Project Report for COL865 - Special Topics in Computer Applications - Social Computing

Leveraging Twitter data to comprehend emotion detection and cross-cultural polarity utilising Sentiment analysis for Covid19

Group Members:-

Ritika Singh (2021AIZ8329)

Sweta Mahajan (2021AIZ8356)

Indian Institute of Technology, Delhi

School of Artificial Intelligence

July 26, 2022

Contents

List of Figures

1 Abstract

This research seeks to utilise data through social media platform Twitter, at which people engage among themselves by making posts called tweets. They tweets about hot button issues often utilise relevant hashtags. Twitter is a data-rich channel. Assessing tweets would provide significant and vital observations into what people are talking about, their thoughts on this matter, their perspectives on a particular premise, and greater societal trends. Many enterprises and government agencies needs to track the sentiment of the people on matters of importance which will help them take better decisions without hurting public sentiments.

COVID-19 has been recognized as a dominant and serious pandemic around the world. On February 28, 2020, the World Health Organization (WHO) declared COVID-19 a Public Health Emergency of International Concern (PHEIC). This project is intended to focus and provide a better insight into the concept of sentiment analysis in a social media platform by studying related literature published between 2015 to 2021 and implementing certain aspects of it. Our goal is to improve the analysis of Twitter data for Covid-19 data in order to measure the impact and behaviour of users toward various event categories. The following work will concentrate on studying the data and its attributes, as well as investigating modelling techniques to determine the frequency distribution for each event. The ability to collect and analyse such data can provide vital insights into how products, services, political figures, companies, governments, and so on are perceived.

The automated process of identifying and classifying subjective information in text data is known as sentiment analysis. This could be a personal belief, an assessment, or emotions about any issue or brand experience. The most commonly used type of sentiment analysis is 'polarity detection,' which entails classifying remarks as Positive, Negative, or Neutral. It employs Natural Language Processing (NLP) to understand human language and machine learning to deliver accurate results automatically. The procedure includes steps for extracting, converting, and interpreting opinions from text and categorising them as positive, negative, or natural sentiment.

2 Introduction

The transmission of this malady had already produced substantial economic chaos, individual health hazards, and confusion, which all have monopolised both the media and internet. The pervasive use of smartphones and tablets allows individuals to share their self on social networking sites such as Facebook and Twitter. Sentimentality plays a vital part in impactful interpersonal interactions and it requires substantial effort in making rational decisions. Message is indeed a critical feature of affective computing since most users use emails and texts via workstation to voice their views. At the root level, opinions influence actions that take place or are profoundly affected as a direct consequence of those opinions. Opinion analysis, also known as sentiment analysis, is crucial in attempting to make as close an estimation as feasible. It is a crucial component seeing as meticulously designed and executed sentiment analyses could indeed produce faster and more accurate projections for both business and politics. In it's most fundamental, sentiment analysis is based on the beliefs conveyed or articulated by individuals and participants. People publish information in hundreds of billions of bytes on social platforms, blog posts, and brand reviewers in the Digital space, that encompasses almost all known digital realm and territory of human influence.The sheer volume and variety of data on Twitter, as well as the public nature of tweets, have enabled the use of Twitter information in data analysis. The first is to measure the life cycle of a specific topic by counting the number of tweets over time, and the second is to measure user sentiment toward a specific topic using NLP and ML algorithms.

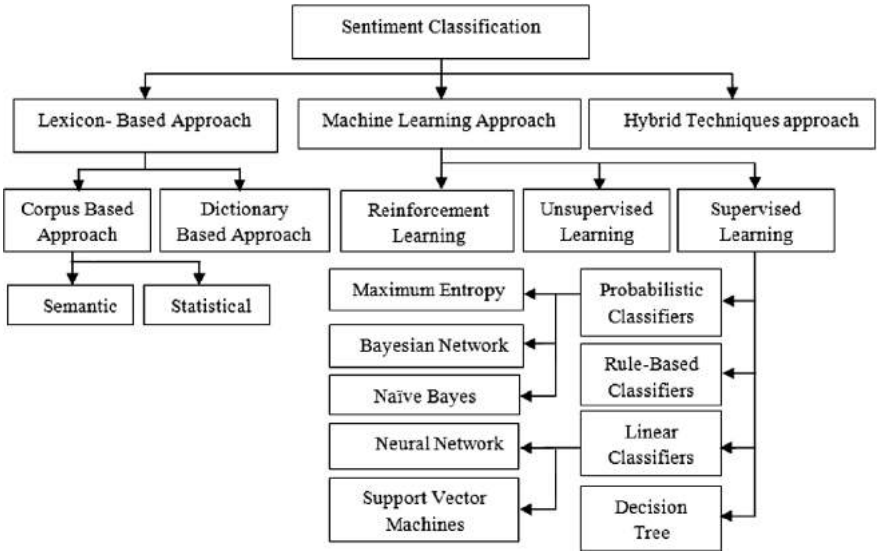


Figure 1: Types of sentiment classifiers

Sentiment Analysis can face a variety of challenges, including issues with accuracy, binary classification, data sparsity, and polarity shift. The models used in sentiment analysis tend to be valid only around the event as new hashtags and informal acronyms develop over time which the model is not aware of. While alternative techniques for sentiment classification have indeed been proposed and designed, there is still a need for an efficient method for extracting as well as generating reliable semantic analysis on a

continual basis. Various social networking channels have been used to interact, discuss daily activities and personal opinions during COVID-19 global epidemic, including the many meaningful messages (writings) by participants expressing their current mood about one's individual situation, medical status, suggestions to keep well, and other related data. Those very texts might provide vast understanding of how individuals are responding to the disease outbreak. The overwhelming number of tweets makes it hard to analyse the data. Moreover, over time Twitter has developed an informal language of its own called "Twitter lingo" which includes many acronyms that help define the emotion attached with the tweet. Hence making the model aware of this lingo is crucial.

Sentiment Analysis (SA) is a heavy computational study into human behaviours, perspectives, and emotional responses intrinsically as such human sentiments have all been described in terms of any entity. An element can be shown to demonstrate an incident, an individual, or maybe even a topic. SA incorporates three terminology to extract sentiment. They are : object and feature, opinion holder, and opinion and orientation. Natural Language Processing (NLP), text reviews and mathematical tools for extracting relevant information and classifying sentiments from customer reviews are indeed the paramount backbones onto which SA is built.

This project paper provides a comprehensive survey of prominent Sentiment Analysis tools and models, with the objective of submitting a concise evaluation report on which to build research work.

Sentiment Analysis is traditionally seen as a classification task where depending on the content and keywords used in the tweets, the polarity of the tweets can be categorized into positive or negative. SVM, NB, RF, DL methods have been popularly used. Whether twitter users represent society. They compared accuracy, precision, recall, F1 score of the NB, TAN, BF TAN, SVM, RF models. Drawbacks: 1) Only texts in the tweet is analysed, hashtag information is not utilised. 2) the approach is heavily time and event dependent, so the model is valid only for a short time span around the event.

3 Literature Survey

1. **Saragih, M. H. (2017). "Sentiment Analysis of Customer Engagement on Social Media in Transport Online". IEEE. [13]**
   The study outlines the sentiment analysis of social media report information. Researchers have used the naive Bayes and Levenshtein methodologies to classify the feeling into distinct classes from social media news information. This methodology offers the best efficiency for real-time news information through social media, and a greater result in terms of accuracy and reliability. They observed that the Levenshtein formula delivers a relatively basic text processing on data. It works fast and offers a high level of precision while handling massive quantities of data.

2. **Khan, Muhammad Taimoor, Mehr Durrani, Armughan Ali, Irum Inayat, Shehzad Khalid, and Kamran Habib Khan (2016). "Sentiment Analysis and The Complex Natural Language". Complex Adaptive Systems Modeling. [5]**
   This study demonstrates that methodologies depend solely on lexical features, and the cumulative effectiveness of the technique is sensitive to the quality of the lexical items. It is premised on the reality that polarity of a block of content can be established mostly by polarity of phrases which constitute it. Owing to the complexity of natural languages, one such technique is not designed to cover all features of language, notably colloquialism, facetiousness, and contradiction. Leveraging sentimental phrases is woefully inadequate. A few issues have emerged, like the fact that certain phrases have multiple interpretations depending on the specific application, that certain paragraphs incorporating sentiment lexicons might not always articulate any sentiment, yet many sentences without sentiment may indeed insinuate an opinion.

3. **Mittal, S. A. (2016). "Sentiment Analysis of E-Commerce and Social Networking Sites". IEEE. [7]**
   The paper identifies the need for as well as the impact of sentiment analysis on an interactive website. They have indeed conferred a catalogue of emotional sentiments, catchphrases, and remarks retrieved from posts and wall posts. They found whether or not the online reviews and comments are useful to customers or not, and also which online services are most famous among consumers.

4. **Kaur, H. J. (2015). "Sentiment Analysis from Social Media in Crisis Situations". IEEE. [4]**
   The paper exemplifies based on geography location flood data sets collected from Twitter and understands individual citizens' opinions. The researchers classified information by using Naive Bayes formula, and the final outcome was a 67 percent precision and predictability. They ought to have collected collect several resolutions from individual citizens that will be useful for each government and non-government organisation to cope with such a situation in a far more efficient way. The strategies applied are less complex than lexicon-based formulas.

5. **Gonzalo A. Ruz, Pablo A. Henríquez, Aldo Mascareño (2020). "Sentiment analysis of twitter data during critical events through bayesian networks classifiers". Future Generation Computer Systems [11].**
   This article analyses Spanish tweets for the 2017 catalan independence referendum and the 2010 Chilean earthquake. Their aim 1)to verify whether usual approaches English text like SVM and Random Forests work for Spanish dataset 2) Using Bayesian networks, to find out the relations among words for sentiment classification in Twitter. In this work, they have used the words instead of the word embedding. They compared performance of Naive Bayes, the Tree Augmented Naive bayes, Bayes Factor Tree Augmented Naive bayes, SVM, Random Forest. They found lexical features(1 word, 2-words) are most effective input features in their experiment.

6. **Venkata K. Neppalli, Cornelia Caragea, Anna Squicciarini, Andrea Tapia, Sam Stehle (2017). "Sentiment analysis during Hurricane Sandy in emergency response". International Journal of Disaster Risk Reduction [9].**
   The authors showed how sentiments vary in space and time around the hurricane. They use bag of words and sentiment features e.g., polarity clues and emoticons. They have used unigrams, polarity clues,emoticons, internet acronyms, punctuation, SentiStrength score as the input features and an SVM to classify the tweets. They also investigate the relationship between the emotional divergence(which is different from sentiment analysis) and informativeness of a tweet. Emotional divergence in a short text measures the range of the emotions expressed in it, whereas the overall emotion is captured by sentiment analysis. They discovered tweet with low emotional divergence is generally informative in nature, whereas content with high emotional divergence is more of a personal opinion and might not convey any useful information. They extracted the retweeted tweets and and compared how the number of retweets of a tweet vary with the emotional divergence of the tweets.

7. **Goularas, Dionysis and Kamis, Sani (2019). "Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data". International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML). [3]**
   This paper presents comparisons between different deep learning based methods for sentiment analysis. Particularly, the CNN and LSTM networks proved to be better for sentiment analysis tasks among the deep learning based models although they performed a bit worse than the state of the art. The results using Word2Vec and GloVe embeddings were also compared. RCNN and GRU networks are not utilized because in their experiments they had very similar performance with CNN and LSTM networks respectively. They used 300 epochs and sigmoid activation function. Accuracy, Precision, Recall, F score were used to compare the results. The paper says training data is the key to success in analysing sentiments of twitter data. The authors observed that CNN and LSTM combined give better performance than when they are used alone. This could be due to the dimensionality

reduction property of CNN's and the preservation of word dependencies of LSTM networks.

8. **Ghasiya, Piyush and Okamura, Koji (2021). "Investigating COVID-19 News Across Four Nations: A Topic Modeling and Sentiment Analysis Approach". IEEE Access. [2]**
   This paper performs topic modelling and sentiment analysis using Top2Vec and RoBERTa models respectively. It compares the topics and the sentiments discovered across four nations i.e., UK, India, Japan, South Korea. Topic modeling results highlight that education, economy, US, and sports are the frequently reported themes across the four countries. Analysis shows that the worst affected country, i.e. the UK (in their dataset) also has the highest percentage of negative sentiment. The work done here varies from the traditional task of sentiment analysis as the authors here use a transformer based model called RoBERTa which takes context into account unlike models like LogisticRegressionCV, LinearSVC, and Naïve Bayes which does not take into account the order of the words in a sentence. Eventually, they compared the results about the four nations.

9. **Khairiyah Mohamed Ridhwan and Carol Anne Hargreaves (2021). "Leveraging Twitter data to understand public sentiment for the COVID-19 outbreak in Singapore". International Journal of Information Management Data Insights. [8]**
   This paper carries out topic modelling and sentiment analysis exclusively for Singapore. They discovered the dominant topics discussed during the timeline in Singapore. They performed overall sentiment analysis using VADER and an overall emotion (anger, sadness, joy etc) analysis. using pre-trained RNN. For topic modelling, they used Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM) to find out the relevant topics in the dataset and used LDA to find the optimal number of topics. For each of the topics discovered, they investigated the sentiment associated with it. The authors concluded the overall sentiment was positive whereas sentiment polarity varied from topic to topic which shows taking topics into account is a better way to understand tweets better. Future research work 1) Network analysis of twitter users can be performed to find the influential people who can be used to disseminate information informally to the public. 2) Topic based modelling can be done in facebook or reddit where users can be analysis in various age categories.

10. **Symeon Symeonidis and Dimitrios Effrosynidis and Avi Arampatzis (2018). "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis". Expert Systems with Applications. [14]**
    As the name suggests this paper talks about number of ways to pre-process the data to get better accuracy. They empirically compare 16 commonly used pre-processing techniques for Sentiment Analysis on two Twitter datasets. For this task they use Logistic Regression, Linear SVC, Bernoulli Naïve Bayes and Convolutional Neural Networks. They analyse the effect of different pre-processing techniques on the classification accuracy and the number of features they produce. They find

lemmatization, removing numbers, and replacing contractions to improve accuracy while removing punctuation does not. They also perform ablation study to investigate how accuracy varies as a combination of these pre-processing techniques are adopted.

11. **Matalon, Yogev and Magdaci, Ofir and Almozlino, Adam and Yamin, Dan (2021). "Using sentiment analysis to predict opinion inversion in Tweets of political communication". Scientific Reports, NATURE.** [6]
Studies on the opinion diffusion process have brought to light the phenomenon of Opinion Inversion which referes to the act of inverting a message's content before disseminating it, propagating a opposite view relative to that of the original author. The authors have investigated this approach in connection to the Palestine-Israeli conflicts. Using a Random Forest model based on the Natural Language Processing features of the Source text and user attributes, we could predict whether a Source will undergo O.I. upon retweet with an ROC-AUC of 0.83. We found that roughly 80 percent of the factors that explain O.I. are associated with the original message's sentiment towards the conflict. O.I. can be predicted in advance using simple artificial intelligence tools and that prediction might be used to optimize content propagation.

12. **Diaz, Mark and Johnson, Isaac and Lazar, Amanda and Piper, Anne Marie and Gergle, Darren (2018). "Addressing Age-Related Bias in Sentiment Analysis". Association for Computing Machinery** [1]
The authors perform a systematic analysis of age related bias in 15 of the sentiment analysis models and few word embeddings. They found significant age bias in the output for example, sentences with "young" adjectives are 66 percent more likely to be scored positively than the same sentences with "old" adjectives. Since analysing one model could have idiosyncratic findings particular to that model, the authors explored around 15 models. This also allows them to compare common implementation techniques that might be responsible for bias. To attain this task, they examine the sentiment output scores using multinomial log-linear regressions.

13. **Yafeng Ren, Ruimin Wang, Donghong Ji (2016). "A topic-enhanced word embedding for Twitter sentiment classification". Information Sciences.** [10]
The authors modify the existing methods to take the topic information into account while predicting the word embeddings which gives improved macro F-measure. Word representation is very much crucial for accuracy in sentiment analysis in Twitter as there are less number of words in each tweet. First the topic information is generated through Latent Dirichlet Allocation (LDA) and a recursive autoencoder framework is used to learn topic-enhanced word embedding. They use SVM for classification task. Experimental results on the dataset show that topic-enhanced word embedding is very effective for Twitter sentiment classification.

4 Project Checlpoint 2 begins here:

5 Authentication process

Twitter API will be used to scrape the data from Twitter. To use this API, a developer account on Twitter's website must be created. A request was made to gain access to Academic Research (elevated plus).

	Essential	Elevated	Elevated+ (coming soon)	Academic Research
Getting access	Sign up	Apply for additional access within the developer portal	Need more? Sign up for our waitlist	Apply for additional access
Price	Free	Free		Free
Access to Twitter API v2	✓	✓		✓
Access to standard v1.1	✗	✓		✓
Access to premium v1.1	✗	✓		✓
Access to enterprise	✗	✓		✓
Project limits	1 Project	1 Project		1 Project
App limits	1 App per Project	3 Apps per Project		1 App per Project
Tweet caps	Retrieve up to 500k Tweets per month	Retrieve up to 2 million Tweets per month		Retrieve up to 10 million Tweets per month

Figure 2: Difference in the access levels for twitter data extraction

The Post on twitter has been one of Twitter's vital resources. A Tweet, at it's most simplest form, could perhaps constitute up to 280 characters and it can be posted private or public, contingent mostly on configurations of an account. Moreover, a range of different items, including media, a location, polling data, and Web links, could be linked to a Tweet. Although there are innumerable HTTP, choice, and methods of delivery currently offered for conveying, publication, and acting on Tweets, the above collection of REST end - points merely reverts a Twitter post or cluster of Tweets stipulated by a Tweet ID.

