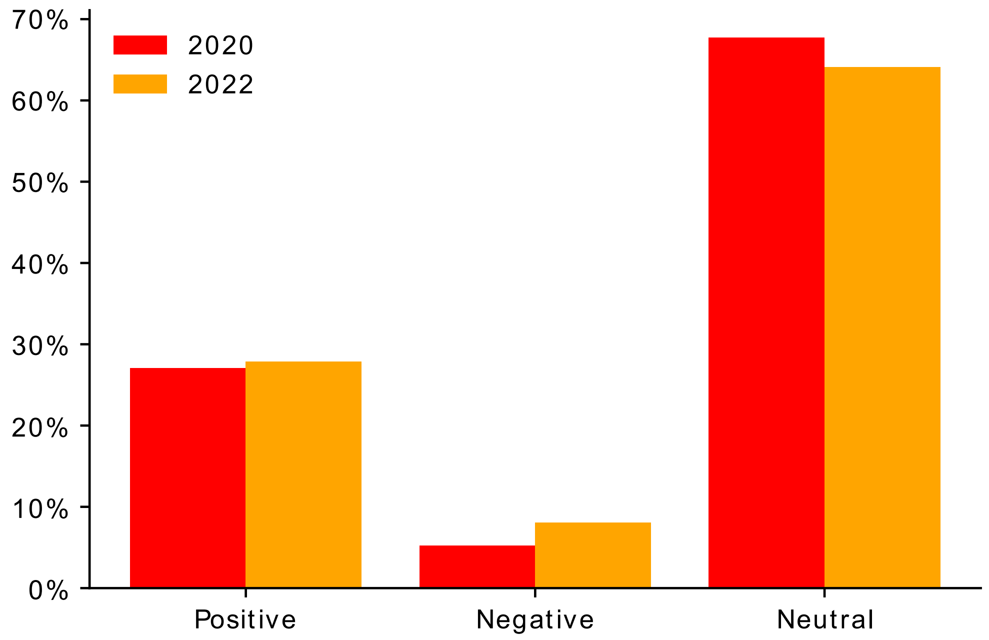


Figure 11: Difference in sentiment from 2020 and 2022. The histograms display the frequency distribution of tweets with regards to sentiment according to TextBlob.

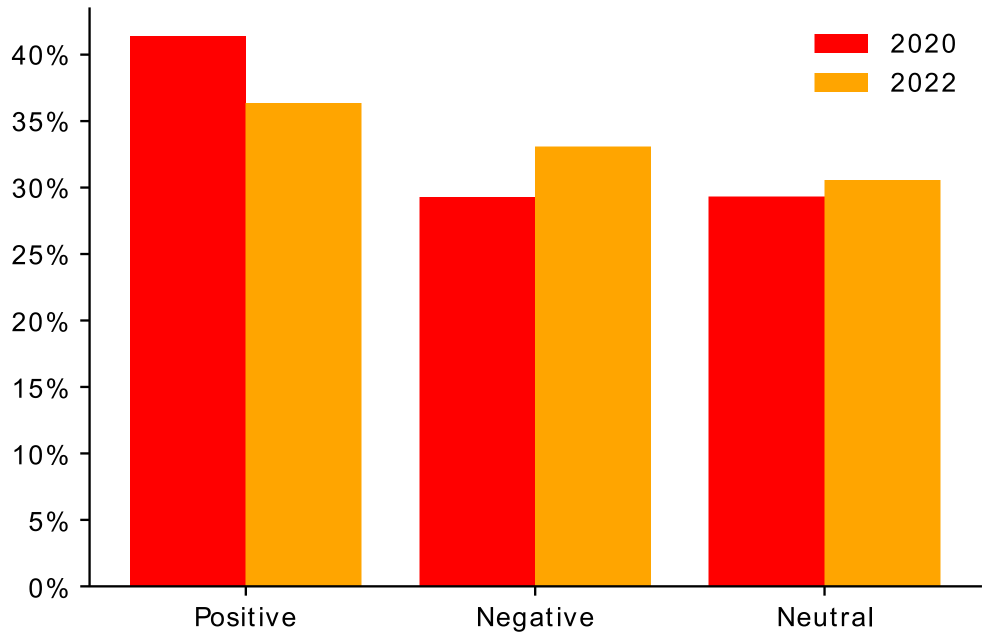


Figure 12: Difference in sentiment from 2020 and 2022. The histograms display the frequency distribution of tweets with regards to sentiment according to VADER.

