Industrial Project Report

Submitted in partial fulfillment of the degree of

B-tech in Electrical Engineering and electronics and communication
By

Pratham Bikram Sahi[11901622003]
Arnab Sarkar[11900322018]
sanjay singha[11900322013]
Khuhsi kumari sah[11901622005]

Students of first-year

SILIGURI INSTITUTE OF TECHNOLOGY

THIS IS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

AFFILIATED TO

Maulana Abul Kalam Azad University of Technology

Under the supervision of :- Mr. Ripam Kundu

Sikharthy Infotech Pvt. Ltd.

PROJECT ON :- UBER DATA ANALYSIS WITH MACHINE LEARNING

By

Arnab Sarkar[11900322018]
Khuhsi Kumari sah[11901622005]
Sanjay Singha[11900322013]
Pratham Bikram Sahi[11901622003]

UNDER THE GUIDANCE OF

Mr. Ripam Kundu

Project Guide

Sikharthy Infotech Pvt. Ltd.

THIS IS SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

B.Tech

IN

Electrical Engineering and Electronics and Communication Engineering

SILIGURI INSTITUTE OF TECHNOLOGY

AFFILIATED TO

Maulana Abul Kalam Azad University of Technology

Department of Electrical Engineering and Electronics and Communication

I hereby forward the documentation prepared under my supervision by **Ripam Kundu Sir** entitled **Siliguri Institute Of Technology** to be accepted as fulfillment of the requirement for the Degree of Bachelor of Technology in Electrical Engineering, **Siliguri Institute Of Technology** affiliated to **Maulana Abul Kalam Azad University of Technology** (**MAKAUT**).

Mr.Ripam Kundu **(Software Developer)** **Project Guide** **Sikharthy Infotech Pvt. Ltd.** **Shilpi Ghosal** **(Director)** **Sikharthy Infotech Pvt. Ltd.**	**HOD** **Department Of Electrical Engineering and Electronics and communication , SIT**

TPO
Siliguri Institute of Technology

Certificate of Approval

The foregoing project is hereby approved as a creditable study for the B.Tech in Electrical Engineering and electronics and communication engineering presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorsed or approved any statement made, opinion expressed or conclusion therein but approve this project only for the purpose for which it is submitted.

Final Examination for
Evaluation of the Project --

Signatures of Examiners

ABSTRACT

The paper explains the working of Sentiment Analysis, Sentiment analysis is a methodology for analyzing a piece of text to discover the sentiment hidden within it. It accomplishes this by combining machine learning and natural language processing (NLP). Sentiment analysis allows you to examine the feelings expressed in a piece of text[1].

Here are some resources that can help you get started with sentiment analysis in Python:

- [Sentiment Analysis using Python](https://techvidvan.com/tutorials/python-sentiment-analysis/)

- [Twitter Sentiment Analysis using Python](https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/)

- [A Complete Sentiment Analysis Project Using Python's Scikit-Learn](https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2)

- [Sentiment Analysis Using Machine Learning Techniques on Python](https://ieeexplore.ieee.org/abstract/document/8663224/)

- [Getting Started with Sentiment Analysis using Python - Hugging Face](https://huggingface.co/blog/sentiment-analysis-python)

- [Sentiment Analysis: First Steps With Python's NLTK Library](https://realpython.com/python-nltk-sentiment-analysis/)

I hope this helps!

Source: Conversation with Bing, 10/5/2023(1) Sentiment Analysis using Python [with source code] - TechVidvan. https://techvidvan.com/tutorials/python-sentiment-analysis/ Accessed 10/5/2023.

(2) Twitter Sentiment Analysis using Python - GeeksforGeeks. https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/ Accessed 10/5/2023.

(3) A Complete Sentiment Analysis Project Using Python's Scikit-Learn. https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-

using-pythons-scikit-learn-b9ccbb0405c2"9 HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2"ccbb HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2"0405 HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2"c HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2" HYPERLINK "https://towardsdatascience.com/a-complete-sentiment-analysis-project-using-pythons-scikit-learn-b9ccbb0405c2"2 Accessed 10/5/2023.

(4) Sentiment Analysis Using Machine Learning Techniques on Python | IEEE .... https://ieeexplore.ieee.org/abstract/document/ HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/" HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/" HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/"8663224

HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/" HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/" HYPERLINK "https://ieeexplore.ieee.org/abstract/document/8663224/"/ Accessed 10/5/2023.

(5) Getting Started with Sentiment Analysis using Python - Hugging Face. https://huggingface.co/blog/sentiment-analysis-python Accessed HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023"10 HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023"/ HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023"5 HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023"/ HYPERLINK

"https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023" HYPERLINK "https://huggingface.co/blog/sentiment-analysis-python%20Accessed%2010/5/2023"2023.

(6) Sentiment Analysis: First Steps With Python's NLTK Library. https://realpython.com/python-nltk-sentiment-analysis/ Accessed 10/5/2023.

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the assistance and participation of a large number of individuals in this attempt. Our project report has been structured under the valued suggestion, support, and guidance of **Mr. Ripam Kundu**. Under his guidance, we have accomplished the challenging task in a very short time.

Finally, we express our sincere thankfulness to our family members for inspiring me all throughout and always encouraging us.

Group Mamber Signature

INTRODUCTION

We will use Python and its different libraries to complete the sentiment analysis model**.**

Getting Started with Sentiment Analysis using Python

Sentiment analysis is the automated process of tagging data according to their sentiment, such as positive, negative and neutral. Sentiment analysis allows companies to analyze data at scale, detect insights and automate processes.

In the past, sentiment analysis used to be limited to researchers, machine learning engineers or data scientists with experience in natural language processing. However, the AI community has built awesome tools to democratize access to machine learning in recent years. Nowadays, you can use sentiment analysis with a few lines of code and no machine learning experience at all! 🤯

1. What is Sentiment Analysis?

Sentiment analysis is a natural language processing technique that identifies the polarity of a given text. There are different flavors of sentiment analysis, but one of the most widely used techniques labels data into positive, negative and neutral. For example, let's take a look at these tweets mentioning @VerizonSupport:

- "dear @verizonsupport your service is straight 💩 in dallas.. been with y′all over a decade and this is all time low for y′all. i′m talking no internet at all." → Would be tagged as "Negative".