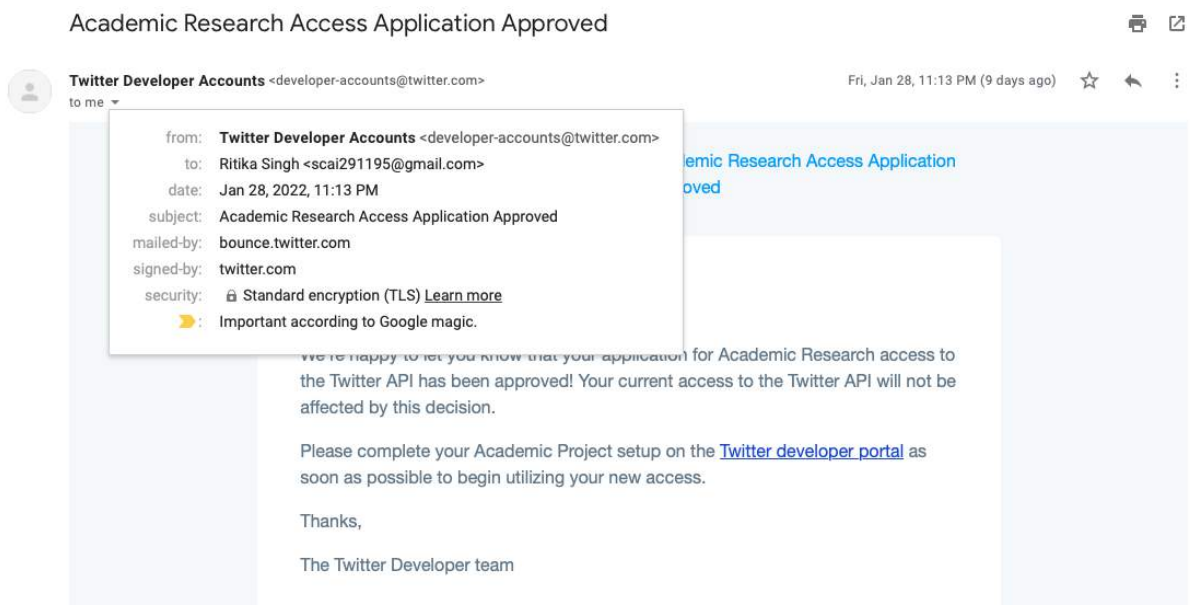


Figure 3: Confirmation of academic access(elevated access) approved

The twitter platform may indirectly influence traditional media agenda especially during critical events like the pandemic COVID19 as journalists gather information from tweets and retweet valuable tweets shared by users [11].

5.1 Authorizing requests

APIs employ permission to guarantee that user requests obtain secure access to the data. It may also involve multi factor authentication from the sender of such a request and verifying that they also have the proper authorization to access or modify the pertinent information. When you're generating an API, users can chose from a handful of validation configurations. Whether you're incorporating a third-party API, the Application provider would then define permissions needed.

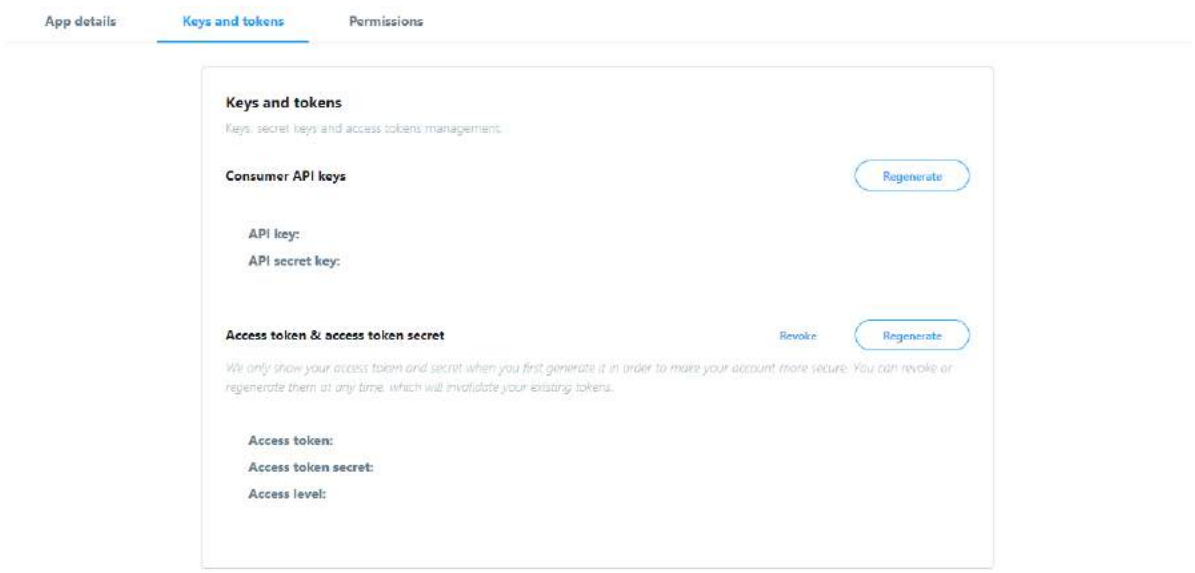


Figure 4: Where to find login credentials.

Following verification, the following keys are received: Consumer Key, Consumer Secret, Access Token, and Access Token Secret. Make a note of them and keep them somewhere safe. The Twitter API can be used to programmaticaly retrieve and analyse Twitter data, as well as to build for the Twitter conversation. The Twitter API has grown over time by providing new access privileges for developers and university researchers for being able to accelerate one‘s access to boost and study the larger dialogue.

Then we'll create a script to fetch the twitter posts. To proceed, import all the required packages and configure the token and key variables. Authorization primarily enables the user to offer some other website/service a restricted access access token for permission to extra resources through an authentication provider with which they may have previously successfully verified.

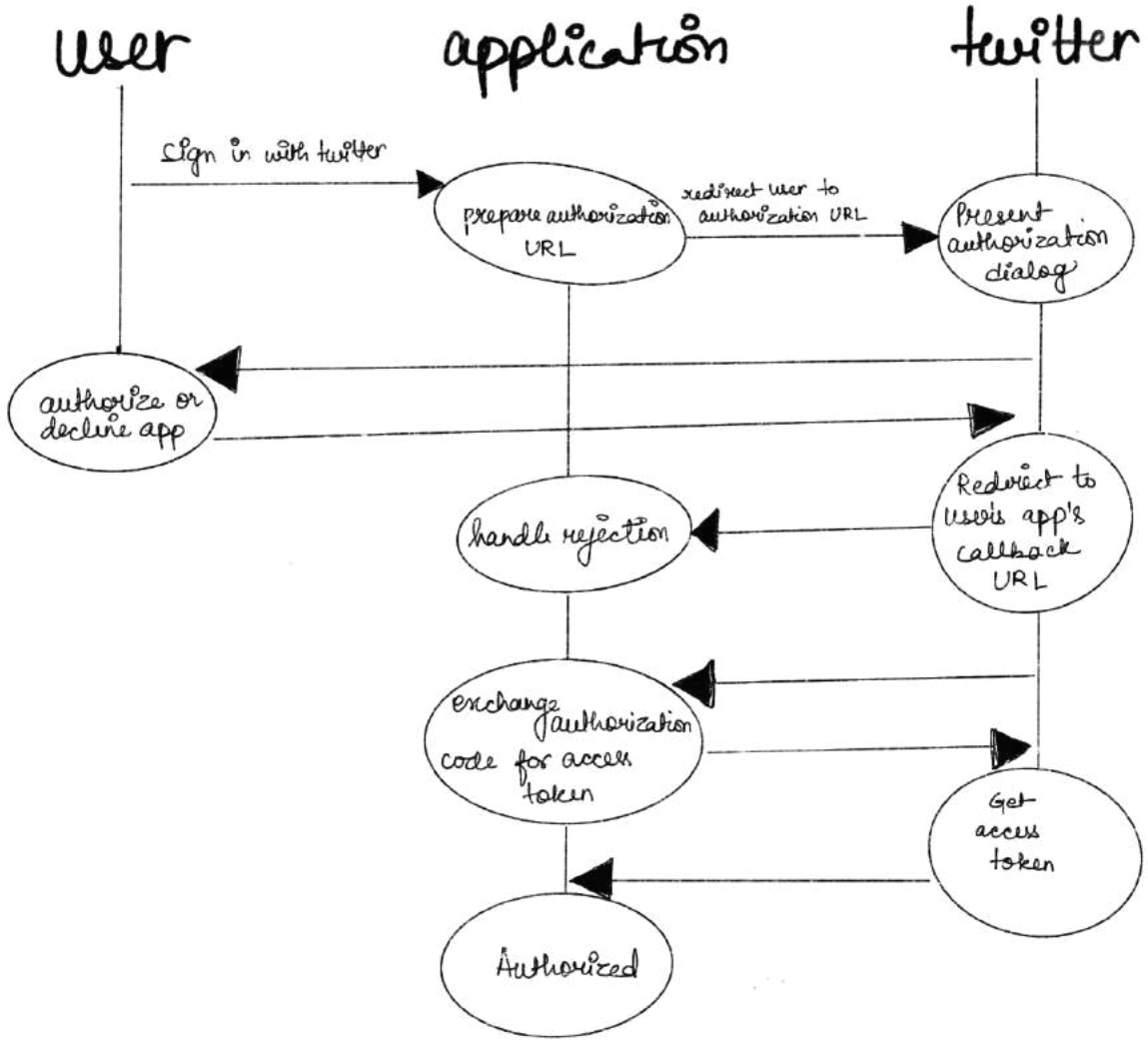


Figure 5: Flowchart of authorization for fetching twitter data

The tweets related to COVID19 are obtained via the twitter API. Certain keyword like "COVID", "corona", "SARS-COV 2" were given to extract related tweets. In this report text and sentiment features will be used for the experimental tests, and the aim is to build a deep learning framework that uses a twitter post as input and tries to tell us emotion/ feelings associated with it. It is a multi-class classification problem as we have 3 kinds of sentiments (Positive, Negative, and Neutral).

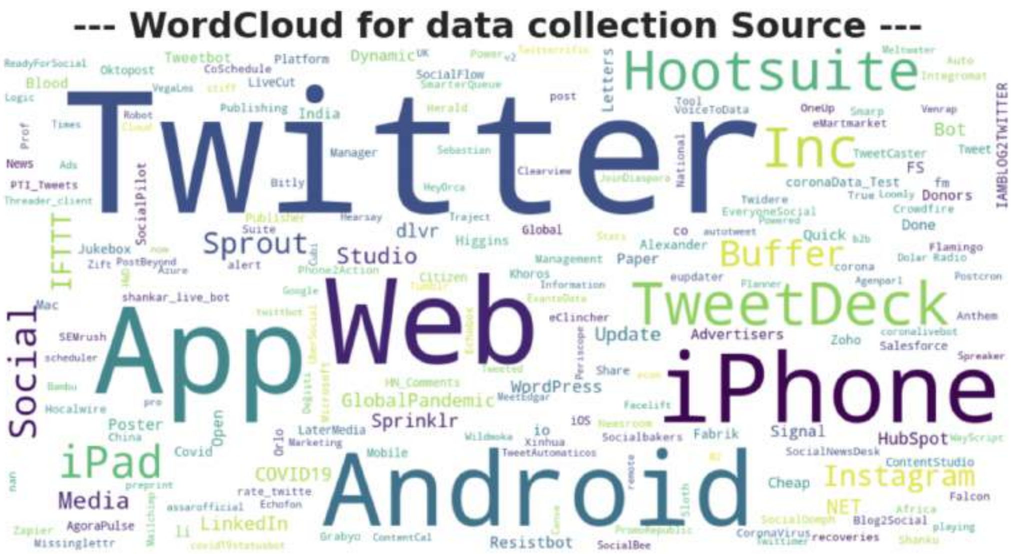
6 Data Collection

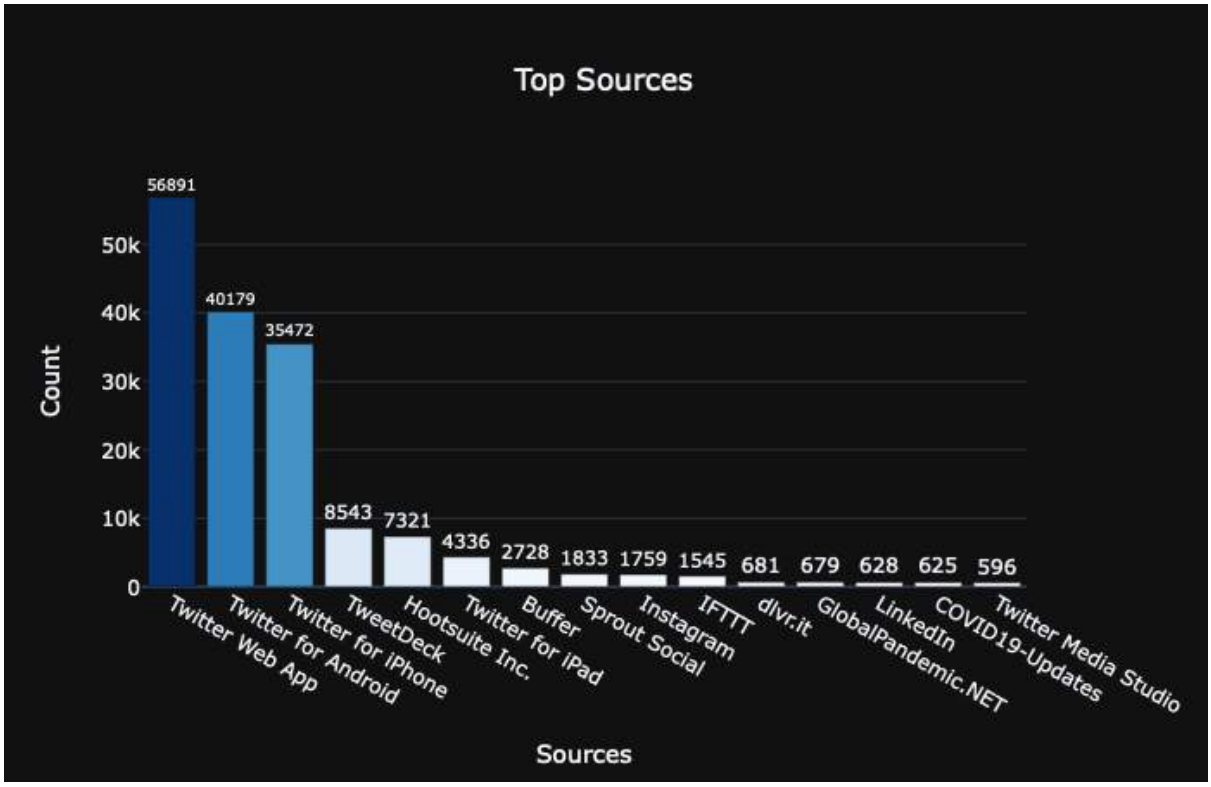
We use the Python Twitter API – Tweepy to look for public twitter posts which incorporate the terms "COVID", "cowin", "vaccine", and so on. The company's Twitter API is an excellent tool for mining algorithms. API can be used by data research scientists to obtain Twitter data under limited use. The Twitter API enables users to conduct query processing including extracting Tweets relevant to the subject within a given timeframe.

The following steps can be taken to determine the sentiment of tweets:

1. Use Twitter data sets that you've downloaded and entered into the ipython notebook.
2. Remove all stop words from the tweets.
3. In this step, each word in the dataset has to be tokenized and sent into the software.
4. Compare each word to the dictionary's good and negative feelings words. Then, either increase the positive or negative count.
5. Finally, we may calculate the emotion result percentage based on the positive and negative count to determine the polarity.

```
def wordcloud(string,title,color):
    wc = WordCloud(background_color=color, width=1200,height=600,mask=None,random_state=1,
                   max_font_size=200,stopwords=stop_words,collocations=False).generate(string)
    fig=plt.figure(figsize=(20,8))
    plt.axis('off')
    plt.title('--- WordCloud for data collection {} --- '.format(title),weight='bold', size=30)
    plt.imshow(wc)

stop_words=set(STOPWORDS)
source_string = " ".join(df['source'].astype('str'))
hastage_string = " ".join(df['hashtags'].astype('str'))
location_string = " ".join(df['user_location'].astype('str'))

wordcloud(source_string,'Source','white')
```





We use a variety of methods to connect to the Twitter API, retrieve tweets, remove stop words, classify tweets, and finally return the results, as can be seen in the algorithm.

Many people's views on the data can be found in the tweets, which are conveyed in a variety of ways. In this study, we used an already-labeled collection of tweets from Twitter. It is easier to analyse data when it is labelled with a positive and negative polarity. There's a lot of room for error and redundancy in raw data that has polarity. Because the quality of the data has an impact on the findings, pre-processing the raw data is necessary. It focuses on the process of removing redundant words and punctuation and increasing the data's efficiency.

There are a plethora of unique qualities in the preprocessed dataset after it has been enhanced. Aspects (adjectives) are extracted from the dataset using the feature extraction method. To show positive and negative polarity in a text, this adjective is later employed in the unigram model [15]. Using a unigram model, an adjective is extracted and separated from the rest. It removes the preceding and following words that appear in the phrases with the adjective. In the previous example, "painting Beautiful" was reduced to just "Beautiful" using a unigram model.