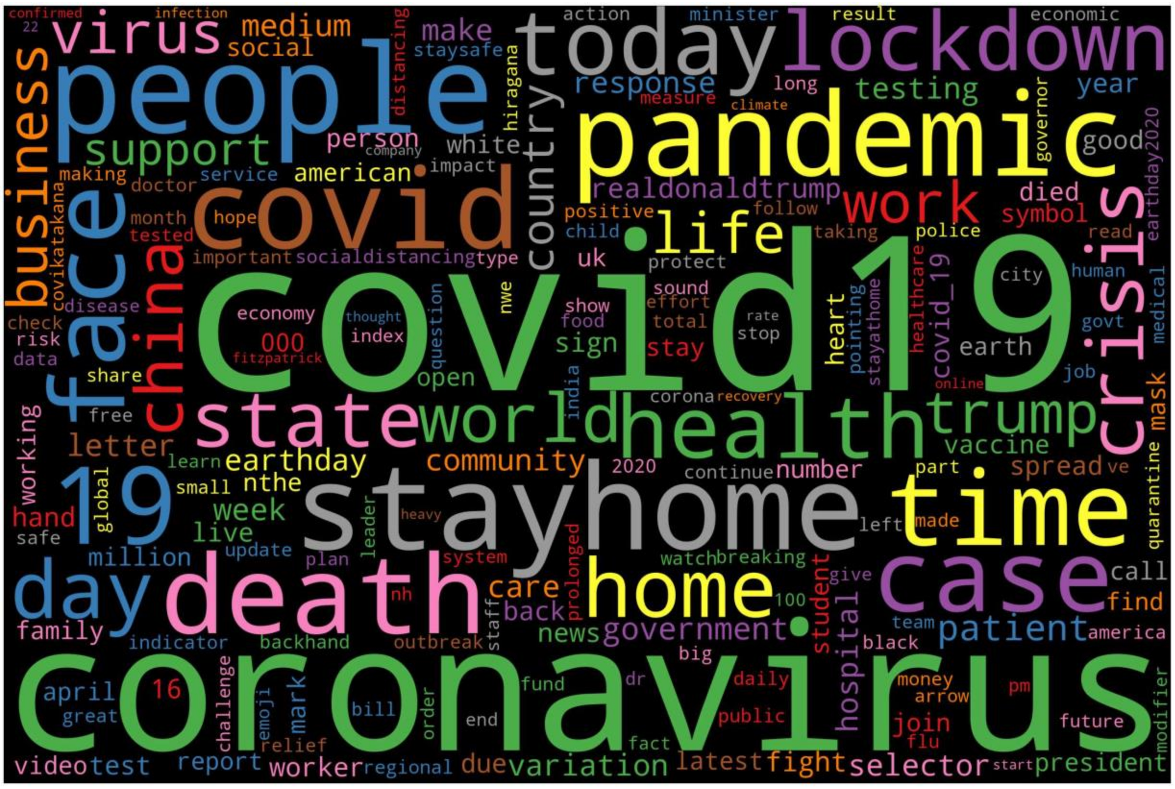


Figure 13: 2020 WordCloud

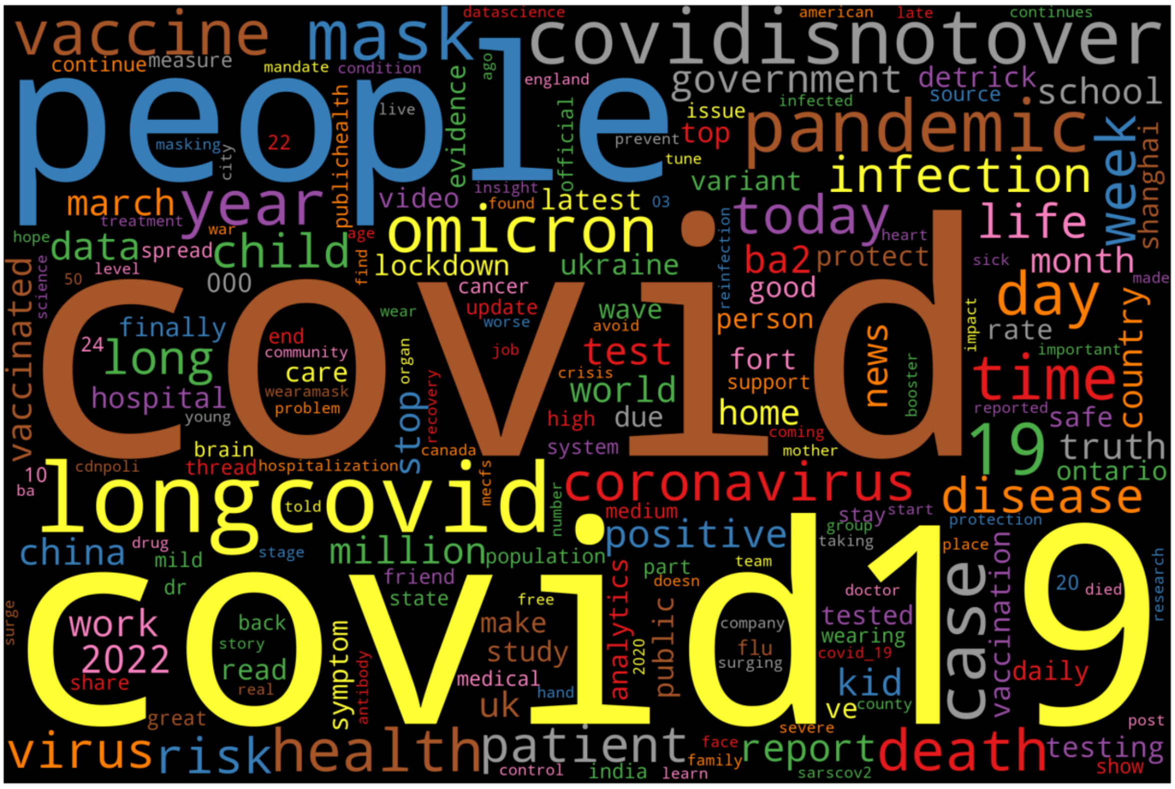


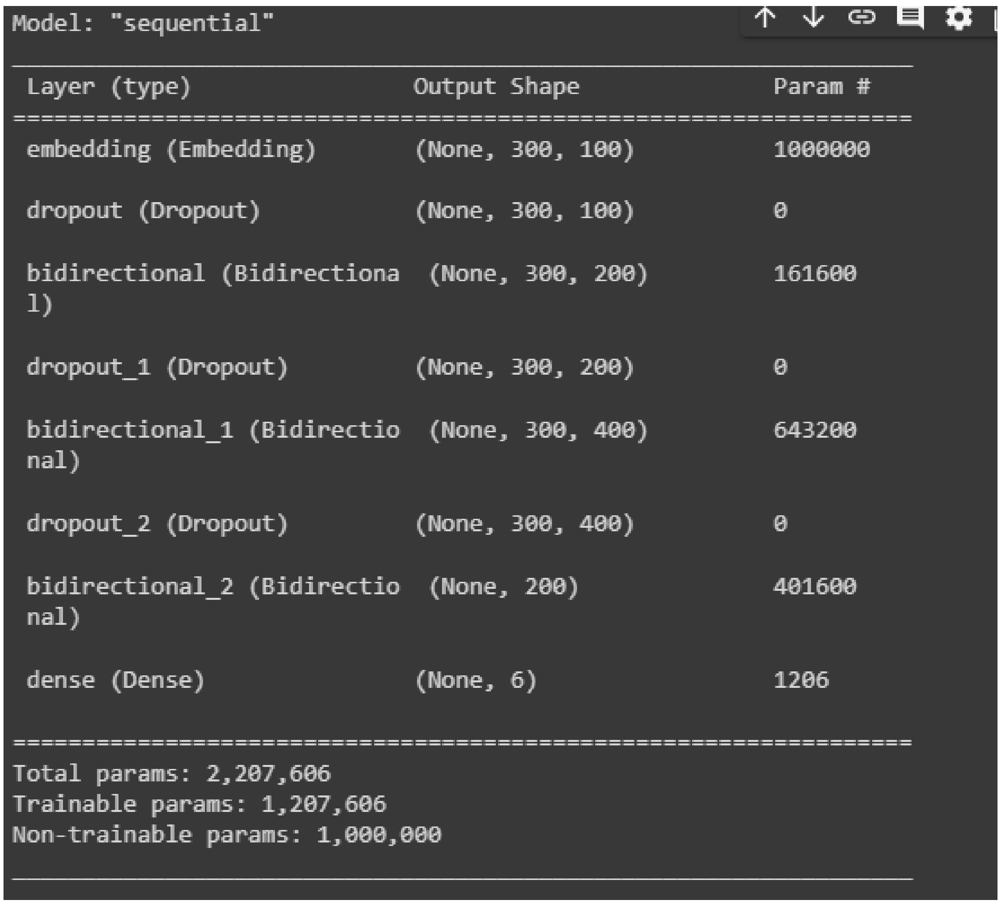
Figure 14: 2022 WordCloud

Table 5:TF-IDF word score for Corpus1(2020)

No.	Term	TF-IDF Score
1.	covid19	44186.93
2.	coronavirus	30987.27
3.	people	12507.46
4.	stayhome	11591.37
5.	pandemic	10836.62
6.	covid	10018.76
7.	death	9582.968
8.	time	8636.371
9.	face	8414.087
10.	case	8098.131
11.	today	8072.942
12.	health	7999.169
13.	lockdown	7264.073
14.	home	7140.71
15.	day	7002.378
16.	crisis	6670.096
17.	state	6634.335
18.	world	6533.159
19.	china	6238.218
20.	trump	6235.947

Table 6:TF-IDF word score for Corpus1(2020)