- "@verizonsupport ive sent you a dm" → would be tagged as "Neutral".

- "thanks to michelle et al at @verizonsupport who helped push my no-show-phone problem along. order canceled successfully and ordered this for pickup today at the apple store in the mall." → would be tagged as "Positive".

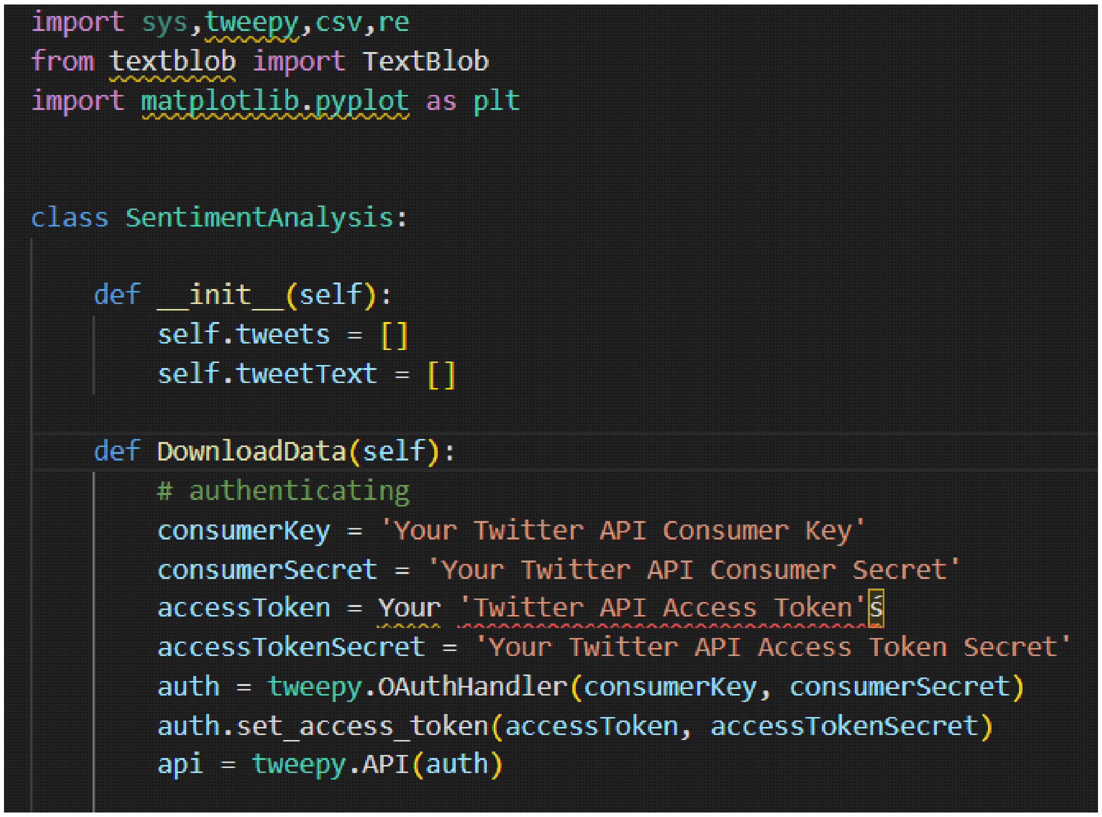
Sentiment analysis allows processing data at scale and in real-time. For example, do you want to analyze thousands of tweets, product reviews or support tickets? Instead of sorting through this data manually, you can use sentiment analysis to automatically understand how people are talking about a specific topic, get insights for data-driven decisions and automate business processes.

Sentiment analysis is used in a wide variety of applications, for example:

- Analyze social media mentions to understand how people are talking about your brand vs your competitors.
- Analyze feedback from surveys and product reviews to quickly get insights into what your customers like and dislike about your product.
- Analyze incoming support tickets in real-time to detect angry customers and act accordingly to prevent churn.

Before analysis, you need to install textblob and tweepy libraries using !pip install command on your Jupyter Notebook.

```
import sys,tweepy,csv,re
from textblob import TextBlob
import matplotlib.pyplot as plt
```



```
import sys,tweepy,csv,re
from textblob import TextBlob
import matplotlib.pyplot as plt


class SentimentAnalysis:

    def __init__(self):
        self.tweets = []
        self.tweetText = []

    def DownloadData(self):
        # authenticating
        consumerKey = 'Your Twitter API Consumer Key'
        consumerSecret = 'Your Twitter API Consumer Secret'
        accessToken = Your 'Twitter API Access Token's
        accessTokenSecret = 'Your Twitter API Access Token Secret'
        auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
        auth.set_access_token(accessToken, accessTokenSecret)
        api = tweepy.API(auth)
```

Tweepy supports both OAuth 1a (application-user) and OAuth 2 (application-only) authentication. Authentication is handled by the tweepy.AuthHandler class.

OAuth 2 is a method of authentication where an application makes API requests without the user context. Use this method if you just need read-only access to public information.

You first register our client application and acquire a consumer key and secret. Then you create an AppAuthHandler instance, passing in our consumer key and secret.

Before the authentication, you need to have Twitter Developer Account. If you don't have, you can apply by using this link. Getting Twitter developer account usually takes a day or two, or sometimes more, for your application to be reviewed by Twitter.

Step 2: Authentication for Twitter API

```
# Authentication
consumerKey = “Type your consumer key here”
consumerSecret = “Type your consumer secret here”
accessToken = “Type your accedd token here”
accessTokenSecret = “Type your access token secret here”
auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth)
```

After your authentication, you need to use tweepy to get text and use Textblob to calculate positive, negative, neutral, polarity and compound parameters from the text.