In [1]:

```
import os
import sys
import datetime
import csv
```

In [5]:

```
from textblob import TextBlob
import sys
import tweepy
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os
import nltk
import pycountry
import re
import string
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from langdetect import detect
from nltk.stem import SnowballStemmer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import CountVectorizer
```

In [6]:

```
consumerKey = "gHbIJKKUw8epSVMHerB46jFB3"
consumerSecret = "9GooX0S7SOdkva6c1hlTSzrbOTh3fddbgWF4XXyobDHYsYvHiL"
accessToken = "1487100508016099330-cPWINEzI6DZCuCVlsO2vm4bTWMMV5e"
accessTokenSecret = "YHA2HAchMCwvF8LrRNgPrMAsLNDqXe3xqwRV9IpRxlotI"
auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)
```

In [16]:

```
def percentage(part,whole):
 return 100 * float(part)/float(whole)

keyword = input("Please enter keyword or hashtag to search:")
#keyword = input("Please enter keyword or hashtag to search:")
noOfTweet = int(input("Please enter how many tweets to analyze:"))
#noOfTweet = int(input ("Please enter how many tweets to analyze: "))
tweets = tweepy.Cursor(api.search_tweets, q=keyword).items(noOfTweet)
positive = 0
negative = 0
neutral = 0
polarity = 0
tweet_list = []
neutral_list = []
negative_list = []
positive_list = []
for tweet in tweets:

 #print(tweet.text)
 tweet_list.append(tweet.text)
 analysis = TextBlob(tweet.text)
 score = SentimentIntensityAnalyzer().polarity_scores(tweet.text)
 neg = score['neg']
 neu = score['neu']
 pos = score['pos']
 comp = score['compound']
 polarity += analysis.sentiment.polarity

 if neg > pos:
   negative_list.append(tweet.text)
   negative += 1
 #negative_list.append(tweet.text)

 elif pos > neg:
   positive_list.append(tweet.text)
   positive += 1

 elif pos == neg:
   neutral_list.append(tweet.text)
   neutral += 1


positive = percentage(positive, noOfTweet)
negative = percentage(negative, noOfTweet)
neutral = percentage(neutral, noOfTweet)
polarity = percentage(polarity, noOfTweet)
positive = format(positive, '.1f')
negative = format(negative, '.1f')
neutral = format(neutral,'.1f')
```

```
Please enter keyword or hashtag to search:covid
Please enter how many tweets to analyze:100
```

In [17]:
```
tweet_list = pd.DataFrame(tweet_list)
neutral_list = pd.DataFrame(neutral_list)
negative_list = pd.DataFrame(negative_list)
positive_list = pd.DataFrame(positive_list)
print("total number:",len(tweet_list))
print("positive number:",len(positive_list))
print("negative number:", len(negative_list))
print("neutral number: ",len(neutral_list))
```

```
total number: 100
positive number: 19
negative number: 26
neutral number:  55
```

In [18]:
```
filename = 'twitter_data_analysis'+(datetime.datetime.now().strftime("%Y-%m-%d-%H"))+'.csv'
```

In [60]:
```
api =tweepy.API(auth,wait_on_rate_limit=True)
with open (filename, 'a+', newline='') as csvFile:
    csvWriter = csv.writer(csvFile)

    for tweet in tweepy.Cursor(api.search_tweets, q='covid', lang = 'en', count=10).items():
     tweets_encoded = tweet.text.encode('utf-8')
     tweets_decoded = tweets_encoded.decode('utf-8')
     csvWriter.writerow([datetime.datetime.now().strftime("%Y-%m-%d  %H:%M"),
                         tweet.id, tweets_decoded,tweet.retweet_count,tweet.favorite_count,tweet.user.screen_name,
                         tweet.created_at, tweet.geo, tweet.place.name if tweet.place else None, tweet.coordinates,
                         tweet._json["user"]["location"]])
```

```
Rate limit reached. Sleeping for: 810
Rate limit reached. Sleeping for: 807
Rate limit reached. Sleeping for: 811
Rate limit reached. Sleeping for: 809
```

In [68]:
```
def get_twitter_client():
    """
    @return:
        - the client to access the authentification API
    """
    auth = get_twitter_auth()
    client = tweepy.API(auth, wait_on_rate_limit=True)
    return client
```

In [71]:
```
Covid = get_tweets_from_user("WHO")
print("Data Shape: {}".format(Covid.shape))
```

```
Data Shape: (3198, 6)
```

In [73]:
```
Covid.head()
```

Out[73]:

	date	author	twitter_name	text	number_of_likes	number_of_retweets
0	2022-03-05 15:00:50+00:00	World Health Organization (WHO)	WHO	Dr @mvankerkhove explains what studies have sh...	104	48
1	2022-03-05 13:31:11+00:00	World Health Organization (WHO)	WHO	#Madagascar has been hit by 4 extreme weather ...	148	55
2	2022-03-05 13:01:37+00:00	World Health Organization (WHO)	WHO	In North-Eastern #Nigeria🇳🇬 WHO provided essen...	81	13
3	2022-03-05 12:30:47+00:00	World Health Organization (WHO)	WHO	RT @WHOUkraine: Health Cluster in #Ukraine, le...	0	92
4	2022-03-05 12:30:44+00:00	World Health Organization (WHO)	WHO	RT @WHOUkraine: Кластер охорони здоров'я в #Ук...	0	28

In [74]:
```
Covid_news = get_tweets_from_user("COVIDNewsByMIB")
```

In [75]:
```
Covid_news.head()
```

Out[75]:

	date	author	twitter_name	text	number_of_likes	number_of_retweets
0	2022-03-04 07:11:32+00:00	#IndiaFightsCorona	COVIDNewsByMIB	#IndiaFightsCorona : \n\n➡️13,450 patients rec...	5	1
1	2022-03-04 07:11:31+00:00	#IndiaFightsCorona	COVIDNewsByMIB	#IndiaFightsCorona:\n\n➡️178.29 crore vaccine ...	8	1
2	2022-03-04 07:11:29+00:00	#IndiaFightsCorona	COVIDNewsByMIB	#IndiaFightsCorona: \n\n➡️Active caseload stan...	6	0
3	2022-03-04 07:11:27+00:00	#IndiaFightsCorona	COVIDNewsByMIB	#IndiaFightsCorona:\n\n#COVID19 UPDATE (As on ...	20	6
4	2022-03-04 06:56:11+00:00	#IndiaFightsCorona	COVIDNewsByMIB	#IndiaFightsCorona:\n\n📍More than 9 Lakh (9,23...	10	2

In [76]:
```
print("Data Shape: {}".format(Covid_news.shape))
```

```
Data Shape: (3182, 6)
```

Our next step will be to design a filter that would remove tweets based on particular keywords. For our project, we need to find tweets that include the words we're looking for. Use particular phrases like "corona," "coronavirus," and "covid19" to narrow your search, as illustrated in the example below. As a result of the aforementioned, you'll get raw data with time stamps and a few Twitter ids. To use the data in your project, you must first clean it. Tokenization, text parsing, word elimination, and other techniques are all part of the process of cleaning up the data. Cleaning the data is a topic for a future post.
This format can be used to obtain data for a variety of different types of projects, from political campaigns to environmental concerns to the stock market to legos. The power of Twitter may be used to analyse a wide range of situations.
Link to verify the obtained data extracted in file as stated above:
https://drive.google.com/file/d/1JyXheOVKcXQ-DuRMs$_t$*Zi*8*ZBkGWKSdvH*/*view*?*usp* = *sharing*

7 Basic Text Pre-Processing

An extensive list of necessary procedures is included in the various stages of text production. Tasks such as these are included in this category.

1. Stop-Word Removal : Because these words have no specific meaning in English, we can eliminate them from our vocabulary to make it smaller. Stop-words like "a," "an," "the," "as," "in," and "on" can be eliminated from our vocabulary to reduce the amount of our vocabulary.
2. Lower Casing : Because it is possible that the case of a word will have no bearing on the solution, it is best to use all lower case. At the same time, we're reducing the size of our vocabularies.
3. Stemming : It is possible to stem a word by removing all of the suffixes so that all of the term's variants can be rendered in the same form (for example, "walk" and "walking" are both reduced to "walk").
4. Tokenization: NLP software commonly breaks down words and sentences into "tokens."

In [15]:
```
df1.drop(columns=['id'], inplace=True)
```

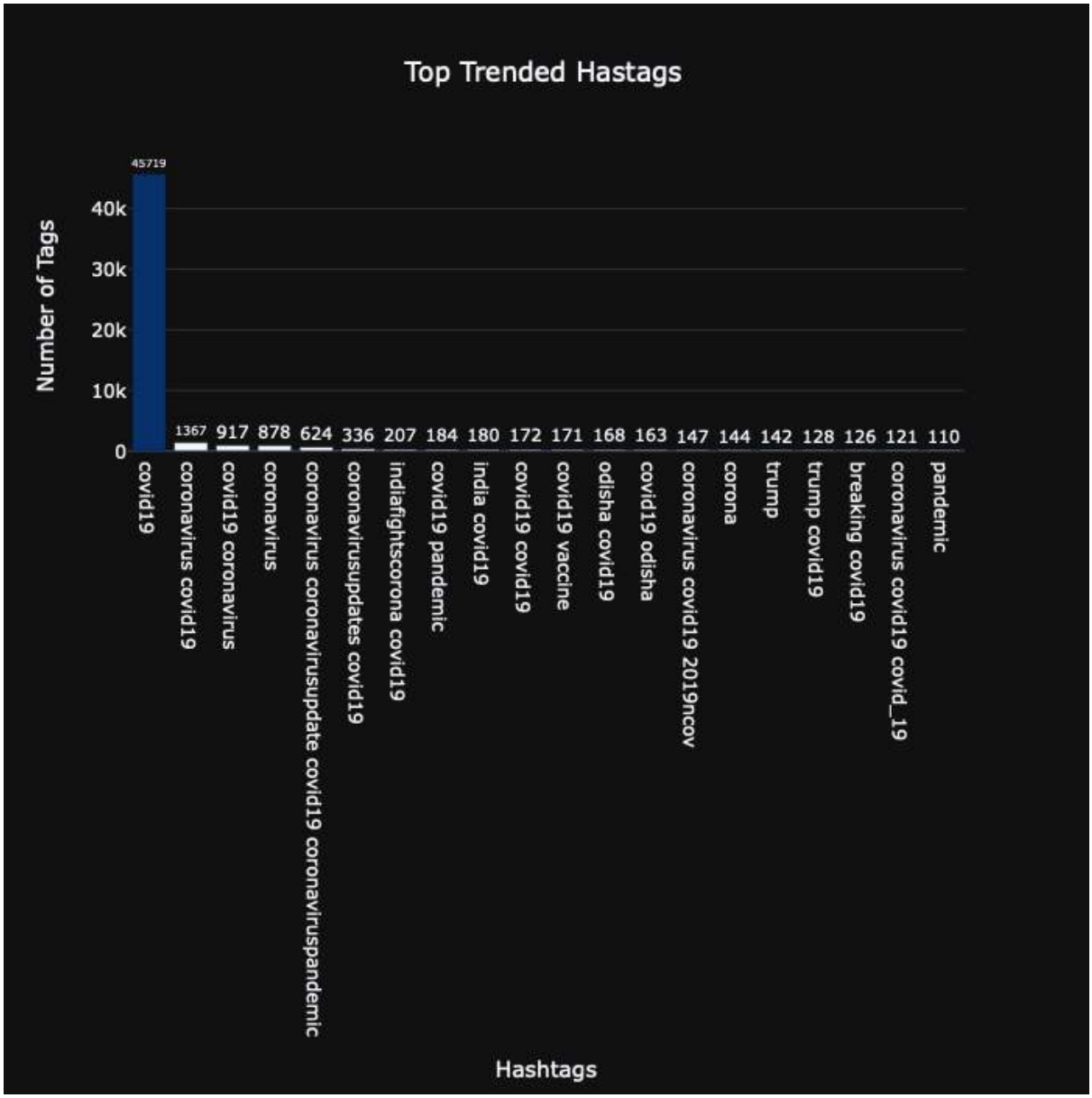
In [16]:
```
df1 = df1.drop_duplicates('text')
print(df.shape)
```

```
(16180, 15)
```

In [17]:
```
def clean_tweet_text(text):
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#', '', text)
    text = re.sub(r'RT[\s]+', '', text)
    text = re.sub(r'https?:\/\/\S+', '', text)
    text = text.lower()
    return text

# the following line makes use of an apply function-- it will call clean_tweet_text on every element in the 'text' column
df1['text'].transform(clean_tweet_text)
df1.head()
```

Out[17]:

	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text	hashtags	source	re
0	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	2009-04-08 17:52:46	405	1692	3247	False	2020-12-20 06:06:44	Same folks said daikon paste could treat a cyt...	['PfizerBioNTech']	Twitter for Android	
2	eli🇺🇦🇪🇺👌	Your Bed	heil, hydra 😩🙄	2020-06-25 23:30:28	10	88	155	False	2020-12-12 20:33:45	#coronavirus #SputnikV #AstraZeneca #PfizerBio...	['coronavirus', 'SputnikV', 'AstraZeneca', 'Pf...	Twitter for Android	
6	Gunther Fehlinger	Austria, Ukraine and Kosovo	End North Stream 2 now - the pipeline of corru...	2013-06-10 17:49:22	2731	5001	69344	False	2020-12-12 20:06:00	it is a bit sad to claim the fame for success ...	['vaccination']	Twitter Web App	
9	Ch.Amjad Ali	Islamabad	#ProudPakistani #LovePakArmy #PMIK @insafiansp...	2012-11-12 04:18:12	671	2368	20469	False	2020-12-12 19:30:33	#CovidVaccine \n\nStates will start getting #C...	['CovidVaccine', 'COVID19Vaccine', 'US', 'paku...	Twitter Web App	
10	Tamer Yazar	Turkey-Israel	Im Market Analyst, also Editor... working (fre...	2009-09-17 16:45:16	1302	78	339	False	2020-12-12 19:29:07	while deaths are closing in on the 300,000 mar...	['PfizerBioNTech', 'Vaccine']	Twitter Web App	



8 Models Deployed

8.1 TextBlob

The Textblob library enables us to swiftly and efficiently feed a statement through and identify its emotion, which includes its polarity, which ranges from negative (-1) to positive (1) statements, as well as its subjectivity, which considers the sentence's factual (0) versus opinionated (1) quality.

- Noun phrase extraction
- Part-of-speech tagging
- Sentiment analysis
- Classification (Naive Bayes, Decision Tree)
- Tokenization (splitting text into words and sentences)
- Word and phrase frequencies
- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction
- Add new models or languages through extensions
- WordNet integration

Finally, we may begin processing these tweets' sentiment analysis. This will be accomplished by iterating through the dataframe and performing the following steps: Textblobing each tweet string Columnize the resulting polarity and subjectivity. Create a new column called 'Sentiment,' in which we categorise tweets as Negative, Neutral, or Positive, depending on their polarity.

```
In [5]:
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
from textblob import TextBlob
from wordcloud import WordCloud
import plotly.express as px
```

```
In [7]:
df1 = pd.read_csv('covid_vaccinedata.csv', quotechar='"', delimiter=',').dropna()
df1
```

Out[7]:

	id	user_name	user_location	user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text	hashtags	source	retweets	favorites	is_retweet
0	1340539111971516416	Rachel Roh	La Crescenta-Montrose, CA	Aggregator of Asian American news; scanning di...	2009-04-08 17:52:46	405	1692	3247	False	2020-12-20 06:06:44	Same folks said daikon paste could treat a cyt...	['PfizerBioNTech']	Twitter for Android	0	0	False
2	1337858199140118533	eli	Your Bed	hell, hydra	2020-06-25 23:30:28	10	88	155	False	2020-12-12 20:33:45	#coronavirus #SputnikV #AstraZeneca #PfizerBio...	['coronavirus', 'SputnikV', 'AstraZeneca', 'Pf...	Twitter for Android	0	0	False
6	1337851216875606579	Gunther Fehlinger	Austria, Ukraine and Kosovo	End North Stream 2 now - the pipeline of corru...	2013-06-10 17:49:22	2731	5001	69344	False	2020-12-12 20:06:00	it is a bit sad to claim the fame for success ...	['vaccination']	Twitter Web App	0	4	False
9	1337842295857623042	Ch Amjad Ali	Islamabad	#ProudPakistani #LovePakArmy #PMIK @insaflansp...	2012-11-12 04:18:12	671	2368	20469	False	2020-12-12 19:30:33	#CovidVaccine \n\nStates will start getting #C...	['CovidVaccine', 'COVID19Vaccine', 'US', 'peka...	Twitter Web App	0	0	False
10	1337841934170255365	Tamer Yazar	Turkey-Israel	Im Market Analyst, also Editor,,, working (fre...	2009-09-17 16:45:16	1302	78	339	False	2020-12-12 19:29:07	while deaths are closing in on the 300,000 mar...	['PfizerBioNTech', 'Vaccine']	Twitter Web App	0	0	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
27802	1368226947622653442	Andy The legally and popularly elected Gardener.	Untidied States	merciless Infj door slammer.	2019-01-01 23:12:42	93	83	9876	False	2021-03-06 15:48:19	If you want to know how much antivaxx disinfor...	['SputnikV']	Twitter for iPhone	0	1	False
27804	1368224770995675139	Sarwar	E	W pirsarwarras	2009-07-01 00:18:11	1685	1439	6268	False	2021-03-06 15:39:40	#BREAKING \n#Venezuela \nVenezuelan President ...	['BREAKING', 'Venezuela', 'SputnikV']	Twitter Web App	0	2	False
27806	1368224272267829256	Workout Solutions	Toronto, Canada and Worldwide	George Tsanis – Workout Solutions Health and F...	2010-09-20 17:01:08	1164	172	1368	False	2021-03-06 15:37:41	Moscow Russia everything is open business as u...	['COVID19']	Twitter for iPhone	1	2	False

In [12]:
```
df2
```

Out[12]:

	0	1	2	3	4	5	6
0	05/03/22 19:54	1.500120e+18	RT @theseoulstory: SEVENTEEN S.Coups, Dokyeom ...	779	0	Shaaaxiao06	2022-03-05 14:24:31+00:00
1	05/03/22 19:54	1.500120e+18	RT @rameshlaus: Released during the driest exa...	1591	0	ViswasamKishor1	2022-03-05 14:24:31+00:00
2	05/03/22 19:54	1.500120e+18	RT @ErinSandersNP: @ahandvanish 1/ Patients wi...	45	0	LaraMcRitchie	2022-03-05 14:24:31+00:00
3	05/03/22 19:54	1.500120e+18	nah haven't felt this shitty in a long time fu...	0	0	danascllv	2022-03-05 14:24:31+00:00
4	05/03/22 19:54	1.500120e+18	BBC News - Covid in Scotland: Unvaccinated wom...	0	0	Blairmenachi	2022-03-05 14:24:30+00:00
...	...	...	...	...	...	...	...
7012	2022-03-05 20:43	1.500109e+18	Covid safety and Spring Break for kids\n\nhttp...	2	1	KTSMtv	2022-03-05 14:00:18+00:00
7013	2022-03-05 20:43	1.500109e+18	Characteristics and Outcomes of Hospitalized P...	0	0	genomeorganizer	2022-03-05 14:00:18+00:00
7014	2022-03-05 20:43	1.500109e+18	RT @RosaZambonini: My dad was taken to hospita...	63	0	DylansNanna	2022-03-05 14:00:17+00:00
7015	2022-03-05 20:43	1.500109e+18	RT @ABC: While much of the country is relieved...	25	0	MajamboKE	2022-03-05 14:00:17+00:00
7016	2022-03-05 20:43	1.500109e+18	RT @Jay_Beecher: I knew I'd end up blocking hu...	16	0	caterita2008	2022-03-05 14:00:17+00:00