No.	Term	TF-IDF Score
1.	covid	24479.99
2.	covid19	24153.37
3.	people	10173.2
4.	longcovid	9681.456
5.	covidisnotover	9355.41
6.	pandemic	8398.545
7.	mask	8001.38
8.	case	7680.718
9.	omicron	6746.918
10.	year	6197.592
11.	time	6165.984
12.	vaccine	6087.344
13.	day	5636.658
14.	health	5593.961
15.	death	5279.887
16.	week	4805.976
17.	today	4568.455
18.	coronavirus	4556.032
19.	patient	4176.897
20.	virus	4138.394



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 100)          1000000

 dropout (Dropout)           (None, 300, 100)          0

 bidirectional (Bidirectiona  (None, 300, 200)         161600
 l)

 dropout_1 (Dropout)         (None, 300, 200)          0

 bidirectional_1 (Bidirectio  (None, 300, 400)         643200
 nal)

 dropout_2 (Dropout)         (None, 300, 400)          0

 bidirectional_2 (Bidirectio  (None, 200)              401600
 nal)

 dense (Dense)               (None, 6)                 1206

=================================================================
Total params: 2,207,606
Trainable params: 1,207,606
Non-trainable params: 1,000,000
_________________________________________________________________
```

Figure 15 : Model Summary

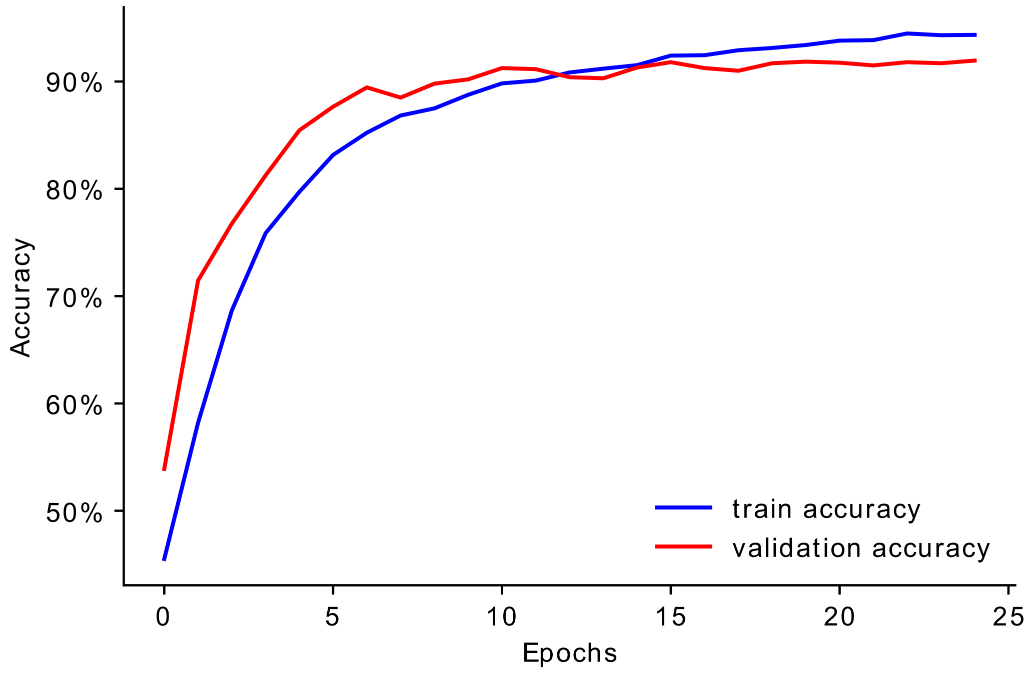
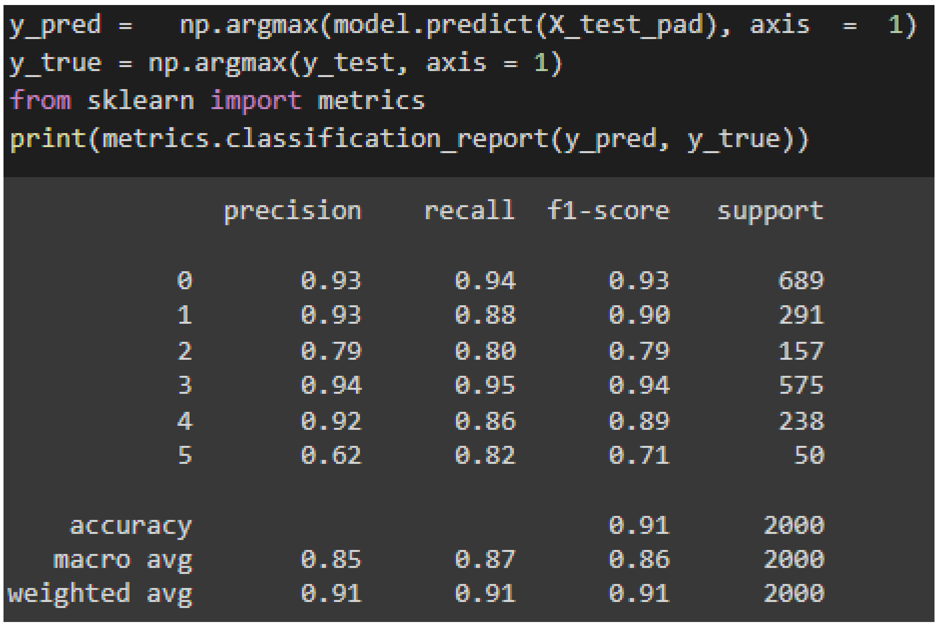


Figure 16 : Train vs Validation Accuracy