Step 3: Getting Tweets With Keyword or Hashtag

```
#Sentiment Analysis
def percentage(part,whole):
 return 100 * float(part)/float(whole)
keyword = input(“Please enter keyword or hashtag to search: ”)
noOfTweet = int(input (“Please enter how many tweets to analyze: ”))
tweets = tweepy.Cursor(api.search, 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[ˎ neg ˎ ]
 neu = score[ˎ neu ˎ ]
 pos = score[ˎ pos ˎ ]
 comp = score[ˎ compound ˎ ]
 polarity += analysis.sentiment.polarity

 if neg > pos:
 negative_list.append(tweet.text)
 negative += 1
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, ˎ .1fˎ )
negative = format(negative, ˎ .1fˎ )
neutral = format(neutral, ˎ .1fˎ )
```

The scenario in this post like that, the user should type keyword or hashtag (lockdown2 london) and type how many tweets (2500) that want to get and analyse.

The number of tweets parameter is important because of the limit.

After getting 2500 tweets about lŏckdown2 london ̗, let ̦s have a look number of tweets that which sentiments

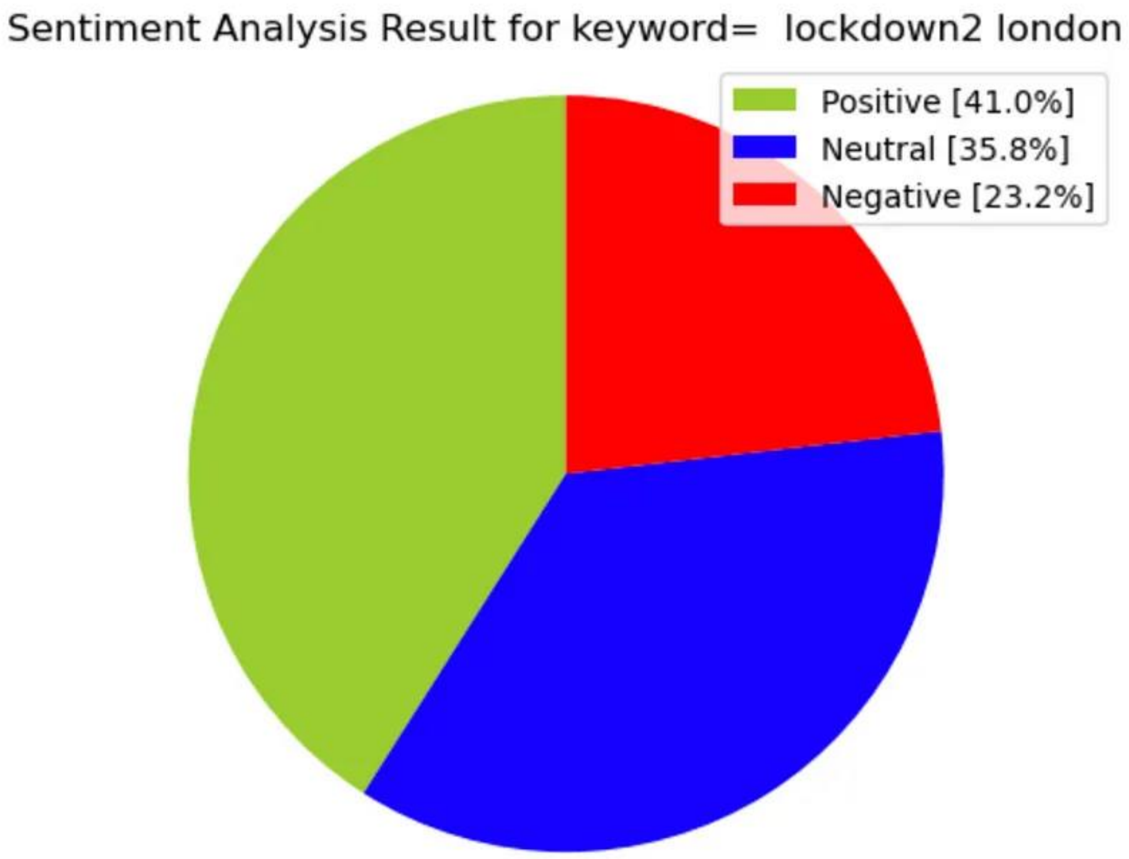
```
#Number of Tweets (Total, Positive, Negative, Neutral)
tweet_list = pd.DataFrame(tweet_list)
neutral_list = pd.DataFrame(neutral_list)
negative_list = pd.DataFrame(negative_list)
positive_list = pd.DataFrame(positive_list)
```

```
print("total number: ",len(tweet_list))
print("positive number: ",len(positive_list))
print("negative number: ", len(negative_list))
print("neutral number: ",len(neutral_list))
```

You could get 2500 tweets and;

1025 (41.0%) of tweets include positive sentiment
580 (23.2%) of tweets include negative sentiment
895 (35.8%) of tweets include neutral sentiment

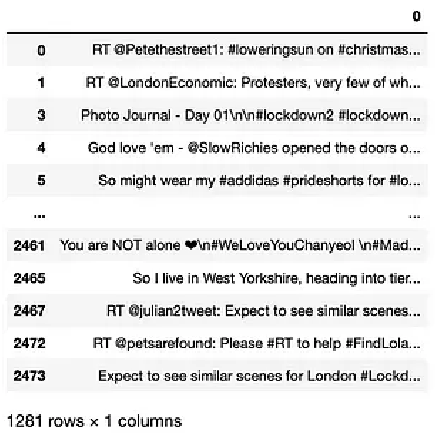
```
#Creating PieCart
labels = ['Positive ['+str(positive)+'%]' , 'Neutral
['+str(neutral)+'%]', 'Negative ['+str(negative)+'%]']
sizes = [positive, neutral, negative]
colors = ['yellowgreen', 'blue', 'red']
patches, texts = plt.pie(sizes,colors=colors, startangle=90)
plt.style.use('default')
plt.legend(labels)
plt.title("Sentiment Analysis Result for keyword= "+keyword+"")
plt.axis('equal')
plt.show()
```