7017 rows × 22 columns

In [13]:
```
df1.info()
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16180 entries, 0 to 27808
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                16180 non-null  int64
 1   user_name         16180 non-null  object
 2   user_location     16180 non-null  object
 3   user_description  16180 non-null  object
 4   user_created      16180 non-null  object
 5   user_followers    16180 non-null  int64
 6   user_friends      16180 non-null  int64
 7   user_favourites   16180 non-null  int64
 8   user_verified     16180 non-null  bool
 9   date              16180 non-null  object
 10  text              16180 non-null  object
 11  hashtags          16180 non-null  object
 12  source            16180 non-null  object
 13  retweets          16180 non-null  int64
 14  favorites         16180 non-null  int64
 15  is_retweet        16180 non-null  bool
dtypes: bool(2), int64(6), object(8)
memory usage: 1.9+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7017 entries, 0 to 7016
Data columns (total 22 columns):
```

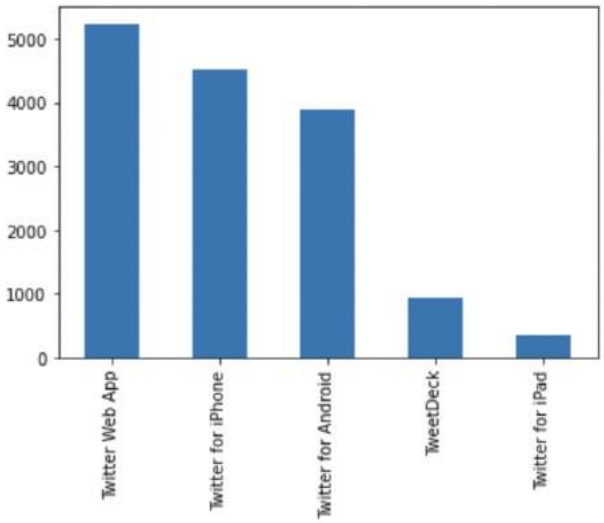
In [19]:
```
df1.describe()
```

Out[19]:

	user_followers	user_friends	user_favourites	retweets	favorites
count	1.617000e+04	16170.000000	16170.000000	16170.000000	16170.000000
mean	1.719752e+05	1518.035931	14619.516079	4.241187	18.428386
std	1.141814e+06	7640.144298	37054.083845	45.617605	246.240119
min	0.000000e+00	0.000000	0.000000	0.000000	0.000000
25%	2.180000e+02	170.000000	426.000000	0.000000	0.000000
50%	9.470000e+02	492.500000	2377.000000	0.000000	1.000000
75%	4.284000e+03	1372.750000	11847.250000	1.000000	5.000000
max	1.486666e+07	380428.000000	686342.000000	2360.000000	22815.000000

In [20]:
```
df1['source'].value_counts().head(n=5).plot.bar()
```

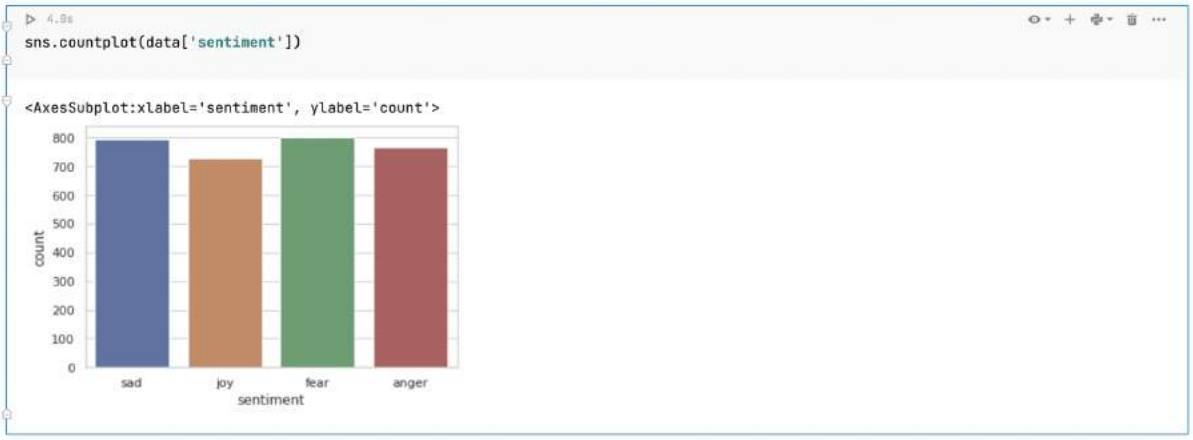
Out[20]:
```
<AxesSubplot:>
```



TextBlob's characteristics include the following:

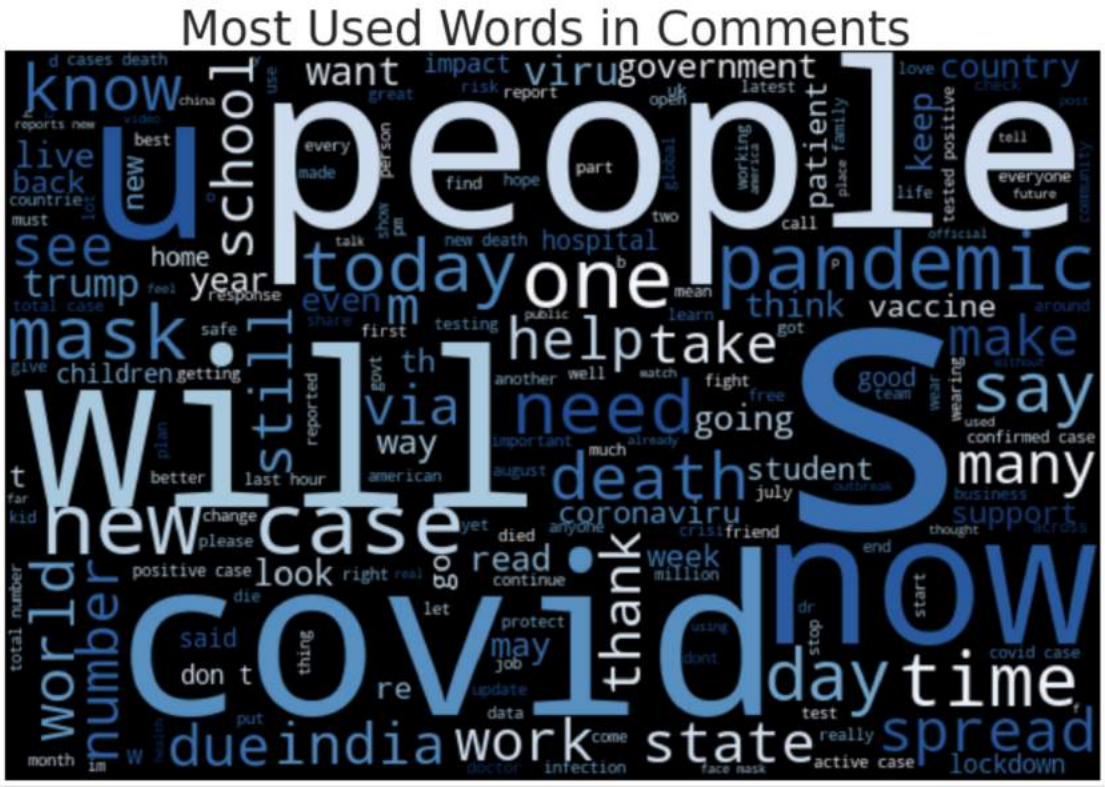
1. Tokenization: This is a process that divides a lengthy paragraph into words and phrases. A token is merely a word. To begin, we import the TextBlob object, TextBlob module, and send the phrase to the TextBlob object for tokenization.
2. Lemmatization: This is the process of tracing a word back to its dictionary definition. To use it with TextBlob, we must access the Word object from the TextBlob module, supply the word as an argument, and call the lemmatize method.
   For instance, we lemmatize the terms "apples," "me-dia," and "greater." Retrieve "apple", "medium", and "excellent" from the output.
3. Tags for parts of speech
   It is the process of recognising a text document's structural parts, such as verbs, nouns, adjectives, and adverbs.
   The interface for utilising the point-of-sale tagging capability.

4. Extraction of Noun Phrases
   It entails the extraction of phrases containing nouns from a particular context. Allow me to demonstrate this feature using the following example.
   The result will list all of the document's nouns.
5. Tweepy: An open-source Python repository for talking with and utilising the Twitter platform's API. Tweepy includes numerous features, including the following: 1. Acquire tweets. 2. Creating and deleting tweets. 3. Subscribing to and unsubscribing from users



```
fig, (ax2) = plt.subplots(1,1,figsize=[17, 10])
wordcloud2 = WordCloud(background_color='black',colormap="Blues",
                       width=600,height=400).generate(" ".join(covid['refine_text']))

ax2.imshow(wordcloud2,interpolation='bilinear')
ax2.axis('off')
ax2.set_title('Most Used Words in Comments ',fontsize=35)
```

```
Text(0.5, 1.0, 'Most Used Words in Comments ')
```

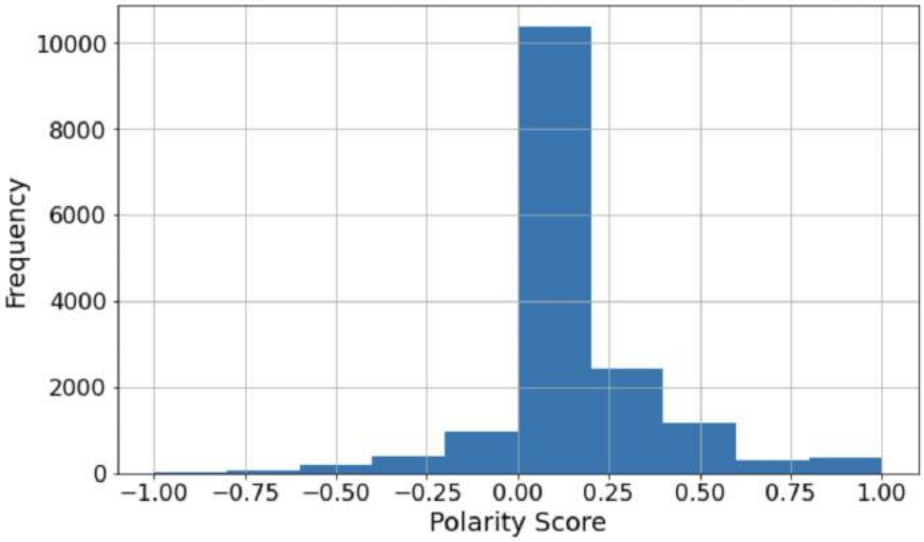



```
print("Tags")
print(blob.tags, end="\n\n")

print("Noun Phrases")
print(blob.noun_phrases, end="\n\n")

print("Words")
print(blob.words, end="\n\n")

print("Sentences")
print(blob.sentences, end="\n\n")
```

```
Tags
[('Today', 'NN'), ('Texas', 'NNP'), ('surpassed', 'VBD'), ('8', 'CD'), ('million', 'CD'), ('vaccinations', 'NNS'), ('It', 'PRP'), ('was', 'VBD'), ('just', 'RB'), ('4', 'CD'), ('days', 'NNS'), ('ago', 'RB')
, ('that', 'IN'), ('we', 'PRP'), ('passed', 'VBD'), ('7', 'CD'), ('million', 'CD'), ('The', 'DT'), ('positivity', 'NN'), ('rate', 'NN'), ('dropped', 'VBD'), ('even', 'RB'), ('lower', 'JJR'), ('to', 'TO'),
('6.08', 'CD'), ('%', 'NN'), ('the', 'DT'), ('lowest', 'JJS'), ('since', 'IN'), ('May', 'NNP'), ('31st', 'CD'), ('Hospitalizations', 'NNS'), ('went', 'VBD'), ('down', 'RB'), ('again', 'RB'), ('also', 'RB')
, ('Great', 'NNP'), ('job', 'NN'), ('Texans', 'NNPS'), ('Vaccines', 'NNS'), ('are', 'VBP'), ('always', 'RB'), ('voluntary', 'JJ'), ('never', 'RB'), ('forced', 'VBN')]

Noun Phrases
['texas', 'positivity rate', 'may', 'hospitalizations', 'great job', 'vaccines']

Words
['Today', 'Texas', 'surpassed', '8', 'million', 'vaccinations', 'It', 'was', 'just', '4', 'days', 'ago', 'that', 'we', 'passed', '7', 'million', 'The', 'positivity', 'rate', 'dropped', 'even', 'lower', 'to
', '6.08', 'the', 'lowest', 'since', 'May', '31st', 'Hospitalizations', 'went', 'down', 'again', 'also', 'Great', 'job', 'Texans', 'Vaccines', 'are', 'always', 'voluntary', 'never', 'forced']

Sentences
[Sentence("
Today Texas surpassed 8 million vaccinations."), Sentence("It was just 4 days ago that we passed 7 million."), Sentence("The positivity rate dropped even lower--to 6.08%--the lowest since May 31st."), Sent
ence("Hospitalizations went down again also."), Sentence("Great job Texans!"), Sentence("Vaccines are always voluntary, never forced.")]
```

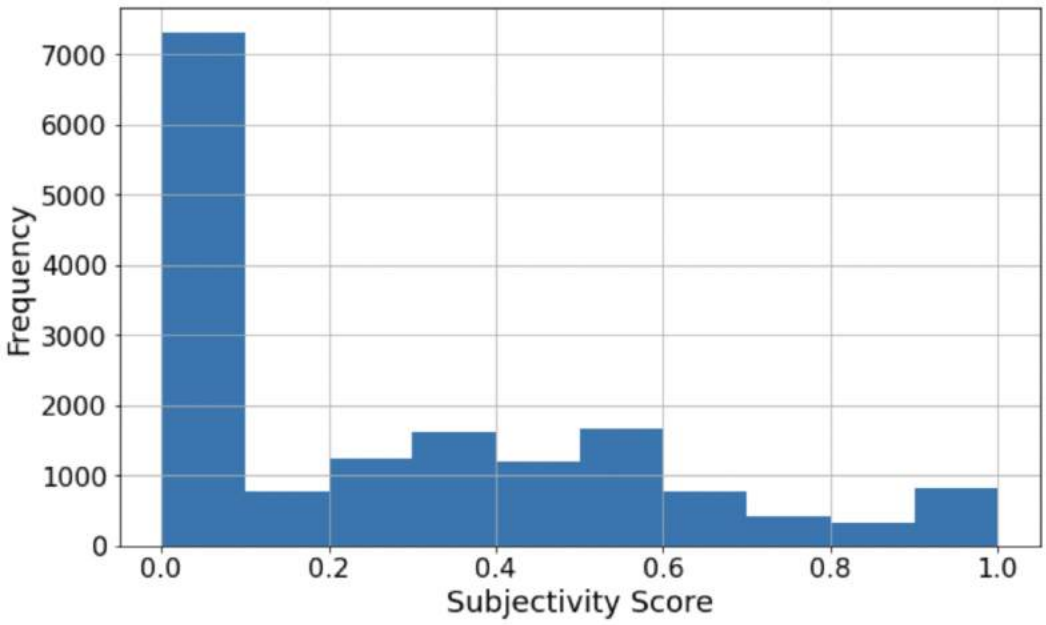
In [31]:

```
fig = plt.figure(figsize=(10, 6))
df1['polarity'].hist()
plt.xlabel('Polarity Score', fontsize=18)
plt.ylabel('Frequency', fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
fig.savefig("polarity_hist.png")
```



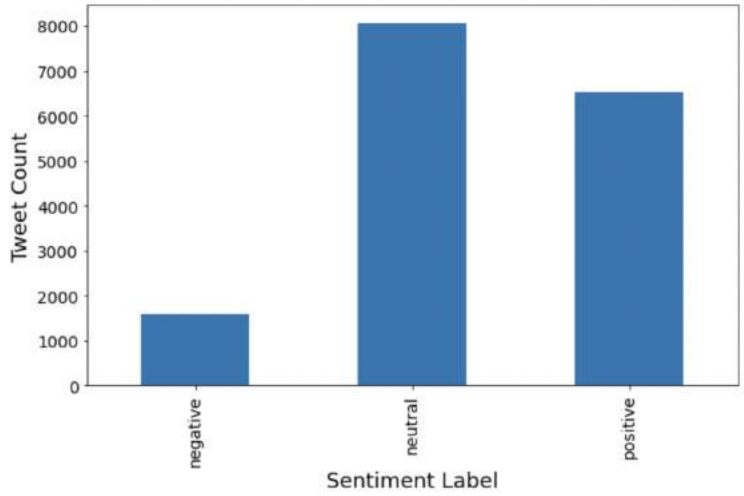
In [33]:

```
fig = plt.figure(figsize=(10, 6))
df1['subjectivity'].hist()
plt.xlabel('Subjectivity Score', fontsize=18)
plt.ylabel('Frequency', fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
fig.savefig("subjectivity_hist.png")
```



In [41]:

```
criteria = [df1['polarity'].between(-1, -0.01), df1['polarity'].between(-0.01, 0.01), df1['polarity'].between(0.01, 1)]
values = ['negative', 'neutral', 'positive']
df1['sentiment'] = np.select(criteria, values, 0)

# plot sentiment counts
fig = plt.figure(figsize=(10, 6))
df1['sentiment'].value_counts().sort_index().plot.bar()
plt.xlabel('Sentiment Label', fontsize=18)
plt.ylabel('Tweet Count', fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
plt.tight_layout()
fig.savefig("sentiment_value_counts", bbox_inches='tight');
```



In [42]:

```
timeline = df1.groupby(['date']).agg(np.nanmean).reset_index()
timeline['count'] = df1.groupby(['date']).count().reset_index()['retweets']
timeline = timeline[['date', 'count', 'polarity', 'retweets', 'favorites', 'subjectivity']]
timeline["polarity"] = timeline["polarity"].astype(float)
timeline["subjectivity"] = timeline["subjectivity"].astype(float)
timeline
```

Out[42]:

	date	count	polarity	retweets	favorites	subjectivity
0	2020-12-12	38	0.034952	0.631579	2.868421	0.227801
1	2020-12-13	52	0.055937	1.942308	7.500000	0.255696
2	2020-12-14	83	0.116874	4.698795	29.204819	0.267803
3	2020-12-15	87	0.094041	1.022989	7.333333	0.258078
4	2020-12-16	78	0.104020	1.000000	5.115385	0.248872
...	...	...	...	...	...	...
88	2021-03-10	465	0.098784	2.481720	9.987097	0.246851
89	2021-03-11	516	0.081946	2.232558	9.248062	0.253772
90	2021-03-12	467	0.108128	1.391863	8.509636	0.259496
91	2021-03-13	315	0.088139	3.161905	11.838095	0.290822
92	2021-03-14	83	0.091934	1.349398	3.060241	0.256714