```
y_pred =   np.argmax(model.predict(X_test_pad), axis  =  1)
y_true = np.argmax(y_test, axis = 1)
from sklearn import metrics
print(metrics.classification_report(y_pred, y_true))
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       689
           1       0.93      0.88      0.90       291
           2       0.79      0.80      0.79       157
           3       0.94      0.95      0.94       575
           4       0.92      0.86      0.89       238
           5       0.62      0.82      0.71        50

    accuracy                           0.91      2000
   macro avg       0.85      0.87      0.86      2000
weighted avg       0.91      0.91      0.91      2000
```

Figure 17 : Performance Metrics

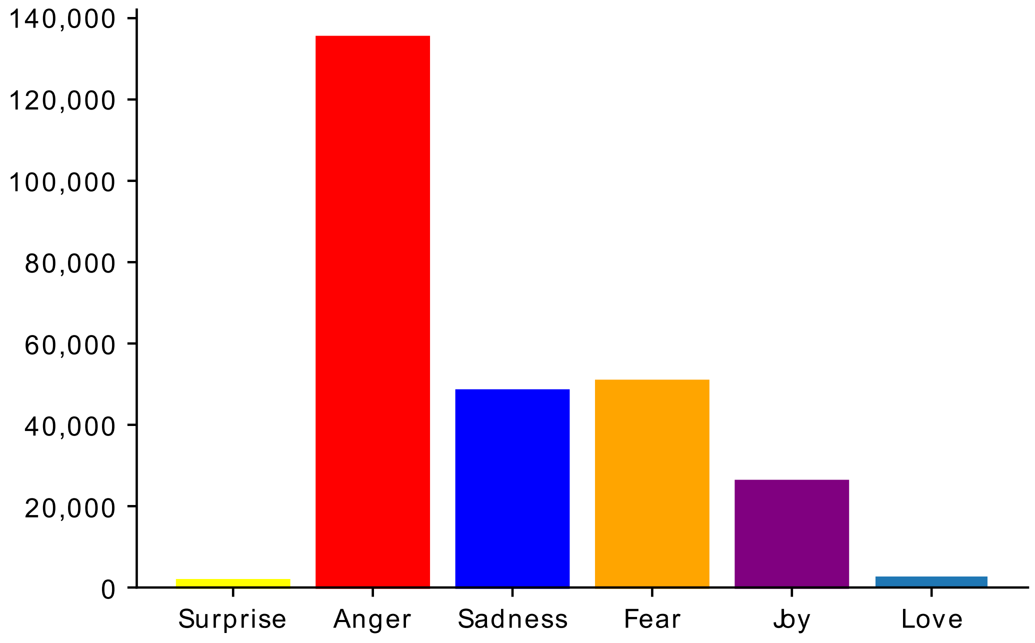


Figure 18: Frequency Distribution of tweets by emotions classified by our RNN-LSTM model.

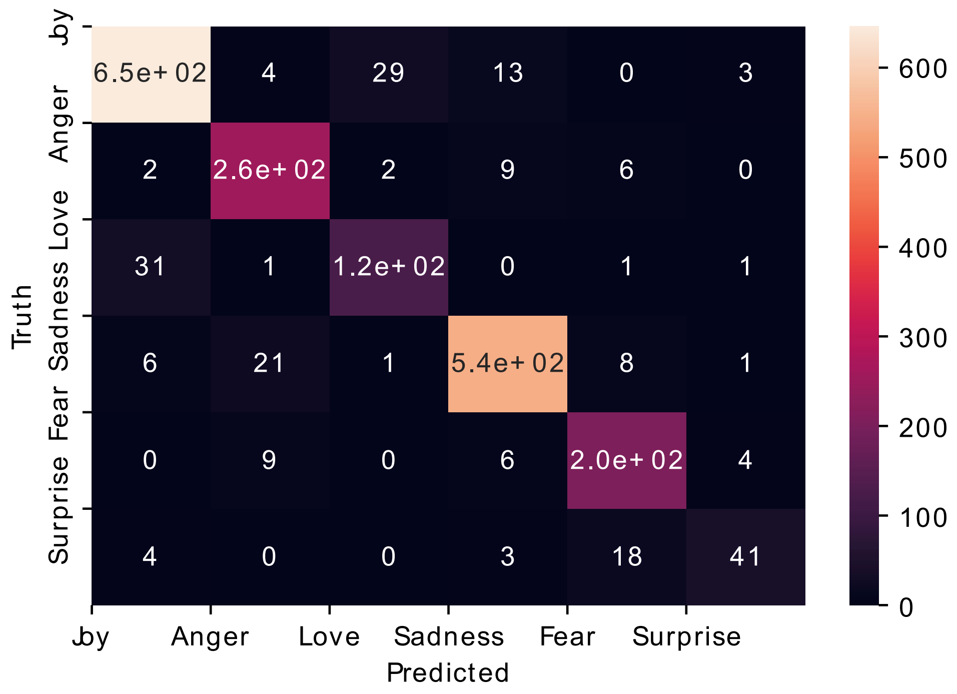


Figure 19: Confusion Matrix

Discussion

Analysing the various histograms and figures related to Ekman, Plutchik, POMS and Text2Emotion tools we can clearly see that there is not much change in the mood of people towards COVID-19 from 2020 to 2022. The same emotions which were dominant in 2020 continue to remain dominant in 2022. What was fascinating to observe was that Joy was the most dominant emotion according to the Ekman and plutchik models. This is quite odd as COVID-19 was associated with death, sickness, discomfort and a departure from the normal way of life, not something that denotes 'Joy'. Text2Emotion painted a gloomy picture with fear being the most dominant emotion in both 2020 and 2022, and POMS denoting that depression being the most dominant emotion in both 2022 and 2020. There are slight differences between the mood from 2020 and 2022, but the general trend remains constant.

When looking at the sentiment, according to textblob most of the tweets were classified as neutral in both 2020 and 2022, however according to VADER most of the tweets were classified as negative.

The most prominent terms in 2020 wordcloud include covid19, coronavirus, pandemic, lockdown, virus, stayhome, death, etc. The most prominent terms in 2022 wordcloud include covid19, longcovid, covidisnotover, omicron, vaccine, mask, etc. Clearly, after two years the terms such as stayhome have lost prominence.

When it comes to the RNN model it classifies the majority of tweets as anger, followed by fear and sadness. This is major departure from the emotions denoted by our Ekman and Text2Emotion models.

8 CONCLUSION

We were successful in building a model that predicts the emotions of given tweets, with an exceptional accuracy score. One pertinent observation we noted was that our RNN model predicted anger to be the most dominant emotion in 2022. Compared to 2020, fear was the most dominant emotion according to Text2Emotion. This makes sense as with the passage of time, people may have gotten accustomed to and fed up of COVID-19.

Areas for Improvement

Even though this project was successful in designing a model to predict emotions to a reasonably high degree of accuracy, in hindsight, there are still some measures we can undertake to enhance the project. The first one would be to balance the training dataset as some emotions were overrepresented. This would have further improved the classification accuracy of the model. Another approach would be to use a larger dataset to train our model. The current dataset used to train our model only has 16,000 labelled records. This would further boost the performance of our model. Also, if the training dataset was domain specific and related to covid-19 it would have made the resulting model extremely powerful. Another modification we could have made was to try the BERT model to predict our emotions. BERT which is a transformer model, is a recent development that can be tested for our purpose. Lastly, we could have user other metadata collected from twitter such as location, number of followers and used that to perform data analysis and group emotions based on these factors.

Challenges

Research on emotion detection on text pales in comparison to emotion detection, on facial expressions, body language, etc. The main reason is that textual data has many variables such as context, sarcasm, slang words, grammatically incorrect words, etc. This hinders any emotion detection model to a considerable extent.

Further Applications

Even though the primary objective of our model was to gauge the emotions of people regarding covid-19, this emotion model can be utilized in a variety of domains to tackle difficult problems. It can be used in the field of mental health to detect signs of depression, sadness and so on. Furthermore, it can be integrated in chatbot applications to provide psychiatric counselling and support to patients and thereby play an enormous role in suicide prevention. It can be also used on social media sights to flag hate speech and racist content. This will greatly help in attenuating the problem of cyberbullying, which is becoming widespread nowadays.