Step 4: Cleaning Tweets to Analyse Sentiment

When you have a look tweet list you can see some duplicated tweets, so you need to drop duplicates records using drop_duplicates function.

```
tweet_list.drop_duplicates(inplace = True)
```

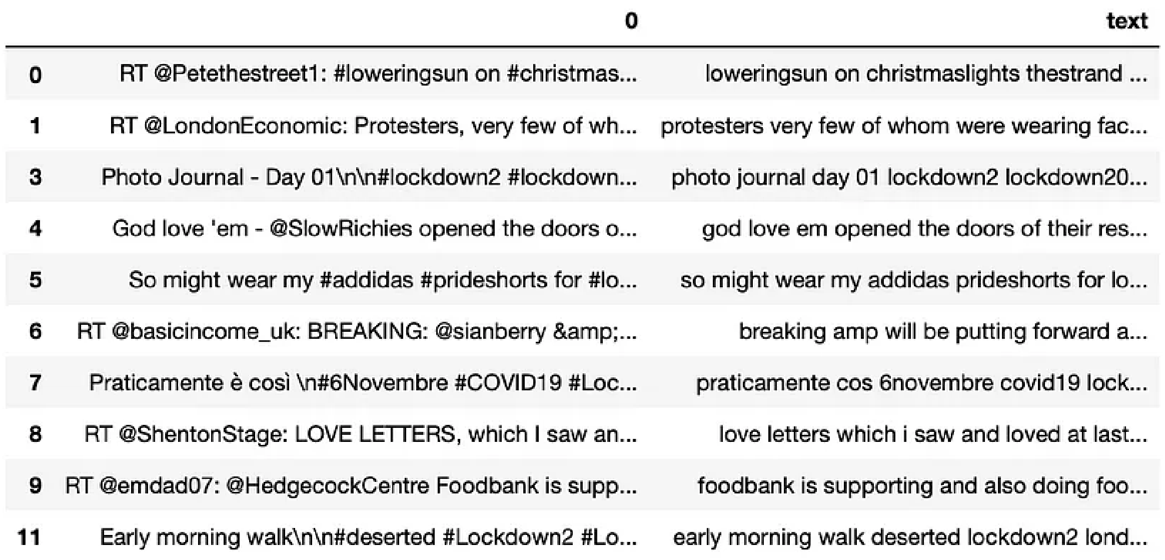


```
                                                    0
   0       RT @Petethestreet1: #loweringsun on #christmas...
   1      RT @LondonEconomic: Protesters, very few of wh...
   3     Photo Journal - Day 01\n\n#lockdown2 #lockdown...
   4       God love 'em - @SlowRichies opened the doors o...
   5        So might wear my #addidas #prideshorts for #lo...
 ...                                                  ...
2461 You are NOT alone ❤\n#WeLoveYouChanyeol \n#Mad...
2465             So I live in West Yorkshire, heading into tier...
2467         RT @julian2tweet: Expect to see similar scenes...
2472        RT @petsarefound: Please #RT to help #FindLola...
2473        Expect to see similar scenes for London #Lockd...
1281 rows × 1 columns
```

Our new data frame has 1281 unique tweets.

Firstly, I create new data frame (tw_list) and a new feature(text), then clean text by using lambda function and clean RT, link, punctuation characters and finally convert to lowercase.

```
#Cleaning Text (RT, Punctuation etc)
#Creating new dataframe and new features
tw_list = pd.DataFrame(tweet_list)
tw_list[ “text ”] = tw_list[0]
#Removing RT, Punctuation etc
remove_rt = lambda x: re.sub( “RT @\w+: “, ” “,x)
rt = lambda x: re.sub( “(@[A-Za-z0–9]+)|([⁰-9A-Za-z
\t])|(\w+:\/\/\S+)”,” “,x)
tw_list[“text”] = tw_list.text.map(remove_rt).map(rt)
tw_list[“text”] = tw_list.text.str.lower()
tw_list.head(10)
```