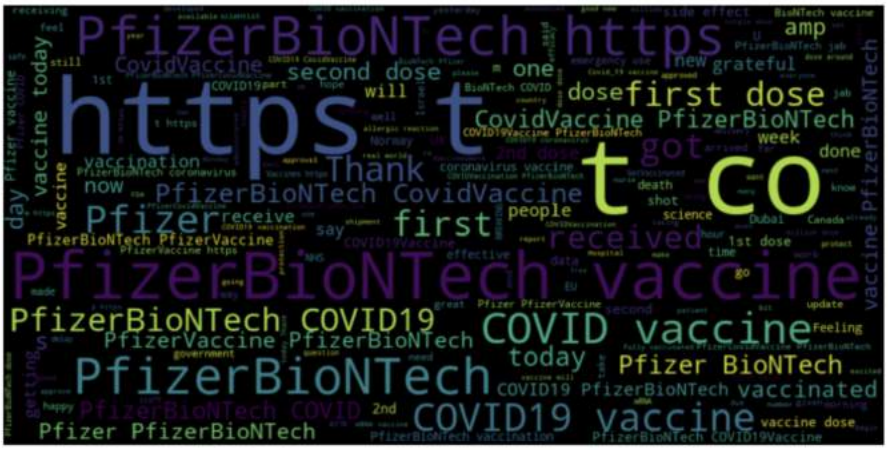
93 rows × 6 columns

```
[nltk_data]     /Users/ritikasingh/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/ritikasingh/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

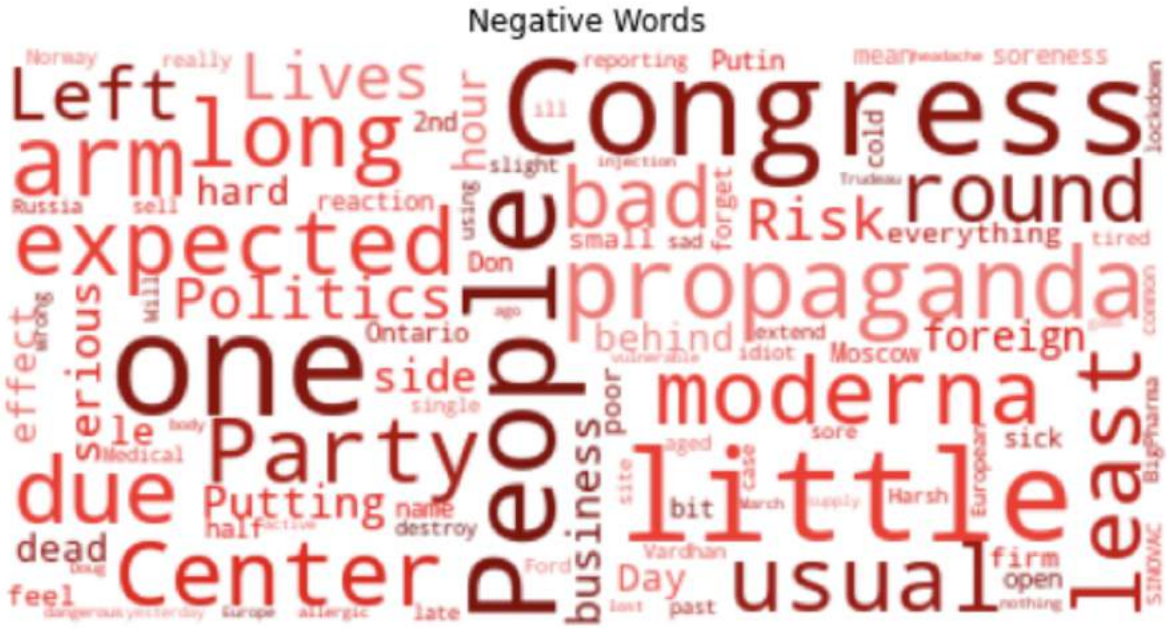
In [64]:

```
tweet_df = pfizy_df
words = ' '.join([word for word in tweet_df['text']])
word_cloud = WordCloud(width=1000, height=500, random_state=20, max_font_size=120).generate(words)

fig, ax = plt.subplots(figsize=(12,6))
plt.imshow(word_cloud, interpolation='bilinear')
plt.axis('off')
```

Out[64]: (-0.5, 999.5, 499.5, -0.5)

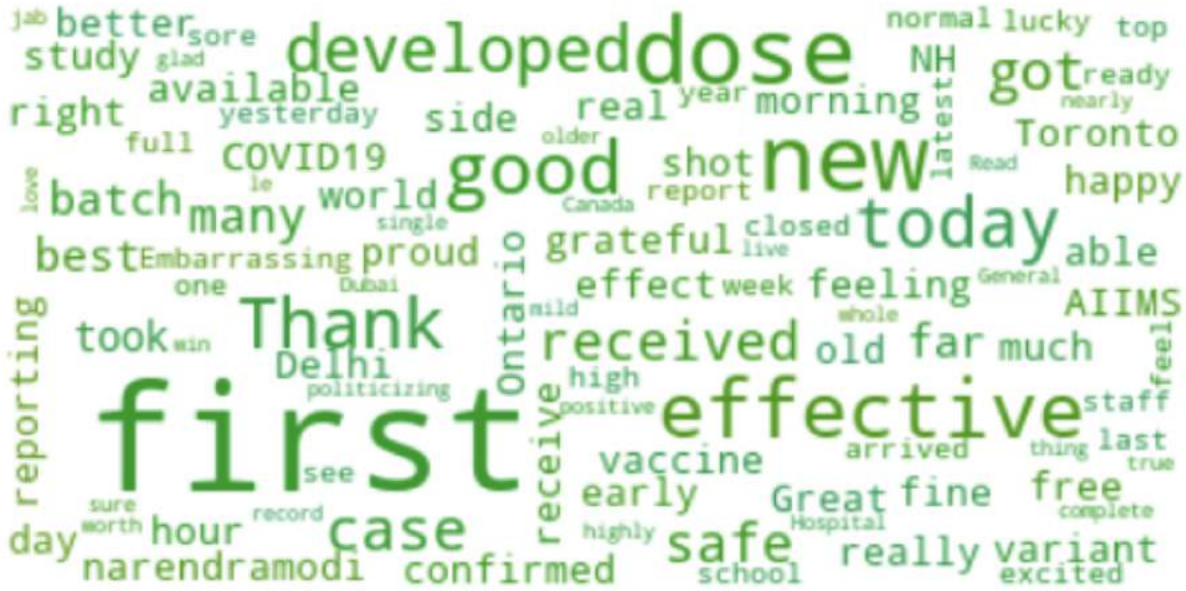


Word clouds of COVID19 general tweets using TextBlob



Neutral Words

Positive Words



In [66]:

```
def get_smart_clouds(df):

    neg_doc = flatten_list(df[df['sentiment']=='negative']['words'])
    neg_doc = [x for x in neg_doc if is_acceptable(x)]

    pos_doc = flatten_list(df[df['sentiment']=='positive']['words'])
    pos_doc = [x for x in pos_doc if is_acceptable(x)]

    neu_doc = flatten_list(df[df['sentiment']=='neutral']['words'])
    neu_doc = [x for x in neu_doc if is_acceptable(x)]

    # Clean all the documents
    neg_doc_clean = clean_document(neg_doc)
    neu_doc_clean = clean_document(neu_doc)
    pos_doc_clean = clean_document(pos_doc)

    # Combine classes B and C to compare against A (ex. "positive" vs "non-positive")
    top_neg_words = get_log_likelihood(neg_doc_clean, flatten_list([pos_doc_clean, neu_doc_clean]))
    top_neu_words = get_log_likelihood(neu_doc_clean, flatten_list([pos_doc_clean, neg_doc_clean]))
    top_pos_words = get_log_likelihood(pos_doc_clean, flatten_list([neu_doc_clean, neg_doc_clean]))

    # Generate syntetic a corpus using our loglikelihood values
    neg_doc_final = get_scaled_list(top_neg_words)
    neu_doc_final = get_scaled_list(top_neu_words)
    pos_doc_final = get_scaled_list(top_pos_words)

    # Visualise our synthetic corpus
    fig = generate_word_clouds(neg_doc_final, neu_doc_final, pos_doc_final)
    return fig