Lastly, it can be used by multinational companies to gauge the customer satisfaction towards a particular product or service. This will enable them to modify certain aspects and features which will help in retaining and attracting new customers. Politicians can also use this to gain an understanding of public opinion towards them and thus enhance their brand and personality accordingly to attract more followers and publicity.

9 REFERENCES

[1] M. A. Palomino and A. Padmanabhan Varma, "Any Publicity is Good Publicity: Positive, Negative and Neutral Tweets Can All Become Trends," 2020 39th International Conference of the Chilean Computer Science Society (SCCC), 2020, pp. 1-8, doi: 10.1109/SCCC51225.2020.9281266.

[2] C. Buckley, “Implementation of the SMART Information Retrieval System [Technical Report],” Cornell University, TR85-686, vol. 4, no. 4, p. 4, 19

[3] H. Schütze, C. D. Manning, and P. Raghavan, Introduction to Information Retrieval. Cambridge University Press Cambridge, 2008.

[4] C. Haynes, M. A. Palomino, L. Stuart, D. Viira, F. Hannon, G. Crossingham, and K. Tantam, “Automatic Classification of National Health Service Feedback,” Mathematics, vol. 10, no. 6, p. 983, 2022.

[5] N. Colnerič and J. Demšar, "Emotion Recognition on Twitter: Comparative Study and Training a Unison Model," in IEEE Transactions on Affective Computing, vol. 11, no. 3, pp. 433-446, 1 July-Sept. 2020, doi: 10.1109/TAFFC.2018.2807817.

[6] Saravia, Elvis & Liu, Hsien-Chi & Huang, Yen-Hao & Wu, Junlin & Chen, Yi-Shin. (2018). CARER: Contextualized Affect Representations for Emotion Recognition. 3687-3697. 10.18653/v1/D18-1404.

[7] Marco A. Palomina, Aditya Padmanabhan Varma, Gowriprasad Kuruba Bedala, and Aidan Connelly (In Press), "Investigating the Lack of Consensus among Sentiment Analysis Tools" Human Language Technology. Challenges for Computer Science and Linguistics. Springer International Publishing. (2020)

[8] Acheampong, Francisca & Chen, Wenyu & Nunoo-Mensah, Henry. (2020). Text-Based Emotion Detection: Advances, Challenges and Opportunities.

[9] Nandwani, P., Verma, R. A review on sentiment analysis and emotion detection from text. Soc. Netw. Anal. Min. 11, 81 (2021). https://doi.org/10.1007/s13278-021-00776-6

[10] Kusal S, Patil S, Kotecha K, Aluvalu R, Varadarajan V. AI Based Emotion Detection for Textual Big Data: Techniques and Contribution. *Big Data and Cognitive Computing*. 2021; 5(3):43. https://doi.org/10.3390/bdcc5030043

[11] Guo, J. (2022) Deep learning approach to text analysis for human emotion detection from big data. Journal of Intelligent Systems, Vol. 31 (Issue 1), pp. 113-126. https://doi.org/10.1515/jisys-2022-0001

[12] Alla, K.R., Kandibanda, N., Katta, P., Muthavarapu, A., Kuchibhotla, S. (2022). Emotion Detection from Text Using LSTM. In: Yang, XS., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Sixth International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems, vol 216. Springer, Singapore. https://doi.org/10.1007/978-981-16-1781-2_49

[13] M. -H. Su, C. -H. Wu, K. -Y. Huang and Q. -B. Hong, "LSTM-based Text Emotion Recognition Using Semantic and Emotional Word Vectors," *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, 2018, pp. 1-6, doi: 10.1109/ACIIAsia.2018.8470378.

[14] E. Batbaatar, M. Li and K. H. Ryu, "Semantic-Emotion Neural Network for Emotion Recognition From Text," in IEEE Access, vol. 7, pp. 111866-111878, 2019, doi: 10.1109/ACCESS.2019.2934529.

[15] https://developer.twitter.com/en/developer-terms/policy

[16] http://www.demos.co.uk/files/Road_to_representivity_final.pdf?1441811336

[17] https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/

[18] A. Leunes and J. Burger, "Profile of Mood States Research in Sport and Exercise Psychology: Past, Present, and Future," Journal of Applied Sport Psychology, vol. 12, no. 1, pp. 5–15, 2000

[19] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

APENDIX A

Twitter Streaming Code:

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

consumer_key=""
consumer_secret=" "
access_token=" "
access_token_secret=""
class TwitterStreamer():

    def __init__(self):
        pass

    def stream_tweets(self, fetched_tweets_filename, hash_tag_list):

        listener = StdOutListener(fetched_tweets_filename)
        auth = OAuthHandler(consumer_key,consumer_secret)
        auth.set_access_token(access_token,access_token_secret)
        stream = Stream(auth, listener)
        stream.filter(track=hash_tag_list,languages=['en'])


class StdOutListener(StreamListener):
    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:

            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True


    def on_error(self, status):
         if status == 420:
            return False
         print(status)
```

```
if __name__ == '__main__':


    hash_tag_list = ['#covid19',
'#longcovid','#coronavirus','#stayhome','#socialdistancing','#covid-
19','#covid2019','#coronavirusoutbreak', '#sarscov2', '#virus', '#covidisnotover',
'#covidvaccines','#vaccinated', '#longcovid', '#omicron', '#cases', '#covid', '#pandemic',
'#coronaviruspandemic', '#mask', '#deltacron', '#covidiots']
    fetched_tweets_filename = "final.json"

    twitter_streamer = TwitterStreamer()
    twitter_streamer.stream_tweets(fetched_tweets_filename, hash_tag_list)
```