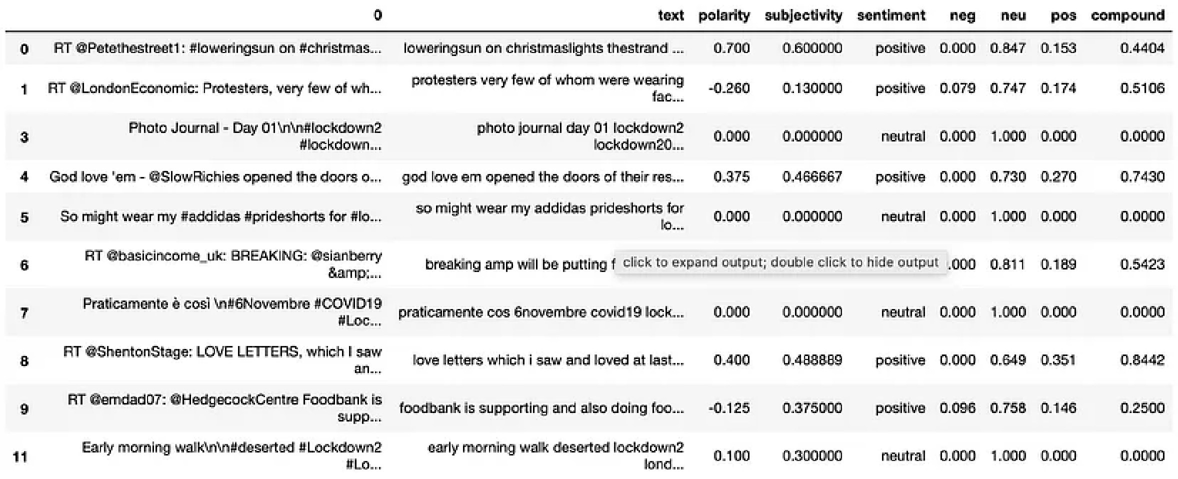


	0	text
0	RT @Petethestreet1: #loweringsun on #christmas...	loweringsun on christmaslights thestrand ...
1	RT @LondonEconomic: Protesters, very few of wh...	protesters very few of whom were wearing fac...
3	Photo Journal - Day 01\n\n#lockdown2 #lockdown...	photo journal day 01 lockdown2 lockdown20...
4	God love 'em - @SlowRichies opened the doors o...	god love em opened the doors of their res...
5	So might wear my #addidas #prideshorts for #lo...	so might wear my addidas prideshorts for lo...
6	RT @basicincome_uk: BREAKING: @sianberry &amp;...	breaking amp will be putting forward a...
7	Praticamente è così \n#6Novembre #COVID19 #Loc...	praticamente cos 6novembre covid19 lock...
8	RT @ShentonStage: LOVE LETTERS, which I saw an...	love letters which i saw and loved at last...
9	RT @emdad07: @HedgecockCentre Foodbank is supp...	foodbank is supporting and also doing foo...
11	Early morning walk\n\n#deserted #Lockdown2 #Lo...	early morning walk deserted lockdown2 lond...

Step 5: Sentiment Analyse

Now, I can use cleaned text to calculate polarity, subjectivity, sentiment, negative, positive, neutral and compound parameters again. For all calculated parameters, I create new features to my data frame

```
#Calculating Negative, Positive, Neutral and Compound values
tw_list[['polarity', 'subjectivity']] = tw_list['text'].apply(lambda Text:
pd.Series(TextBlob(Text).sentiment))
for index, row in tw_list['text'].iteritems():
 score = SentimentIntensityAnalyzer().polarity_scores(row)
 neg = score['neg']
 neu = score['neu']
 pos = score['pos']
 comp = score['compound']
 if neg > pos:
 tw_list.loc[index, 'sentiment'] = "negative"
 elif pos > neg:
 tw_list.loc[index, ‘sentiment’] = “positive”
 else:
 tw_list.loc[index, ‘sentiment’] = “neutral”
 tw_list.loc[index, ‘neg’] = neg
 tw_list.loc[index, ‘neu’] = neu
 tw_list.loc[index, ‘pos’] = pos
 tw_list.loc[index, ‘compound’] = comp
tw_list.head(10)
```



	0	text	polarity	subjectivity	sentiment	neg	neu	pos	compound
0	RT @Petethestreet1: #loweringsun on #christmas...	loweringsun on christmaslights thestrand ...	0.700	0.600000	positive	0.000	0.847	0.153	0.4404
1	RT @LondonEconomic: Protesters, very few of wh...	protesters very few of whom were wearing fac...	-0.260	0.130000	positive	0.079	0.747	0.174	0.5106
3	Photo Journal - Day 01\n\n#lockdown2 #lockdown...	photo journal day 01 lockdown2 lockdown20...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
4	God love 'em - @SlowRichies opened the doors o...	god love em opened the doors of their res...	0.375	0.466667	positive	0.000	0.730	0.270	0.7430
5	So might wear my #addidas #prideshorts for #lo...	so might wear my addidas prideshorts for lo...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
6	RT @basicincome_uk: BREAKING: @sianberry &amp;...	breaking amp will be putting f click to expand output; double click to hide output				.000	0.811	0.189	0.5423
7	Praticamente è così \n#6Novembre #COVID19 #Loc...	praticamente cos 6novembre covid19 lock...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000
8	RT @ShentonStage: LOVE LETTERS, which I saw an...	love letters which i saw and loved at last...	0.400	0.488889	positive	0.000	0.649	0.351	0.8442
9	RT @emdad07: @HedgecockCentre Foodbank is supp...	foodbank is supporting and also doing foo...	-0.125	0.375000	positive	0.096	0.758	0.146	0.2500
11	Early morning walk\n\n#deserted #Lockdown2 #Lo...	early morning walk deserted lockdown2 lond...	0.100	0.300000	neutral	0.000	1.000	0.000	0.0000

You can split your data frame into 3 groups based on sentiment. For this one, create 3 new data frame (tw_list_negative, tw_list_positive, tw_list_neutral) and import from original tw_list data frame

```
#Creating new data frames for all sentiments (positive, negative and neutral)
tw_list_negative = tw_list[tw_list[“sentiment”]==”negative”]
tw_list_positive = tw_list[tw_list[“sentiment”]==”positive”]
```

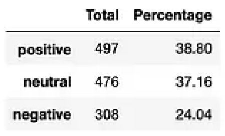
```
tw_list_neutral = tw_list[tw_list["sentiment"]=="neutral"]
```

Let's count values for sentiment features and see total — percentage.

#Function for count_values_in single columns

```
def count_values_in_column(data,feature):
 total=data.loc[:,feature].value_counts(dropna=False)

percentage=round(data.loc[:,feature].value_counts(dropna=False,norm
alize=True)*100,2)
 return pd.concat([total,percentage],axis=1,keys=['Total','Percentage'])
#Count_values for sentiment
count_values_in_column(tw_list,"sentiment")
```



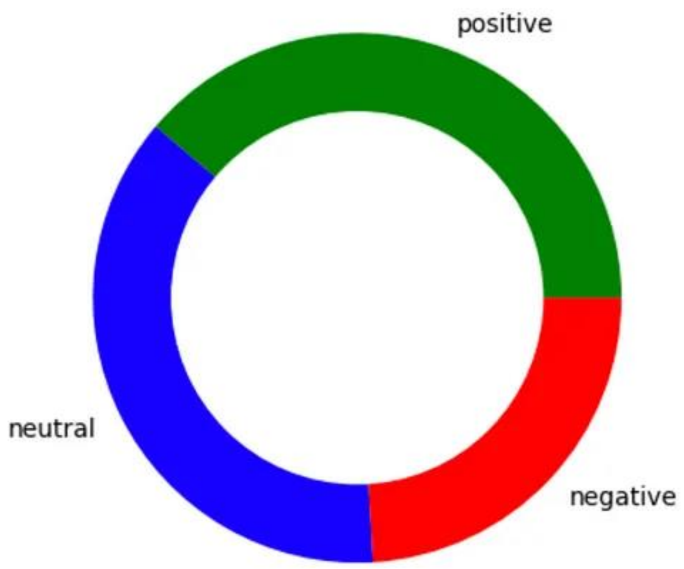
	Total	Percentage
positive	497	38.80
neutral	476	37.16
negative	308	24.04