# Convert string to a list of words
wordcloud_df = df1
wordcloud_df['words'] = wordcloud_df.text.apply(lambda x:re.findall(r'\w+', x ))
get_smart_clouds(wordcloud_df).savefig("sentiment_wordclouds.png", bbox_inches="tight")
```

TextBlob is a Python Natural Language Processing package (NLP). TextBlob made extensive use of the Natural Language ToolKit (NLTK) to accomplish its objectives. NLTK is a library that enables users to easily access a large number of lexical resources and perform categorization, classification, and a variety of other activities. TextBlob is a small toolkit that enables extensive textual data analysis and manipulation. A sentiment is characterised in lexicon-based techniques by its semantic direction and the strength of each word in the phrase. This necessitates the creation of a pre-defined vocabulary that categorises negative and positive words. In general, a text message is represented by a collection of words. After giving unique scores to each word, the ultimate sentiment is derived using some sort of pooling process, such as averaging the sentiments.

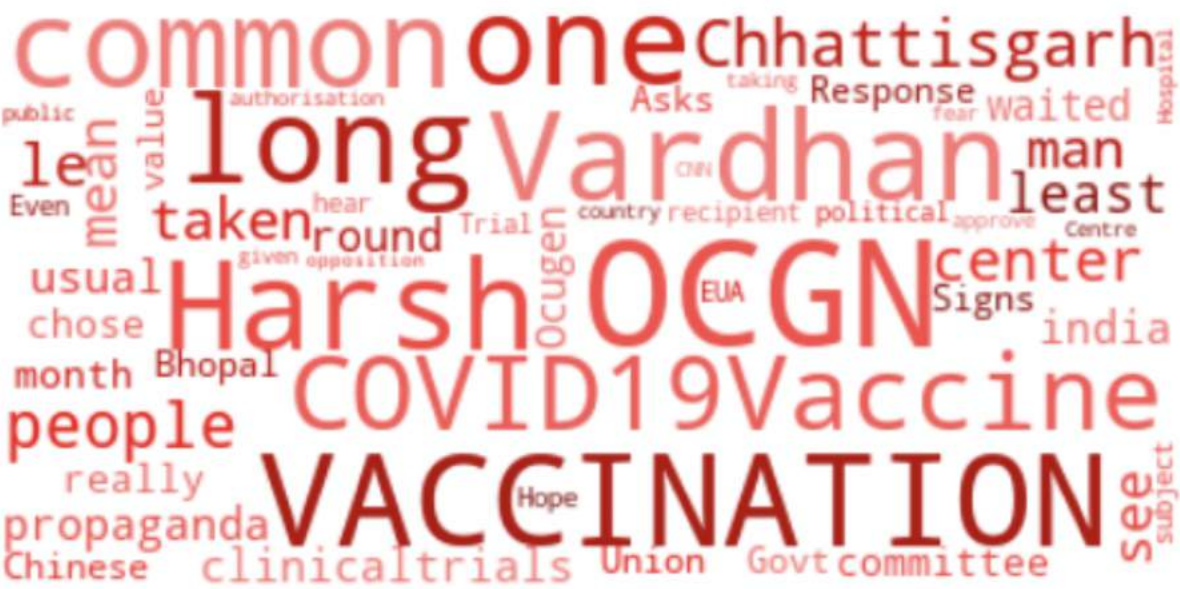
Word clouds of covaxin hashtags using TextBlob

In [54]:

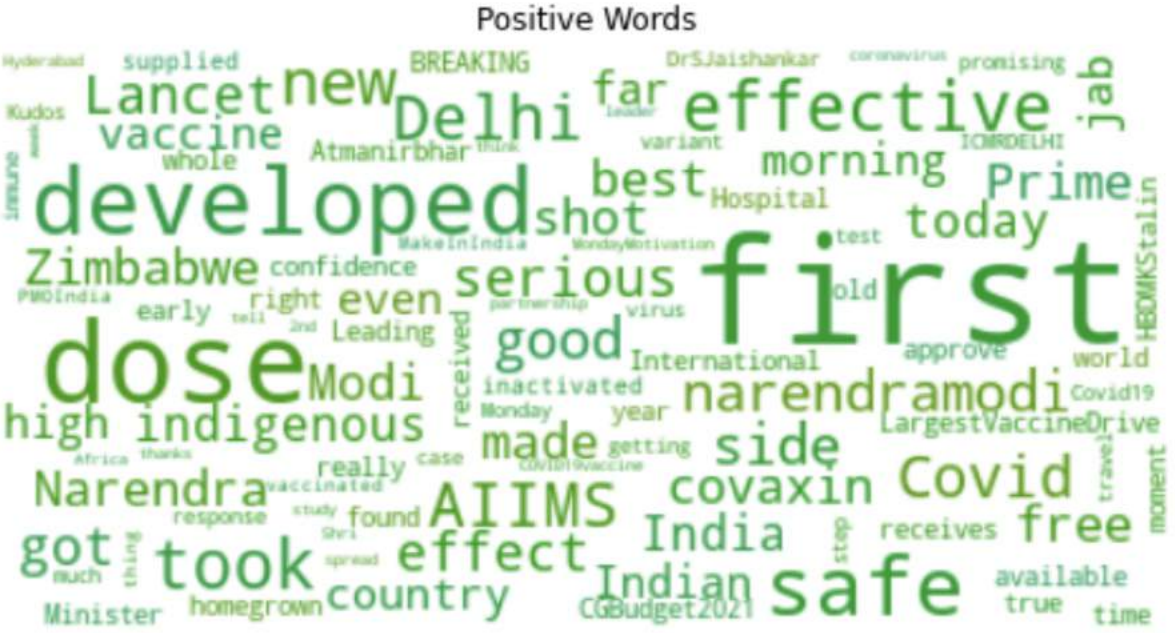
```
covaxin_df, covaxin_timeline = filter_by_vaccy(df1, ['covaxin'])
covaxin_df.sort_values(by='polarity', ascending=True).reset_index(drop=True).head(n=20)
```

hashtags	source	retweets	favorites	is_retweet	polarity	subjectivity	sentiment
['clinicaltrials', 'COVID19Vaccine', 'Covaxin']	Twitter Web App	0	0	False	-1.000000	1.000000	negative
['clinicaltrials', 'COVID19Vaccine']	Twitter Web App	0	0	False	-1.000000	1.000000	negative
['Covaxin']	Twitter for iPhone	0	1	False	-0.900000	1.000000	negative
['COVAXIN']	Twitter for Android	5	133	False	-0.800000	0.900000	negative
['covishield', 'Covaxin']	Twitter for iPhone	0	2	False	-0.700000	0.666667	negative
['COVAXIN']	Twitter for iPhone	0	5	False	-0.650000	0.700000	negative
['Covaxin', 'CoronaVaccine']	Twitter for iPhone	1	1	False	-0.600000	0.800000	negative
['Covaxin', 'CoronaVaccine']	Twitter for Android	0	2	False	-0.600000	0.900000	negative
['Covaxin']	Twitter Web App	0	2	False	-0.600000	1.000000	negative
['Covaxin']	Twitter for Android	1	2	False	-0.500000	0.700000	negative

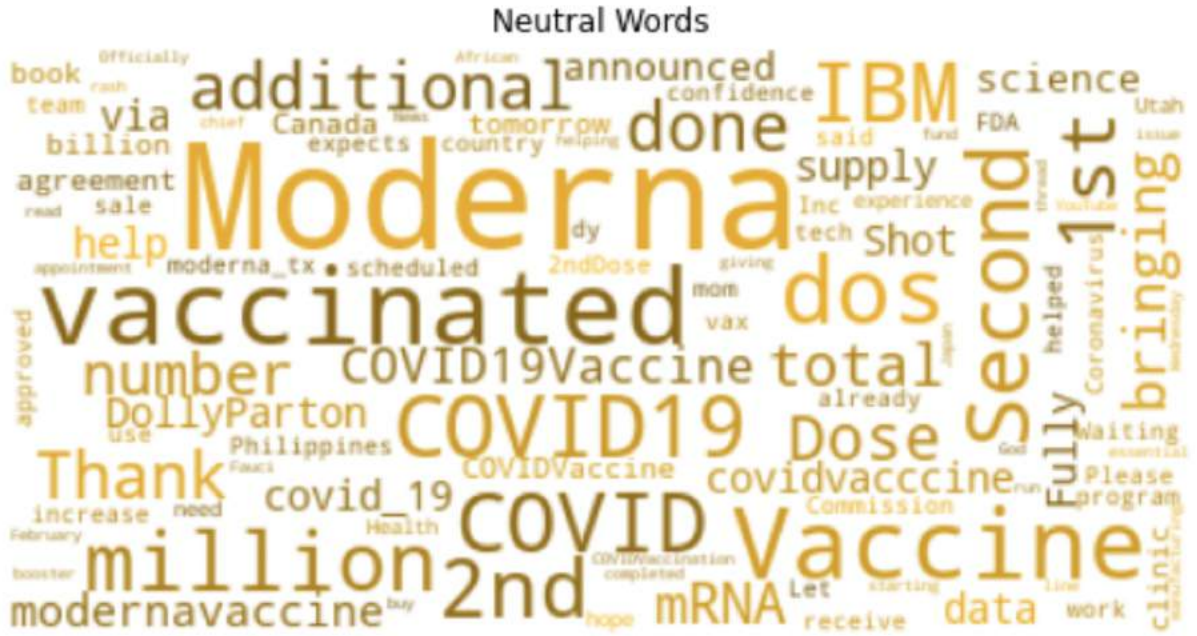
Negative Words

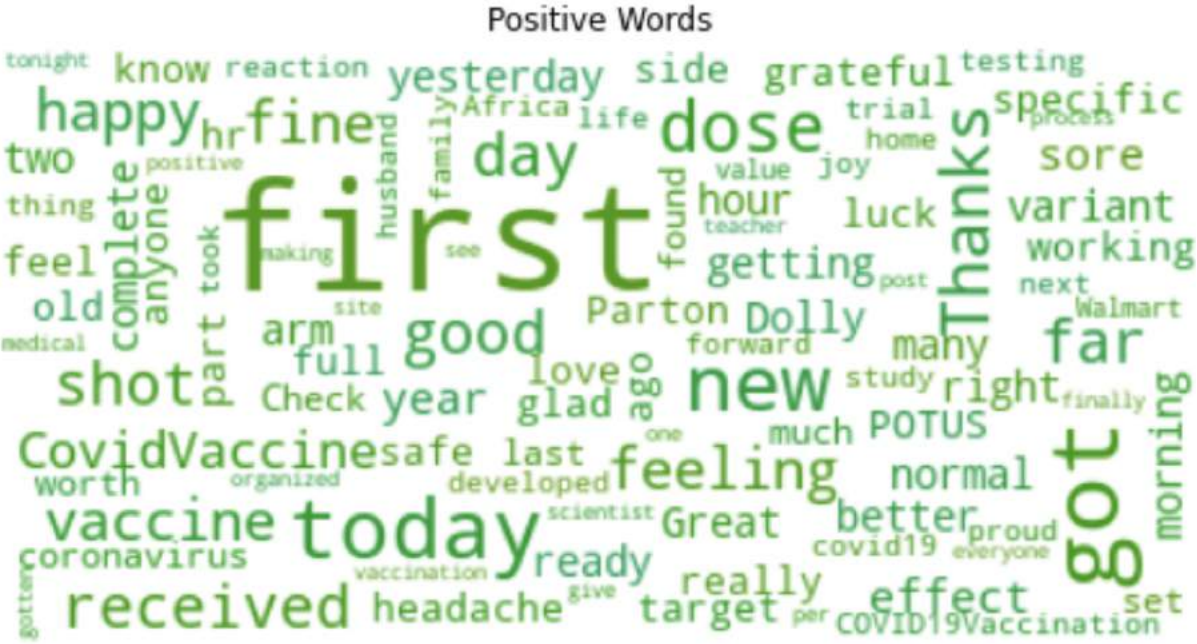


Neutral Words



Word clouds of moderna hashtags using TextBlob





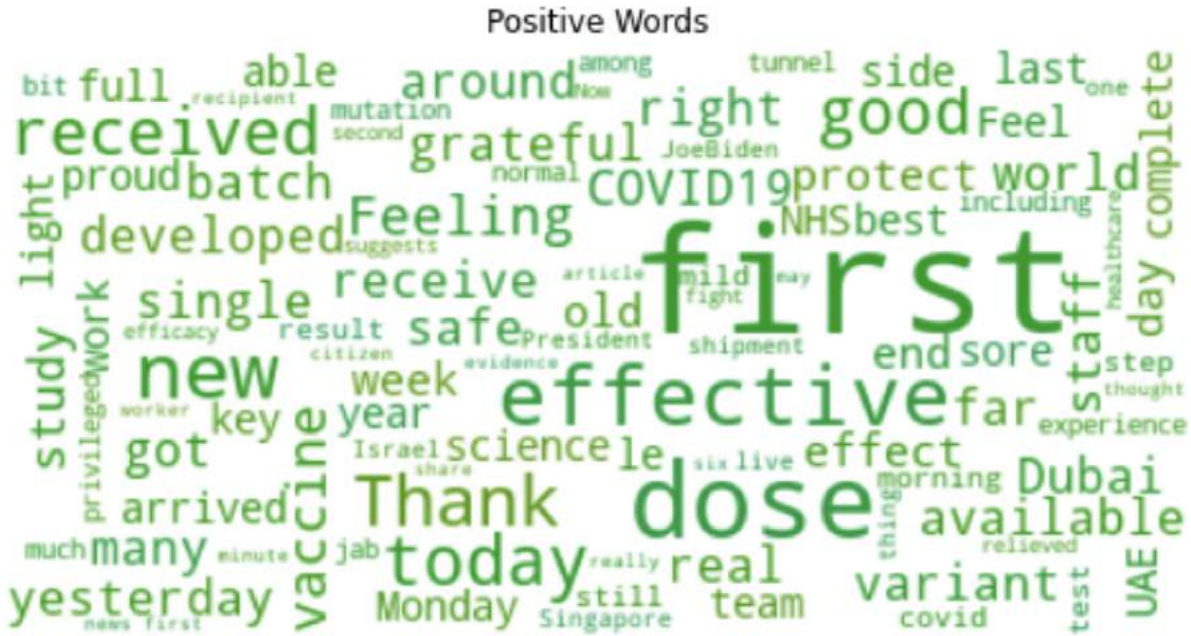
Word clouds of astrazeneca hashtags using TextBlob

In [53]:
```
astra_df, astra_timeline = filter_by_vaccy(df1, ['astrazeneca'])
astra_df.sort_values(by='polarity', ascending=True).reset_index(drop=True).head(n=20)
```

hashtags	source	retweets	favorites	is_retweet	polarity	subjectivity	sentiment
['AstraZeneca']	Twitter Web App	8	10	False	-0.350000	0.833333	negative
['Europe', 'EU']	Twitter for Android	1	0	False	-0.291667	0.541667	negative
['AstraZeneca', 'Canada']	Twitter for iPhone	0	2	False	-0.281250	0.788194	negative
['Malaysia', 'UK', 'AstraZeneca']	Twitter Web App	1	1	False	-0.200000	0.400000	negative
['AstraZeneca']	Twitter for iPhone	1	1	False	-0.166667	0.395833	negative
['IndiaPostUSA', 'AnthonyFauci']	IndiaPost	0	0	False	-0.166667	0.066667	negative
['AstraZeneca', 'Corona']	Twitter Web App	0	0	False	-0.150000	0.450000	negative


Negative Words
PfizerBioNtech
tired
mRNA
mean
shot
little
arm
due
people
reaction
2nd
small
dead
serious
soreness
Norway
hour
allergic
late
cold
Vaccine
Germany
bad
Will
slight
ago
bit
half
BorisJohnson
expected
going
deltoid
country
round
Day
feel
slightly
ingredient

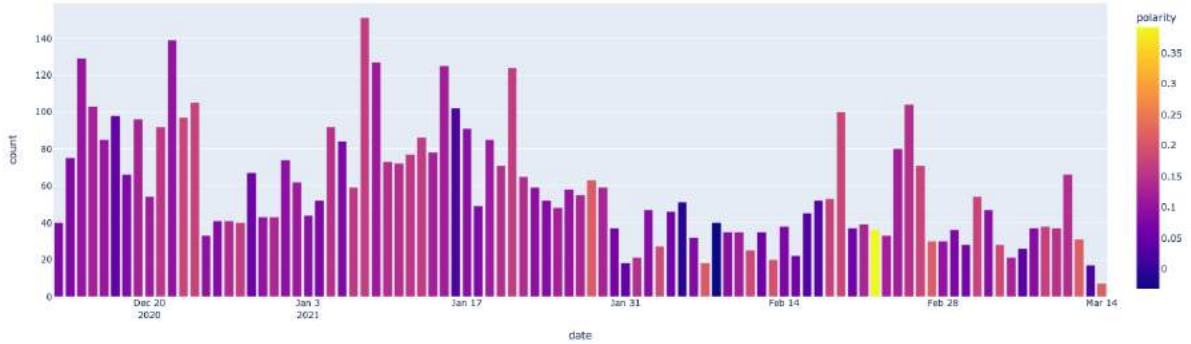

Neutral Words
Dose
FDA
use
Canada
dos
Second
CEO
coronavirus
million
vaccinated
Vaccine
Pfizer
emergency
CovidVaccine
Iran
North
Vaccination
approved
Sunday
PfizerVaccine
death
COVID
Health
1st
2nd
Done
said
announced
Qatar
PfizerBioNTech
BioNTech
PfizerCovidVaccine
approves
World



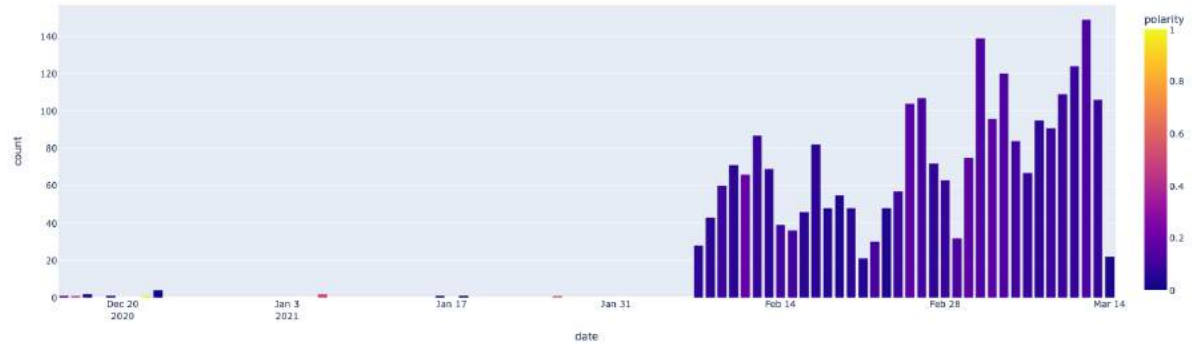
TextBlob returns the sentence's polarity and subjectivity.

Polarity is defined as the interval [-1,1], where -1 indicates a negative feeling and 1 indicates a positive sentiment. Negation words are polar opposites. Semantic labels in TextBlob aid in fine-grained analysis. For instance, emoticons, exclamation points, and emojis.

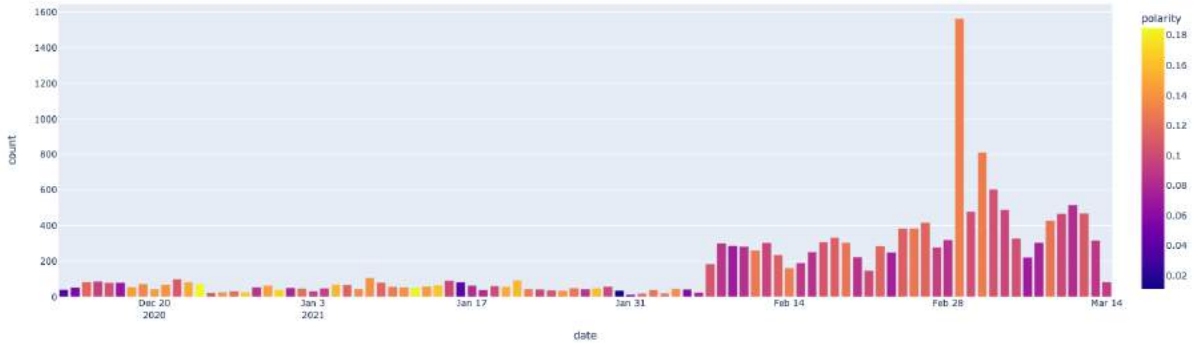
covaxin polarity



moderna polarity



astrazeneca polarity



Subjectivity is defined as the interval between [0,1]. Subjectivity quantifies the degree to which a document contains personal opinion and factual facts. Due of the text's greater subjectivity, it contains personal opinion rather than factual information. TextBlob now includes an additional parameter — intensity. Subjectivity is calculated in TextBlob by examining the 'intensity'. The intensity of a word influences whether it changes the following word. Adverbs are used as modifiers in English ('extremely good').

8.2 VADER

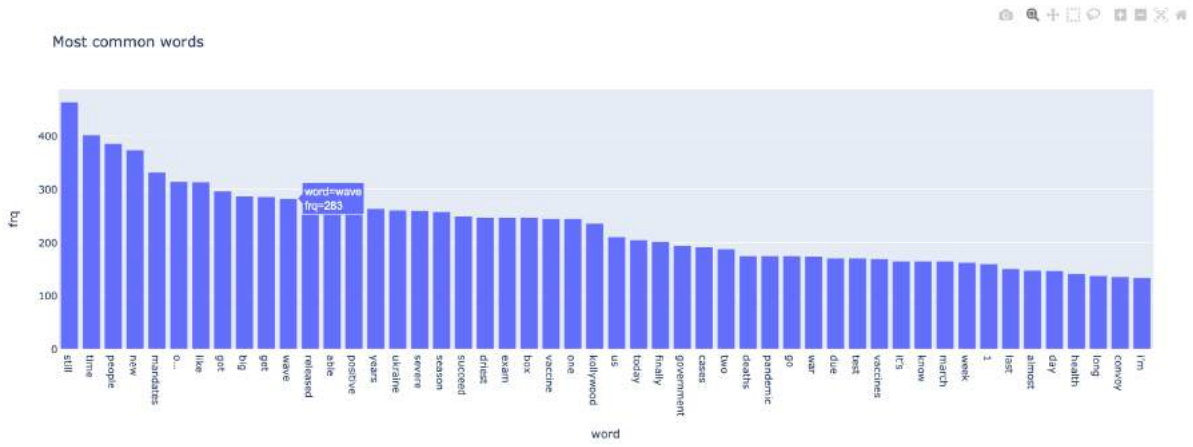
The first model we implement is VADER. VADER (Valence Aware Dictionary for Sentiment Reasoning) is an unsupervised model based on sentiment lexicons. A sentiment lexicon is a list of words which are labelled as positive or negative based on their semantic content.

There are different types of lexicons. Many do not take into account acronyms, emoticons or slang, seemingly important for sentiment analysis of social data. Some are also not able to measure the strength of the lexicons. VADER tries to mould the existing lexicons and supplements it with lexical features pertaining to emoticons, acronyms, slang etc. It uses 'wisdom of the crowd' to estimate the sentiment strength of the words. For example,

they include emoticons like ':)' which is a smiley stands for positive sentiment.VADER is smart enough to make sense that "did not like" as a negative sentiment.

We removed URLs, @user, 'rt' (retweet) and most commonly occuring words like covid, corona etc in this context. We have kept the hashtag information.

```
# Manipulation the dataframe, which columns to keep, colum renaming etc.
def prelim_processing(df):
  df=df[['col6', 'col7', 'col3']]
  df.columns= ['user_name', 'date', 'text']
  df.user_name=df.user_name.astype('category') #converts this column to categorical types
  df.user_name=df.user_name.cat.codes
  # df.set_index('user_name', inplace=True)
  df.date= pd.to_datetime(df.date).dt.date
  return df

#Pre-processing tweets
def fine_processing(df, more_words):
  stop_words=set(stopwords.words('english'))
  stop_words.update(more_words)

  remove_words = lambda x: ' '.join([word for word in x.split() if word not in stop_words])
  remove_url = lambda x: re.sub(r'https\S+', '',str(x)) # which starts with https and there is no whitespace in it
  remove_mentions = lambda x: re.sub(r'@\S+', '',str(x))
  to_lower = lambda x: x.lower()
  puncs = string.punctuation
  mod_puncs =puncs[0:2]+puncs[3:]
  remove_puncs = lambda x: x.translate(str.maketrans('','',mod_puncs))

  texts = df['text']
  mod_texts = texts.apply(remove_url)
  mod_texts = mod_texts.apply(remove_mentions)
  mod_texts = mod_texts.apply(to_lower)
  mod_texts = mod_texts.apply(remove_puncs)
  mod_texts =mod_texts.apply(remove_words)
  return mod_texts