Time Distribution Graph Code:

```
import numpy as np
from scipy.interpolate import make_interp_spline
import matplotlib.pyplot as plt




y = np.array([0,31562,40960,27205,32442,23713,35611,34859,38756,0])

x= [' Mar 24 23:30','Mar 25 11:30','Mar 25 23:30','Mar 26 11:30','Mar 26 23:30','Mar 27
12:30','Mar 28 00:30','Mar 28 12:30','Mar 29 00:30','Mar 29 12:30']

xf=np.arange((len(x)))
X_Y_Spline = make_interp_spline(xf, y)

# Returns evenly spaced numbers
# over a specified interval.
X_ = np.linspace(xf.min(), xf.max(), 500)
Y_ = X_Y_Spline(X_)

# plotting the points

plt.plot(X_,Y_, color= "red",linestyle="solid")
plt.gca().yaxis.set_major_formatter(plt.matplotlib.ticker.StrMethodFormatter('{x:,.0f}'))
plt.xticks(xf,x,rotation=90)

plt.xlabel('Time (GMT)')
# naming the y axis
plt.ylabel('Number of Tweets')
```

```
#plt.title('Distribution of Tweets over Retreival')

plt.grid()
plt.figure(figsize=(20,15))
#plt.savefig('plot.png', dpi=300)
plt.show()
```