You can create a chart by using number of sentiment tweets.

```
# create data for Pie Chart
pichart = count_values_in_column(tw_list,"sentiment")
names= pc.index
size=pc["Percentage"]

# Create a circle for the center of the plot
my_circle=plt.Circle( (0,0), 0.7, color='white')
plt.pie(size, labels=names, colors=['green','blue','red'])
```

```
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.show()
```

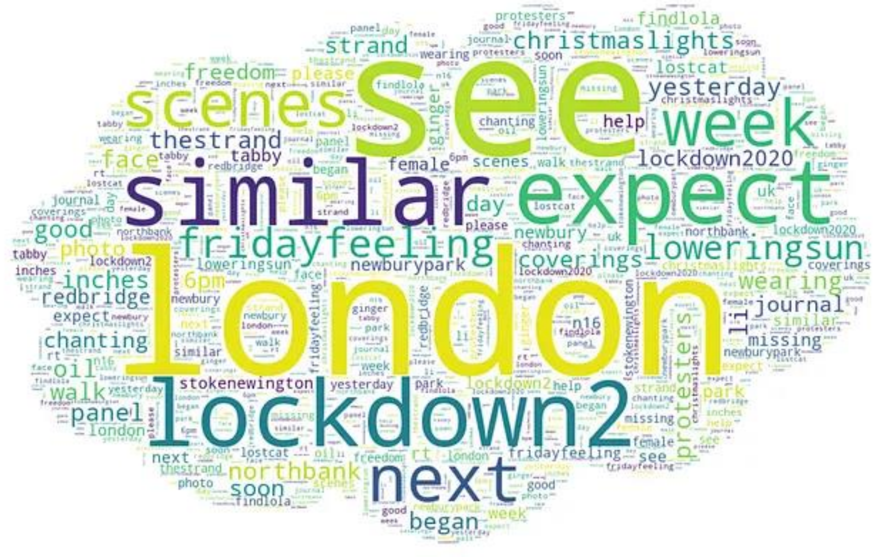


Now you can prepare to create worcloud using 1281 tweets, So you can realize that which words most used in these tweets. To create a worcloud, firstly let’s define a function below, so you can use wordcloud again for all tweets, positive tweets, negative tweets etc.

```
#Function to Create Wordcloud
def create_wordcloud(text):
 mask = np.array(Image.open(“cloud.png”))
 stopwords = set(STOPWORDS)
 wc = WordCloud(background_color=”white”,
 mask = mask,
 max_words=3000,
 stopwords=stopwords,
 repeat=True)
 wc.generate(str(text))
 wc.to_file(“wc.png”)
 print(“Word Cloud Saved Successfully”)
 path=”wc.png”
 display(Image.open(path))
```

After defining the function, you can have a look wordcloud for all tweets

```
#Creating wordcloud for all tweets
create_wordcloud(tw_list[“text”].values)
```



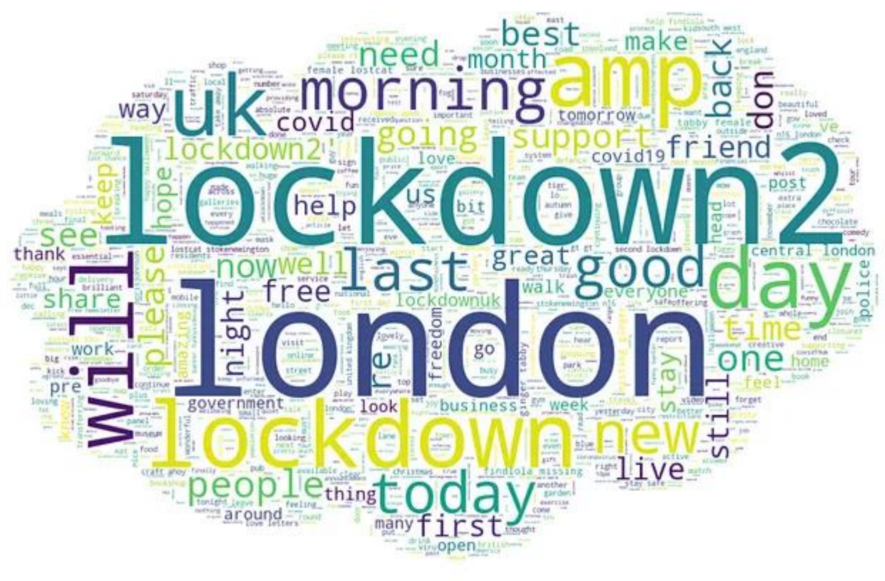
Word Cloud for tweets that have positive sentiments;

#Creating wordcloud for positive sentiment
create_wordcloud(tw_list_positive["text"].values)

Image by the author
Word Cloud for tweets that have negative sentiments;

```
#Creating wordcloud for negative sentiment
create_wordcloud(tw_list_negative["text"].values)
```

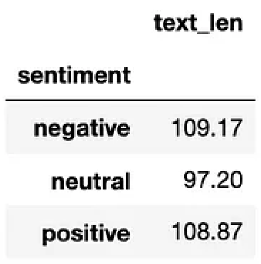
Let's calculate



the tweet length and word count. So you can see the density of words and characters used in tweets based on different sentiment.

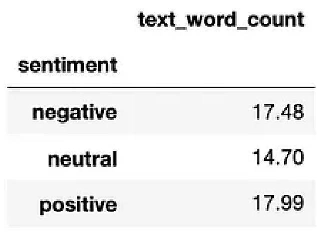
```
#Calculating tweet's lenght and word count
tw_list['text_len'] = tw_list['text'].astype(str).apply(len)
tw_list['text_word_count'] = tw_list['text'].apply(lambda x:
len(str(x).split()))
```

```
round(pd.DataFrame(tw_list.groupby("sentiment").text_len.mean()),2)
```



	text_len
sentiment	
negative	109.17
neutral	97.20
positive	108.87

```
round(pd.DataFrame(tw_list.groupby(“sentiment”).text_word_count.me
an()),2)
```



	text_word_count
sentiment	
negative	17.48
neutral	14.70
positive	17.99

Applying count vectorizer provides the capability to preprocess your text data prior to generating the vector representation making it a highly flexible feature representation module for text. After count vectorizer, it is possible to analyze the words with two or three or whatever you want.