df=prelim_processing(df)
more_words=['covid','#coronavirus','#coronavirusoutbreak','#coronavirusPandemic','#covid19','#covid_19','epitwitter','#ihavecorona'
mod_texts=fine_processing(df, more_words)
```



	index	label
0	positive	2558
1	negative	2309
2	neutral	2150

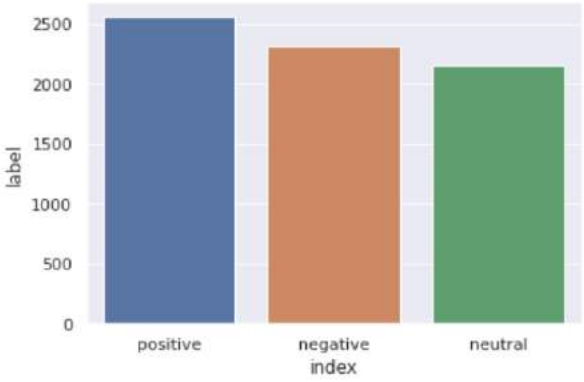


Figure 6: Polarity of the tweets

9 Topic Modelling: Latent Dirichlet Allocation

Topic modelling is a method for unsupervised classification of documents similar to clustering. It is one of the popular methods to identify topics in plethora of text. We have used topic modelling here to identify the trending topics among the COVID-19 tweets.

Assumptions:

- Each document is a "bag of words". Hence LDA does not take into account the order of the sentence or the grammar.
- Commonly occurring words like "am/is/are/but/the/of" are deleted since they do not carry enough information about the topics. Moreover, we can eliminate frequently occurring words which occur in 80-90 percent of the documents. For example, if documents are only concerned with films, all documents will contain words like "actor/celebrity/movie/film" which do not add any specific information.
- The number of topics is fixed apriori.

Algorithm:

- We need to know the words and their associated probability of belonging to a particular topic **t**.
- Go through the document and randomly assign each word in the document to one of the 'k' topics.
- For each document **d**, go through each word **w** and compute:
  - $p(\mathbf{t}|\mathbf{d})$= the proportion of words in document **d** that are assigned to topic **t**. captures how many words belong to topic **t** for a given document
  - $p(\mathbf{w}|\mathbf{t})$= It tries to capture how many docs are in topic **t** because of word **w**.
  - Then we have an updation rule to update the terms.

LDA represents each document as a mixture of topics. Moreover, a topic is mixture of words.

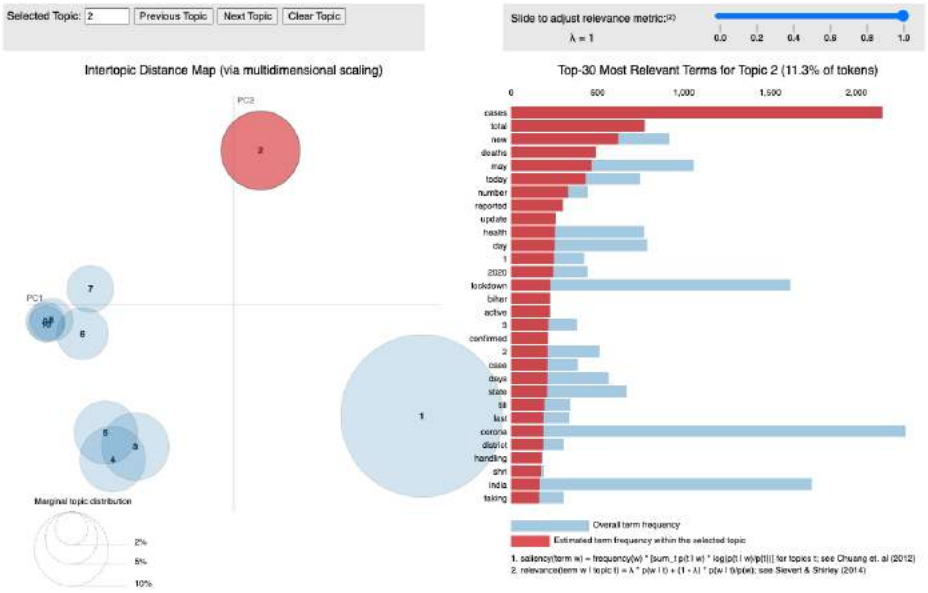


Figure 7: Top words for topic 1

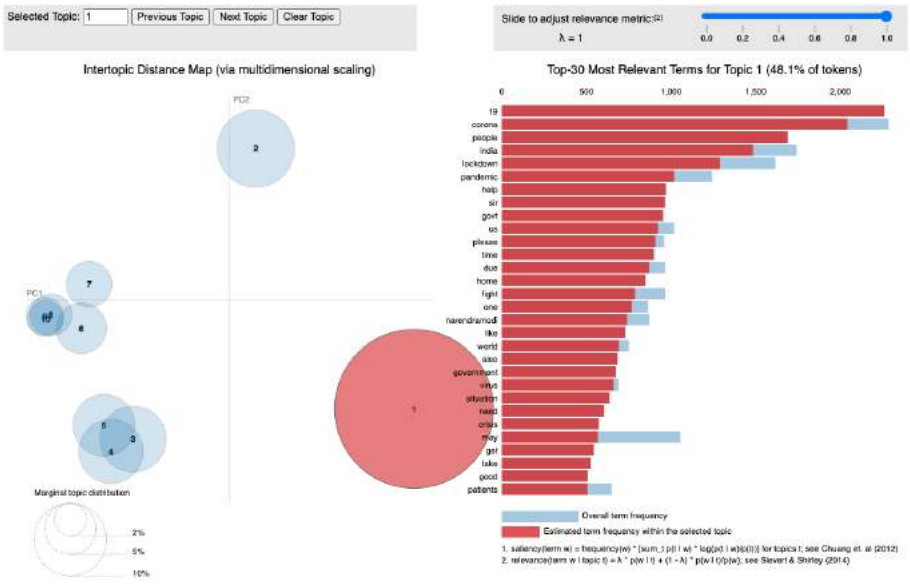


Figure 8: Top words for topic 2

10 Emotional Divergence vs Tweet sentiments

The term emotional divergence measures the diversity of the emotions expressed in a text. In this work, we have tried looking at how people interact with emotionally divergent covid-19 tweets.

Fig[9] and Fig[10] show how the no of retweets and no of likes by the user varies wrt the emotional divergence of the tweets during May 6th, 2020 to May 12th, 2020.

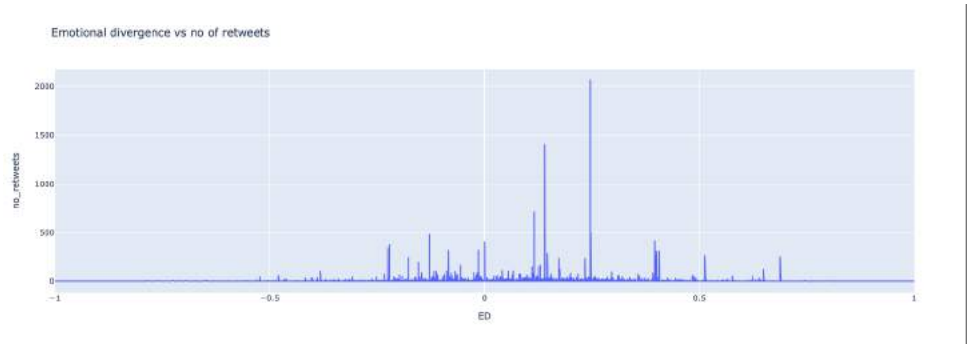


Figure 9: No of retweets vary with emotional divergence of tweets

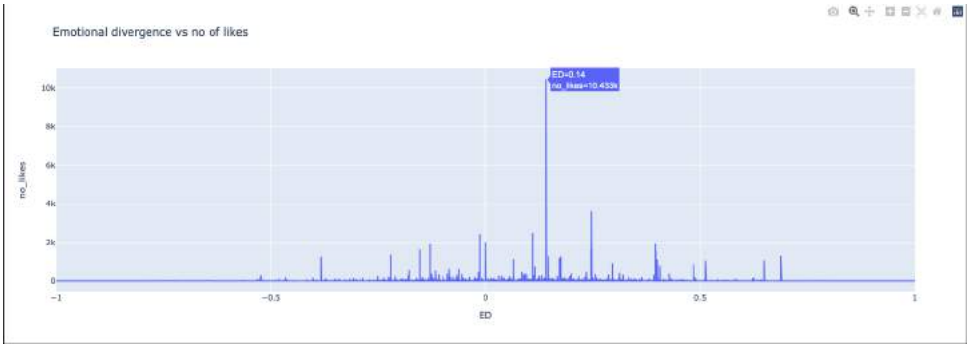


Figure 10: No of likes vary with emotional divergence of tweets

Fig[11] and Fig[12] show how the no of retweets and no of likes by the user varies wrt the emotional divergence of the tweets during September 14th, 2021 to September 20th, 2021.

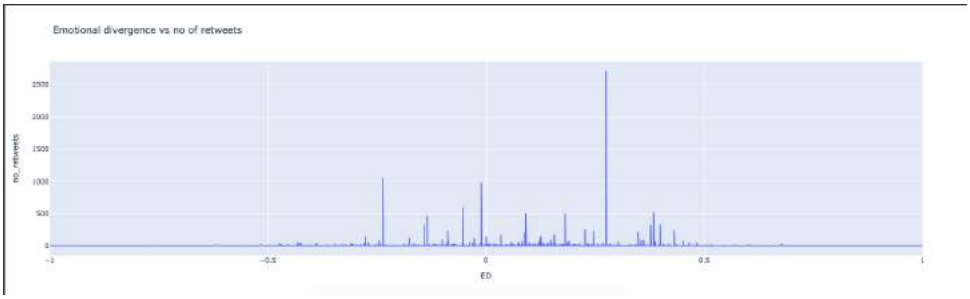


Figure 11: No of retweets vary with emotional divergence of tweets

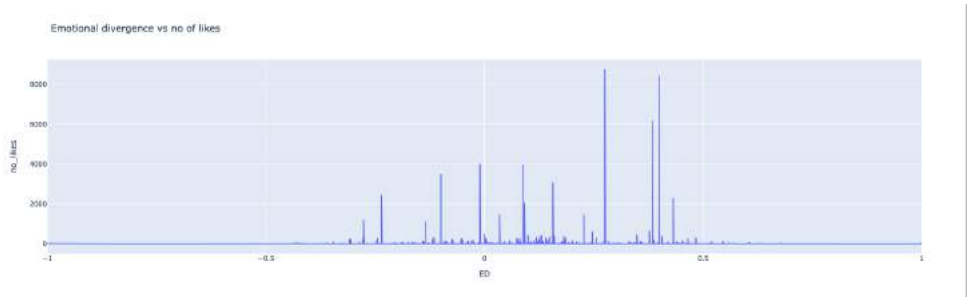


Figure 12: No of likes vary with emotional divergence of tweets

We adopt the definition of "emotional divergence" (ED) from [12], which is defined as "the (normalized) absolute difference between the positive and the negative sentiment score delivered by SentiStrength." Since we have implemented VADER to get the positive and negative scores, we use this to measure the emotional divergence of the tweets. The ED is calculated as $(p-n)/2$, where where p is a positive score and n is a negative score output by the VADER model. VADER gives a positive score ranging from 0 to 1 and negative score from -1 to 0.

11 Machine Learning models

The data should now be suitable to be simulated once it is cleaned and given a quick overview through the use of WordClouds. Text data must be processed into a numeric vocabulary which can be interpreted by machine learning techniques prior all of that can be utilized. Since algorithms can only process numerical data, it is crucial to perform this function. In principle, there exist various techniques for encoding textual information into numeric form, like CountVectorizer, TfIdf, among others. This bag of words concept is employed to construct the Count Vectorizer. It operates by calculating the frequency of instances every word occurs within every document (tweets). If term frequency-inverse document frequency (TfIdf) is employed, the quantitative results increase as the number of words in the collection increases, but the increase is neutralized by the occurrence of the same word in separate samples. In the specific instance of the phrase 'pandemic' emerging many times in a document and being present in approximately 80percent of total of the tweets, count vectorizer will designate a great premium to pandemic, but TfIdf will assign a negligible value to pandemic because it is a common word that appears in many documents and is therefore not a useful word to classify the tweets.

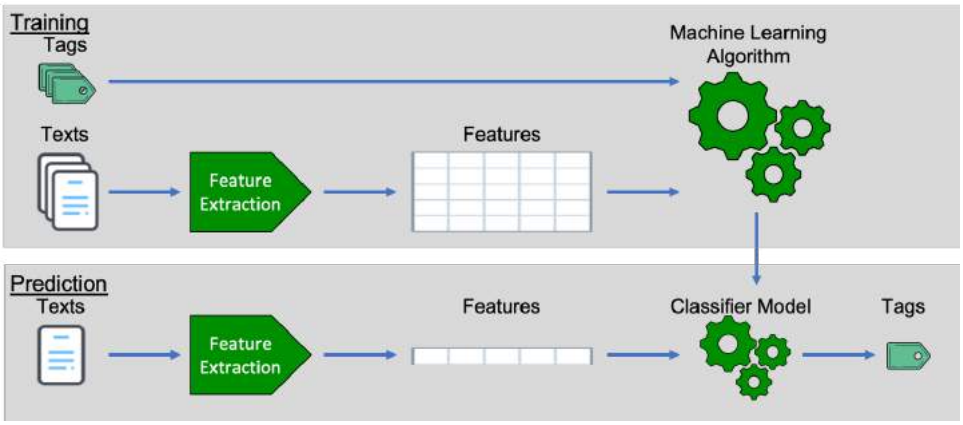


Figure 13: Machine Learning NLP Text Classification Process

11.1 Sentiment Analysis using Logistic Regression

We can encode a tweet or a series of words as a vector of dimension V, wherein V refers to the quantity of our dictionary. Thus the vector gets increasingly sparse as the value of

V rises. Consequently, we eventually wind up with a variety of additional characteristics and end up training a substantial percentage of V features. A longer time frame for training and a longer prediction period might happen as a result. As a conclusion, we will capture frequencies from each word and aggregate these into a frequency vocabulary. Fundamentally, our goal is to split the training data set into two groups: positive tweets and negative tweets. Count what number of times a keyword occurs in both positive and negative tweets and generate a scripting language vocabulary of the keywords' frequency.

```
In [18]: from sklearn.model_selection import train_test_split

         train,valid = train_test_split(data,test_size = 0.2,random_state=0,stratify = data.sentiment.values) #stratification
         print("train shape : ", train.shape)
         print("valid shape : ", valid.shape)

         from sklearn.feature_extraction.text import CountVectorizer
         from nltk.corpus import stopwords
         stop = list(stopwords.words('english'))
         vectorizer = CountVectorizer(decode_error = 'replace',stop_words = stop)

         X_train = vectorizer.fit_transform(train.text.values)
         X_valid = vectorizer.transform(valid.text.values)

         y_train = train.sentiment.values
         y_valid = valid.sentiment.values

         print("X_train.shape : ", X_train.shape)
         print("X_train.shape : ", X_valid.shape)
         print("y_train.shape : ", y_train.shape)
         print("y_valid.shape : ", y_valid.shape)

         train shape :  (2472, 3)
         valid shape :  (618, 3)
         X_train.shape :  (2472, 8551)
         X_train.shape :  (618, 8551)
```

Figure 14: Split of test and train data

```
Best: 0.668 using {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
```

Figure 15: Results for Logistic regression

Create a vector of bias units for each tweet, consisting of the total of all the positive frequencies (words from positive tweets) of all the words, as well as the sum of all the negative frequencies of all the words.

```
Training accuracy Score    :  0.9955501618122977
Validation accuracy Score :  0.6957928802588996
              precision    recall  f1-score   support

       anger       0.59      0.64      0.61       143
        fear       0.58      0.63      0.60       148
         joy       0.85      0.71      0.77       173
         sad       0.77      0.80      0.79       154

    accuracy                           0.70       618
   macro avg       0.70      0.69      0.69       618
weighted avg       0.71      0.70      0.70       618
```

Figure 16: Classification report for Logistic regression

11.2 Sentiment Analysis using Ridge regression

Ridge regression is a regression method rather than a classification method, as the name implies. We use a threshold to convert it to a classifier. In either scenario, we are merely training a linear model whose definition is defined by a higher dimensional space. It performs since this problem is essentially linearly separable - which is, all that is needed to split the groups is a simple hyperplane. The "ridge" value enables it to work in situations where the solution is not totally linearly separable or when the issue is rank deficient (in which case the optimisation would be degenerate). Ridge classifier is constructed by using ridge regression approach, which transforms its predicted value to -1 and 1.

High-dimensional text classification issues (with many characteristics) are more likely to be linearly separable than low-dimensional text classification problems (since any d+1 points in a d-dimensional space can be separated with a linear classifier without respect to their labels). As a result, linear classifiers, whether ridge regression or support vector machines with a linear kernel, are expected to perform well. To avoid over-fitting by separating patterns of each class by large margins, the ridge parameter or C for the SVM (as tdc mentions +1) are used in both situations to manage the complexity of the classifier and aid to avoid over-fitting (i.e. the decision surface passes down the middle of the gap between the two collections of points). The ridge/regularisation parameters, on the other hand, must be correctly set in order to get good performance (I use leave-one-out cross-validation as it is cheap).

Non-linear approaches, on the other hand, are very powerful, and it is difficult to avoid overfitting. This is why ridge regression works so effectively. Perhaps there is a non-linear classifier which trumps the best linear model in terms of generalisation efficiency, but determining such features with the specified threshold of input samples supplied is still too challenging. T he clearer the framework, less the effort we need calculating the coefficients, less the likelihood is to over-fit, therefore the superior the results.

```
# summarize results
print(f"Best: {grid_result.best_score_:.3f} using {grid_result.best_params_}")
```

```
Best: 0.646 using {'alpha': 1.0}
```

Figure 17: Results for Ridge regression

Further challenge is feature selection; ridge regression minimizes over-fitting by normalising input features to preserve them, and modelling is uncomplicated because you only have to determine the value of a single regression parameter in ridge regression. Trying to avoid overfitting by picking the best set of features makes model selection more difficult. Each component has a measure of autonomy, that makes it feasible to over-fit the feature selection criterion and end up with a set of features that are best for this sample of data, but not so good for generalising. Thus, omitting feature selection and instead relying on regularisation can frequently result in superior prediction performance.

11.3 Sentiment Analysis using KNN Classification

The classification method will operate in the following manner: for each tweet in the test dataset (d), we will calculate the Euclidean distance between d and each sample in the training dataset (D). Then, we'll choose the k samples that are closest to d, i.e. those with the shortest distances from d. We will take the class that is assigned to the majority of these k samples and assign it to the test instance.

1. Compute the distance between d and every sample in D.
2. Choose the k samples from D that are nearest to d.
3. Assign d the label y of the majority class.

```
# summarize results
print(f"Best: {grid_result.best_score_:.3f} using {grid_result.best_params_}")
```

```
Best: 0.469 using {'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
```

Figure 18: Results for KNN Classification

To evaluate our classifier's performance, we may now report on classification accuracy, macro-average (precision, recall, and F1), and confusion matrix for various values of k. To validate our conclusions drawn throughout the implementing steps listed above, we utilise Scikit-kNN learn's implementation to train and test the k closest neighbour classifier on the provided dataset and compare its performance measures to those obtained above.

Classification Report for k = 1 is:

	precision	recall	f1-score	support
negative	0.82	0.48	0.60	1834
neutral	0.30	0.68	0.42	615
positive	0.46	0.47	0.46	472
accuracy			0.52	2921
macro avg	0.53	0.54	0.50	2921
weighted avg	0.65	0.52	0.54	2921

Figure 19: Classification report for k=1

A cursory examination of Scikit-classification learn's reports reveals little distinction between the two classifiers. Indeed, the accuracies are more or less consistent within the same range, namely 49 to 52 percent. Although the results are not spectacular, we can do better when it comes to accurately predicting the correct labels. It is possible to increase our input features by utilising a dense vector representation, also known as Word2Vec, instead of employing a bag-of-words format.

11.4 Sentiment Analysis using Multinomial Naive Bayes classifiers

Multinomial Naive Bayes classifiers are frequently used as a starting point for sentiment analysis tasks. The Naive Bayes technique's fundamental principle is to determine the probabilities of classes assigned to texts by combining the probabilities of words and classes.

Given the dependent feature vector (x1,x2,x3...xn) and classes Ck, Bayes' theorem is stated mathematically as the following relationship:

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, \ldots, x_n)}$$

Thus, the relation can be simplified to

$$\hat{y} = \arg\max_k (\ln P(C_k) + \sum_{i=1}^{n} \ln P(x_i \mid C_k))$$

To avoid underflow, log probabilities can be used.

$$\hat{\theta}_{ki} = \frac{N_{ki} + \alpha}{N_k + \alpha n}$$

Thus, the final decision rule is defined as follows:

$$\hat{y} = \arg\max_k (\ln P(C_k) + \sum_{i=1}^{n} \ln \frac{N_{ki} + \alpha}{N_k + \alpha n})$$

A Pipeline class was introduced to make it easier to work with the vectorizer ⟶ transformer ⟶ classifier. Grid search was used to tune n-grams range, IDF usage, TF-IDF normalisation type, and Naive Bayes alpha. The selected hyper-parameters' performance was evaluated using a test set that was not used during the model training stage.

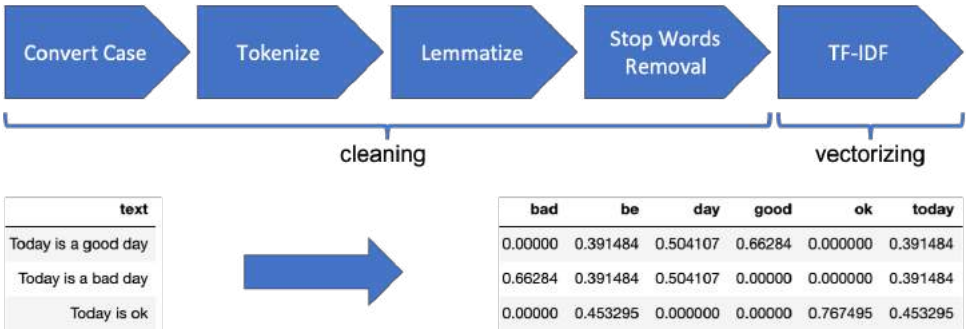


Figure 20: Naive Bayes Data Preprocessing Pipeline

A difficult issue to ask about the NB algorithm is: given that the conditional independence assumption in the NB algorithm is rarely valid in practise, how can the NB algorithm perform so well as a classifier?

In short, the answer resides in the distribution of dependencies rather than in dependency itself; somehow, the effect of dependencies cancels out as a result of dispersion.

We generate our model by fitting it to the MultinomialNB dataset. To determine the arguments that can be passed during fitting the model, it is recommended to consult the sklearn webpage for the module in question.

We quantify the model's quality. We analyse the predictions using the metrics module from the sklearn framework.