Histogram Code:

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
from matplotlib.ticker import PercentFormatter
x=['Surprise','Anger','Sadness','Fear','Joy','Love']
y=[572,2159,4666,1937,5362,1304]
mpl.rcParams['axes.spines.right'] = False
mpl.rcParams['axes.spines.top'] = False
plt.rcParams.update({'font.family':'Times'})
plt.rcParams.update({'font.size': 10})
barlist=plt.bar(x, y)
plt.gca().yaxis.set_major_formatter(plt.matplotlib.ticker.StrMethodFormatter('{x:,.0f}'))
#plt.gca().yaxis.set_major_formatter(PercentFormatter(1,0))
barlist[0].set_color('yellow')
barlist[1].set_color('r')
barlist[2].set_color('b')
barlist[3].set_color('orange')
barlist[4].set_color('g')
barlist[4].set_color('purple')
plt.savefig('train.eps',format='eps',dpi=1200,facecolor='w',bbox_inches='tight')

plt.show()
```

Polarity Count code:

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
sid_obj= SentimentIntensityAnalyzer()

def pos(tweet):
    return sid_obj.polarity_scores(tweet)['pos']
def neu(tweet):
    return sid_obj.polarity_scores(tweet)['neu']
def neg(tweet):
    return sid_obj.polarity_scores(tweet)['neg']
def comp(tweet):
    return sid_obj.polarity_scores(tweet)['compound']
vader_df=pd.DataFrame()
vader_df['Tweet']=df['Tweet Text']
vader_df['Positive']=vader_df['Tweet'].apply(pos)
vader_df['Neutral']=vader_df['Tweet'].apply(neu)
vader_df['Negative']=vader_df['Tweet'].apply(neg)
vader_df['Compound']=vader_df['Tweet'].apply(comp)
vader_df
```

```
pos=[]
neg=[]
neu=[]
for i in df['Polarity']:
    if i>(0.2):
        pos.append(i)
    elif i<(-0.2):
        neg.append(i)
    else:
        neu.append(i)
posa=[abs(number) for number in pos]
nega=[abs(number) for number in neg]
neua=[abs(number) for number in neu]
print(sum(posa))
print(sum(nega))
print(sum(neua))
print(len(posa))
print(len(nega))
print(len(neua))
```

Text2Emotion:

```
 def text_emotion(tweet):
     return te.get_emotion(tweet)
 emotion=pd.DataFrame()
 emotion['Text']=df['Text']
 emotion['Text2Emotion']=emotion['Text'].apply(text_emotion)
 emotion
```

```
text2emo=pd.DataFrame()
text2emo['Text']=emotion['Text']
happy=[]
angry=[]
surprise=[]
sad=[]
fear=[]
for i in emotion['Text2Emotion']:
    happy.append(i['Happy'])
    angry.append(i['Angry'])
    surprise.append(i['Surprise'])
    sad.append(i['Sad'])
    fear.append(i['Fear'])

text2emo['Happy']=happy
text2emo['Angry']=angry
text2emo['Surprise']=surprise
text2emo['Sad']=sad
text2emo['Fear']=fear
text2emo
```