Applying stemmer is also provides the root of words. So you can eliminate words that come from the same root, such as ;

connect
connection
connected
connections
connects

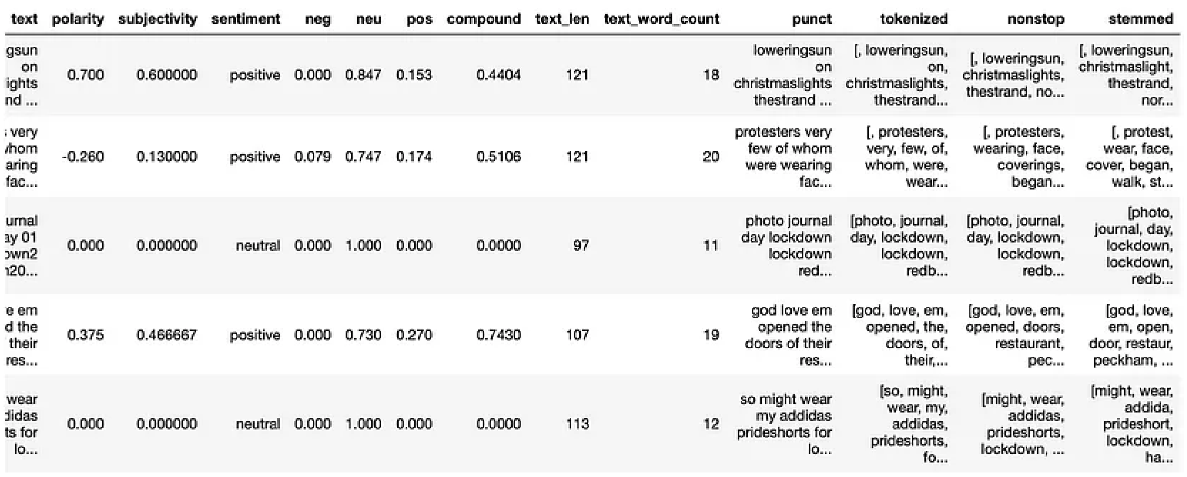
comes from “connect”. If you apply the stemmer function, you can consider these all words as same

```
#Removing Punctuation
def remove_punct(text):
 text = “”.join([char for char in text if char not in string.punctuation])
 text = re.sub(‘[0–9]+’, ‘’, text)
 return text
tw_list[‘punct’] = tw_list[‘text’].apply(lambda x: remove_punct(x))
#Appliyng tokenization
def tokenization(text):
    text = re.split('\W+', text)
    return text
tw_list['tokenized'] = tw_list['punct'].apply(lambda x:
tokenization(x.lower()))
#Removing stopwords
stopword = nltk.corpus.stopwords.words('english')
def remove_stopwords(text):
    text = [word for word in text if word not in stopword]
    return text

tw_list['nonstop'] = tw_list['tokenized'].apply(lambda x:
remove_stopwords(x))
#Appliyng Stemmer
ps = nltk.PorterStemmer()
def stemming(text):
    text = [ps.stem(word) for word in text]
    return text
tw_list['stemmed'] = tw_list['nonstop'].apply(lambda x: stemming(x))
#Cleaning Text
def clean_text(text):
    text_lc = "".join([word.lower() for word in text if word not in
string.punctuation]) # remove puntuation
    text_rc = re.sub('[0-9]+', '', text_lc)
    tokens = re.split('\W+', text_rc)    # tokenization
    text = [ps.stem(word) for word in tokens if word not in stopword]  #
remove stopwords and stemming
    return text
tw_list.head()
```

After applying countverctorizer, two results show us all 1281 tweets have 2966 unique words.

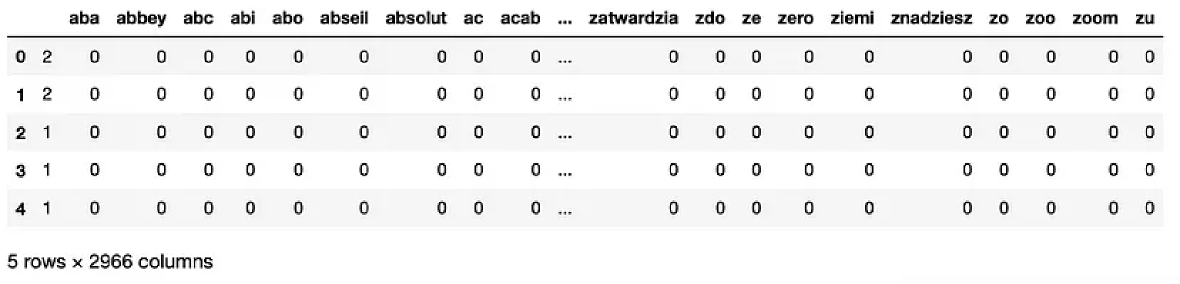
If you have a look at our data frame, you can see new features such as punct, tokenized, nonstop, stemmed.