```
training accuracy Score    :  0.9526699029126213
Validation accuracy Score :  0.6779935275080906
              precision    recall  f1-score   support

       anger       0.54      0.67      0.60       124
        fear       0.66      0.57      0.61       183
         joy       0.71      0.82      0.76       126
         sad       0.81      0.69      0.74       185

    accuracy                           0.68       618
   macro avg       0.68      0.69      0.68       618
weighted avg       0.69      0.68      0.68       618
```

Figure 21: Classification report for MultinomialNB

We have observed that our model is more than 60 percent accurate. We may now tweak our model's parameters to improve its accuracy.

11.5 Sentiment Analysis using Stochastic Gradient Descent

Gradient descent is an iterative process that begins at a random point on a function and gradually descends its slope until it reaches the function's lowest point.

Stochastic Gradient Descent - Classifier (SGD-Classifier) may cause some users to believe SGD is a classifier. However, this is not the case! SGD Classifier is an SGD-optimized linear classifier (SVM, logistic regression, etc.). These are two entirely distinct notions. SGD is an optimization technique, whereas Logistic Regression or linear Support Vector Machine is a machine learning algorithm/model. Consider that a machine learning model defines a loss function, which the optimization approach attempts to decrease or maximise.

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

The term "learning rate" refers to a flexible parameter that has a significant effect on the algorithm's convergence. Increased learning rates cause the algorithm to take enormous leaps down the slope, and it may even jump across the minimum point, missing it entirely. As a result, it is always prudent to maintain a low learning rate, such as 0.01. Additionally, it can be demonstrated mathematically that the gradient descent method makes larger steps down the slope if the starting point is located high above and takes small steps closer to the goal to ensure that it does not miss it and is also fast enough.

Different choices for L entail different classifiers or regressors:

- Hinge (soft-margin): equivalent to Support Vector Classification. $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$.
- Perceptron: $L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$.
- Modified Huber: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))^2$ if $y_i f(x_i) > 1$, and $L(y_i, f(x_i)) = -4y_i f(x_i)$ otherwise.
- Log: equivalent to Logistic Regression. $L(y_i, f(x_i)) = \log(1 + \exp(-y_i f(x_i)))$.
- Least-Squares: Linear regression (Ridge or Lasso depending on R). $L(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$.
- Huber: less sensitive to outliers than least-squares. It is equivalent to least squares when $|y_i - f(x_i)| \leq \varepsilon$, and $L(y_i, f(x_i)) = \varepsilon|y_i - f(x_i)| - \frac{1}{2}\varepsilon^2$ otherwise.
- Epsilon-Insensitive: (soft-margin) equivalent to Support Vector Regression. $L(y_i, f(x_i)) = \max(0, |y_i - f(x_i)| - \varepsilon)$.

Gradient descent is a technique for minimising a cost function. Gradient descent is one of the most widely used optimization algorithms and is by far the most frequently used method for optimising neural networks. However, these types of methods can also be used to optimise linear classifiers such as Logistic Regression and Linear Support Vector Machines.

```
Training accuracy Score    :  0.9991909385113269
Validation accuracy Score :  0.6893203883495146
              precision    recall  f1-score   support

       anger       0.65      0.59      0.62       169
        fear       0.54      0.67      0.60       129
         joy       0.81      0.72      0.76       162
         sad       0.77      0.77      0.77       158

    accuracy                           0.69       618
   macro avg       0.69      0.69      0.69       618
weighted avg       0.70      0.69      0.69       618
```

Figure 22: Classification report for Stochastic Gradient Descent

Why do we utilise SGD classifiers when linear classifiers such as LogReg or SVM are already available?

Because the minimum of the Logistic Regression cost function cannot be calculated directly, we attempt to reduce it using Stochastic Gradient Descent, alternatively called Online Gradient Descent. Another reason to use SGD Classifier is that SVM or logistic regression cannot be used if the record cannot be kept in RAM. SGD Classifier, on the other hand, continues to function.

11.6 Sentiment Analysis using Random Forest Classifier

Following data preprocessing in the appropriate format, we will construct the classifier by using Random Forest Algorithm inside this section. Random Forest Classifier is characterized by a large multitude of decision trees operating as several machine classification methods to accomplish powerful forecasting results in comparison to using a solitary decision tree operating as an independent classifier, in which individual class prediction models' O/P help compensate votes as well as the amount of votes to outcome end up making the simulation O/P for Random Forest. The most advantageous aspect of Random Forest is that it can be utilised to tackle regression and classification issues.

This architecture contains a massive dataset D, which contains m rows and d columns. Because we are utilising Random Forest, this means that countless models of individual decision trees will be generated using the concept of Bagging, which requires the use of multiple machine learning algorithms.

Scikit-learn determines the importance of each node in a decision tree using Gini Importance, assuming only two child nodes (binary tree):

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

- ni sub(j)= the importance of node j
- w sub(j) = weighted number of samples reaching node j
- C sub(j)= the impurity value of node j
- left(j) = child node from left split on node j
- right(j) = child node from right split on node j

Row Sampling and Column Sampling will be used to include the new records into Decision Trees D1–D5. Though with Row and Column Sampling, certain rows and columns will be identical between (RS+FS) models. At the Random Forest level, the final measure of a feature's importance is its average value over all trees. The sum of the relevance values for each feature on each tree is calculated and divided by the total number of trees. Decision Tree D now provides predictions, and the Test Data is used in Decision Trees.

The importance for each feature on a decision tree is then calculated as:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

- fi sub(i)= the importance of feature i
- ni sub(j)= the importance of node j

At the Random Forest level, the final measure of a feature's importance is its average value over all trees. The sum of the importance values assigned to each feature on individual trees is calculated and divided by the total number of trees.

$$RFfi_i = \frac{\sum_{j \in all\ trees} normfi_{ij}}{T}$$

- RFfi sub(i) = the importance of feature i calculated from all trees in the Random Forest model
- normfi sub(ij) = the normalized feature importance for i in tree j
- T = total number of trees

In comparison to Low Bias, High Variance is defined as the tendency of the decision tree to produce greater error when new test data is applied. Now, the Random Forest functions well by determining that each Decision Tree will have a high variance due to the new test data, but when these Trees are joined and a majority vote is taken, the high variance for Decision Trees is changed to a low variance. If we have 1000 records in a dataset and change 200 of them, the result will likewise be low variance, resulting in high accuracy, using combined Decision Trees.

After fitting the model with the Random Forest Classifier and performing prediction tests, we will use the accuracy score and the f1 score.

```
Training accuracy Score    :  1.0
Validation accuracy Score :  0.6343042071197411
              precision    recall  f1-score   support

       anger       0.62      0.59      0.60       162
        fear       0.46      0.69      0.55       107
         joy       0.78      0.54      0.64       208
         sad       0.69      0.78      0.73       141

    accuracy                           0.63       618
   macro avg       0.64      0.65      0.63       618
weighted avg       0.66      0.63      0.64       618
```

Figure 23: Classification report for Random Forest Classifier

Furthermore, Random Forest decision trees exhibit two characteristics: low bias and great variance. Low Bias is a notion that claims that if a decision tree is completed correctly, the output will be more accurate and the training error will be minimal.

11.7 Sentiment Analysis using XGBoost

XGBoost is a high-performance implementation of gradient boosted decision trees. eXtreme Gradient Boosting is the abbreviation for XGBoost. XGBoost outperforms the majority of prediction models. This is why we're going to use it to categorise our tweets.

By optimising the algorithm, we can increase the score. This can be accomplished in part through hyperparameter adjustment.

```
[19:00:00] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the
objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to res
tore the old behavior.
Training accuracy Score    :  0.8972491909385113
Validation accuracy Score :  0.6715210355987055
              precision    recall  f1-score   support

       anger       0.61      0.60      0.61       156
        fear       0.52      0.67      0.59       123
         joy       0.79      0.66      0.71       174
         sad       0.78      0.75      0.77       165

    accuracy                           0.67       618
   macro avg       0.67      0.67      0.67       618
weighted avg       0.69      0.67      0.68       618
```

Figure 24: Classification report for XGB

XGBoost is a member of a family of boosting algorithms designed to transform weak learners into strong ones. A week learner is one who outperforms random guesses by a small margin. Boosting is a sequential procedure in which trees are generated sequentially using the knowledge from a previously grown tree. This process gradually gains knowledge from the data and attempts to improve prediction accuracy in succeeding iterations. XGboost is a high-performance implementation of gradient boosted decision trees. It is based on the gradient boosting framework. This machine learning algorithm includes the following features:

1. Parallel Processing: It is enabled for parallel processing, which means that when you run XG boost, it will automatically utilise all of your laptop's/cores. machine's
2. Regularization: Regularization is a technique used in linear and tree-based models to avoid overfitting.
3. Internally, XGBoost is intended to handle missing values. The missing data are addressed in such a way that the model captures any trend in the missing values.
4. It also enables user-defined goal functions in addition to regression, classification, and ranking tasks. An objective function is used to quantify the model's performance given a set of parameters. Additionally, it accommodates user-defined evaluation measures.
5. XGBoost enables us to save our data matrix and model for later use. If we have a huge data set, rather than redoing the computation, we can simply keep the model and reuse it in the future.
6. Unlike GBM, which terminates tree pruning when a negative loss occurs, XGBoost grows the tree to max depth and then prunes backward until the improvement in the loss function is less than a threshold.

12 Comparsion of results

Sentiment Analysis's purpose spans the positive to negative spectrum. As with other NLP attempts, it is typically treated as a classification problem, yet where precision is critical, it can be viewed as a regression problem. Traditionally, sentiment analysis was performed by using a huge labour force to read and manually review texts. This method is time consuming and prone to human mistake. To automate this procedure, businesses turn to modern analytical methods. Sentiment Analysis presents a problem in that individuals express and interpret sentiment polarity and intensity differently. Additionally, depending on the context, words and sentences might have many meanings. While some of these concerns can be overcome, as with any analytical activity, there is almost always a trade-off between speed and performance. We examine five broad approaches, each with its own set of advantages and disadvantages.

	Model	Test accuracy
1	Logistic Regression	0.695793
4	Stochastic Gradient Decent	0.689320
3	Naive Bayes	0.677994
5	XGBoost	0.671521
2	Random Forest	0.634304

Figure 25: Comparative report

13 The difficulties of sentiment analysis

Sentiment Analysis is a difficult task. The following are some of the difficulties[13] encountered in Twitter Sentiment Analysis.

1. Recognize subjective textual elements: Subjective textual elements contain sentimental value. In certain cases, the same word might be considered subjective, while in others, it can be considered objective. This complicates the task of identifying subjective passages of text. For instance: 1. Mr.language Dennis's was really crude. 2. Crude oil is extracted from seabeds. In the first example, the term "crude" is used subjectively, however in the second example, it is fully objective.
2. Domain dependence[24]: In various domains, the same sentence or phrase may have a distinct meaning. For instance, the term "unpredictable" is considered positive

in the context of films, dramas, and so on, but has a negative connotation when used to a vehicle's steering.

3. Detection of Sarcasm: Sarcastic statements express a negative judgement about a target in an unusual way by utilising positive words. For instance, "Nice perfume." You are required to shower in it." Although the sentence contains solely good words, it conveys a negative emotion.
4. Contradictory expressions: There are some sentences in which only a portion of the text defines the document's overall polarity. For instance, "This film has the potential to be fantastic." It sounds like an excellent plot, with well-known stars and a good supporting cast. In this situation, a straightforward bag of words technique will label it as good sentiment, yet the underlying sentiment is negative.
5. Explicit Negation of Sentiment: In addition to the simple no, not, never, etc., sentiment can be negated in a variety of ways. Such negations are notoriously difficult to discern. "It eschews any suspense and predictability associated with Hollywood films." Suspense and predictability have a negative connotation in this context, and the use of "avoids" cancels their respective connotations.
6. Order dependence: Sentiment Analysis/Opinion Mining requires an understanding of discourse structure. For instance, A is superior to B expresses the polar opposite of B is superior to A.
7. Entity Recognition: It is necessary to extract text regarding a certain thing and then analyse sentiment toward that entity. "I despise Microsoft, but I enjoy Linux," for instance. While a simple bag-of-words analysis would classify it as neutral, it has a distinct sentiment for both of the entities mentioned in the sentence.
8. Developing a classifier to distinguish subjective from objective tweets. Current research focuses mostly on appropriately identifying positive and negative data. There is a need to examine tweets with and without sentiment in greater detail.
9. Comparative analysis. The bag of words model does not perform well in comparisons. For instance, "IITs are better than the majority of private colleges," the tweet would be judged good for both IITs and private colleges using the bag of words approach, as it omits the reference to "better."
10. Analyzing Facebook messages for emotion. There has been less effort on sentiment analysis using Facebook data, owing to a variety of restrictions imposed by the Facebook graph API and security standards governing data access.
11. Globalization: While the majority of current research focuses on English material, Twitter has a diverse user base from all over the world.

14 Applications of sentiment analysis

There are numerous uses for Sentiment Analysis. The following are examples of cross-domain applications:

1. Apps that rely on reviews from other websites. Almost everything can now be found with a quick search on the Internet. For example, you can leave feedback on products you've used, express your thoughts on current events, or express your thoughts on the quality of a service you've received. As a result, a system for

extracting feelings regarding a certain product or service is required. Automated feedback or ratings for a certain product, item, etc. will help us. For both the users and the sellers, this would be a good option As a sub-component of technology, applications

2. Recommendation systems can benefit from sentiment prediction algorithms as well. Items with a high volume of negative comments or low ratings will not be recommended by the recommender system. Abuse of language and other bad aspects of online communication are all too common. Just by identifying a negative attitude and taking action against it, they can be identified.
3. Organizational Applications for Data Mining People today are more likely to check out internet reviews of things before making a purchase. For many firms, the success or failure of their product is decided by the opinions of their customers on the internet. As a result, businesses benefit greatly from Sentiment Analysis. Online reviews are also used by businesses to improve their products, reputation, and customer satisfaction by extracting sentiment from the online reviews.
4. Sentiment Analysis has also been used in sociology and other sectors, such as medicine and sports, to identify trends in human emotions, particularly on social media.
5. Smart Home Use Cases In the future, smart houses are expected to be a common feature in homes. People will be able to operate any element of their home with a tablet in the future, when all homes are networked. The Internet of Things (IoT) has recently been the subject of a lot of research (IoT). The Internet of Things (IoT) will incorporate Sentiment Analysis as well. For example, the home's ambiance could be changed to provide a relaxing and serene atmosphere based on the user's present attitude or emotion. Trend forecasting can make use of Sentiment Analysis as well. Tracking public opinion can provide valuable information about sales trends and consumer satisfaction.

References

[1] Diaz, M., Johnson, I., Lazar, A., Piper, A. M., and Gergle, D. *Addressing Age-Related Bias in Sentiment Analysis.* Association for Computing Machinery, New York, NY, USA, 2018, p. 1–14.

[2] Ghasiya, P., and Okamura, K. Investigating covid-19 news across four nations: A topic modeling and sentiment analysis approach. *IEEE Access 9* (2021), 36645–36656.

[3] Goularas, D., and Kamis, S. Evaluation of deep learning techniques in sentiment analysis from twitter data. In *2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML)* (2019), pp. 12–17.

[4] Kaur, H., and Kumar, R. Sentiment analysis from social media in crisis situations. *International Conference on Computing, Communication & Automation* (2015), 251–256.

[5] Khan, M. T., Durrani, M. Y., Ali, A., Inayat, I., Khalid, S., and Khan, K. H. Sentiment analysis and the complex natural language. *Complex Adaptive Systems Modeling 4* (2016), 1–19.

[6] Matalon, Y., Magdaci, O., Almozlino, A., and Yamin, D. Using sentiment analysis to predict opinion inversion in tweets of political communication. *Scientific Reports 11* (03 2021).

[7] Mittal, S., Goel, A., and Jain, R. Sentiment analysis of e-commerce and social networking sites. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), 2300–2305.

[8] Mohamed Ridhwan, K., and Hargreaves, C. A. Leveraging twitter data to understand public sentiment for the covid-19 outbreak in singapore. *International Journal of Information Management Data Insights 1*, 2 (2021), 100021.

[9] Neppalli, V. K., Caragea, C., Squicciarini, A., Tapia, A., and Stehle, S. Sentiment analysis during hurricane sandy in emergency response. *International Journal of Disaster Risk Reduction 21* (2017), 213–222.

[10] Ren, Y., Wang, R., and Ji, D. A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences 369* (2016), 188–198.

[11] Ruz, G. A., Henríquez, P. A., and Mascareño, A. Sentiment analysis of twitter data during critical events through bayesian networks classifiers. *Future Generation Computer Systems 106* (2020), 92–104.

[12] Saragih, M., and Girsang, G. Sentiment analysis of customer engagement on social media in transport online. pp. 24–29.

[13] Saragih, M. H., and Girsang, A. S. Sentiment analysis of customer engagement on social media in transport online. In *2017 International Conference on Sustainable Information Engineering and Technology (SIET)* (2017), pp. 24–29.

[14] Symeonidis, S., Effrosynidis, D., and Arampatzis, A. A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. *Expert Systems with Applications 110* (2018), 298–310.