text	polarity	subjectivity	sentiment	neg	neu	pos	compound	text_len	text_word_count	punct	tokenized	nonstop	stemmed
gsun on ights nd ...	0.700	0.600000	positive	0.000	0.847	0.153	0.4404	121	18	loweringsun on christmaslights thestrand ...	[, loweringsun, on, christmaslights, thestrand...	[, loweringsun, christmaslights, thestrand, no...	[, loweringsun, christmaslight, thestrand, nor...
very hom aring fac...	-0.260	0.130000	positive	0.079	0.747	0.174	0.5106	121	20	protesters very few of whom were wearing fac...	[, protesters, very, few, of, whom, were, wear...	[, protesters, wearing, face, coverings, began...	[, protest, wear, face, cover, began, walk, st...
urnal ay 01 own2 120...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000	97	11	photo journal day lockdown lockdown red...	[photo, journal, day, lockdown, lockdown, redb...	[photo, journal, day, lockdown, lockdown, redb...	[photo, journal, day, lockdown, lockdown, redb...
e em d the their res...	0.375	0.466667	positive	0.000	0.730	0.270	0.7430	107	19	god love em opened the doors of their res...	[god, love, em, opened, the, doors, of, their,...	[god, love, em, opened, doors, restaurant, pec...	[god, love, em, open, door, restaur, peckham, ...
wear didas ts for lo...	0.000	0.000000	neutral	0.000	1.000	0.000	0.0000	113	12	so might wear my addidas prideshorts for lo...	[so, might, wear, my, addidas, prideshorts, fo...	[might, wear, addidas, prideshorts, lockdown, ...	[might, wear, addida, prideshort, lockdown, ha...

Now, you can apply coun vectorizer the see all 2966 unique words as a new features.

```
#Appliyng Countvectorizer
countVectorizer = CountVectorizer(analyzer=clean_text)
countVector = countVectorizer.fit_transform(tw_list['text'])
print('{} Number of reviews has {} words'.format(countVector.shape[0],
countVector.shape[1]))
#print(countVectorizer.get_feature_names())
1281 Number of reviews has 2966 words
count_vect_df = pd.DataFrame(countVector.toarray(),
columns=countVectorizer.get_feature_names())
count_vect_df.head()
```

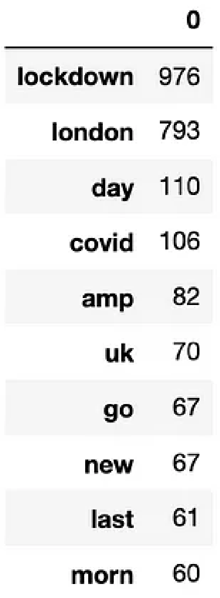


		aba	abbey	abc	abi	abo	abseil	absolut	ac	acab	...	zatwardzia	zdo	ze	zero	ziemi	znadziesz	zo	zoo	zoom	zu
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 2966 columns

You can sort values as a descending to see most used words

```
# Most Used Words
count = pd.DataFrame(count_vect_df.sum())
countdf = count.sort_values(0,ascending=False).head(20)
countdf[1:11]
```

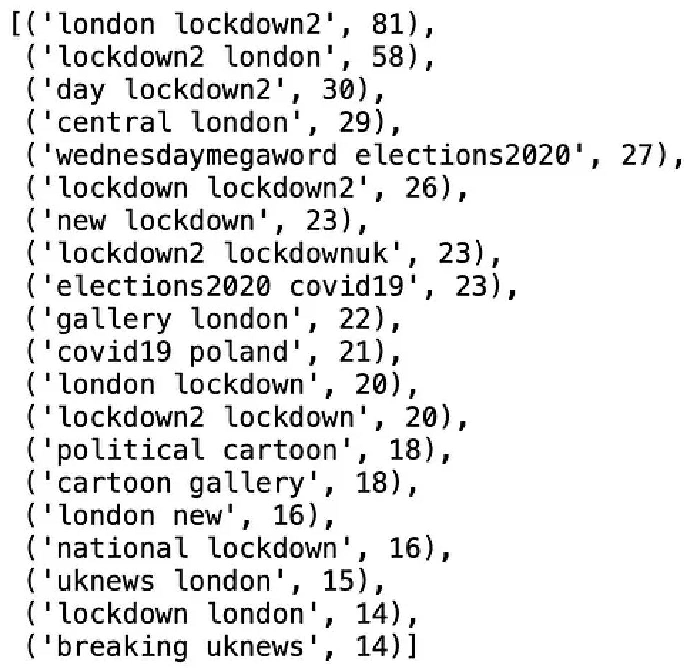


	0
lockdown	976
london	793
day	110
covid	106
amp	82
uk	70
go	67
new	67
last	61
morn	60

Building n gram model helps us to predict most probably word that might follow this sequence. Firstly let's create a function then built n2_bigram, n3_trigram etc.

```
#Function to ngram
def get_top_n_gram(corpus,ngram_range,n=None):
 vec = CountVectorizer(ngram_range=ngram_range,stop_words =
‘english’).fit(corpus)
 bag_of_words = vec.transform(corpus)
 sum_words = bag_of_words.sum(axis=0)
 words_freq = [(word, sum_words[0, idx]) for word, idx in
vec.vocabulary_.items()]
 words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
 return words_freq[:n]
#n2_bigram
n2_bigrams = get_top_n_gram(tw_list[‘text’],(2,2),20)
n2_bigrams
```

```
#n3_trigram
n3_trigrams = get_top_n_gram(tw_list['text'],(3,3),20)
n3_trigrams
```



```
[('london lockdown2', 81),
 ('lockdown2 london', 58),
 ('day lockdown2', 30),
 ('central london', 29),
 ('wednesdaymegaword elections2020', 27),
 ('lockdown lockdown2', 26),
 ('new lockdown', 23),
 ('lockdown2 lockdownuk', 23),
 ('elections2020 covid19', 23),
 ('gallery london', 22),
 ('covid19 poland', 21),
 ('london lockdown', 20),
 ('lockdown2 lockdown', 20),
 ('political cartoon', 18),
 ('cartoon gallery', 18),
 ('london new', 16),
 ('national lockdown', 16),
 ('uknews london', 15),
 ('lockdown london', 14),
 ('breaking uknews', 14)]
```

Finally, you can analyze sentiment using tweets and you can realize which words most used and which words used together.

Conclusion

It was a wonderful and learning experience for us while working on this project. This project took us through the various phases of project development and gave us real insight of data sentiment analysis model. The joy of work and thrill involved while tackling the information about coding.

Thank you