Twitter Mining, Trend Analysis & Named Entity Recognition using Python

Name: Vedant Mane

Class: MSc Computer Science – Part II

Seat Number: 190118

Subject: Social Network Analysis – Innovative Practical

TASK:

Write a program for mining Twitter to identify tweets for a specific period and identify trends and named entities.

Software:

1. Python (3.8.3)
2. Jupyter Notebook
3. Twitter Developer API

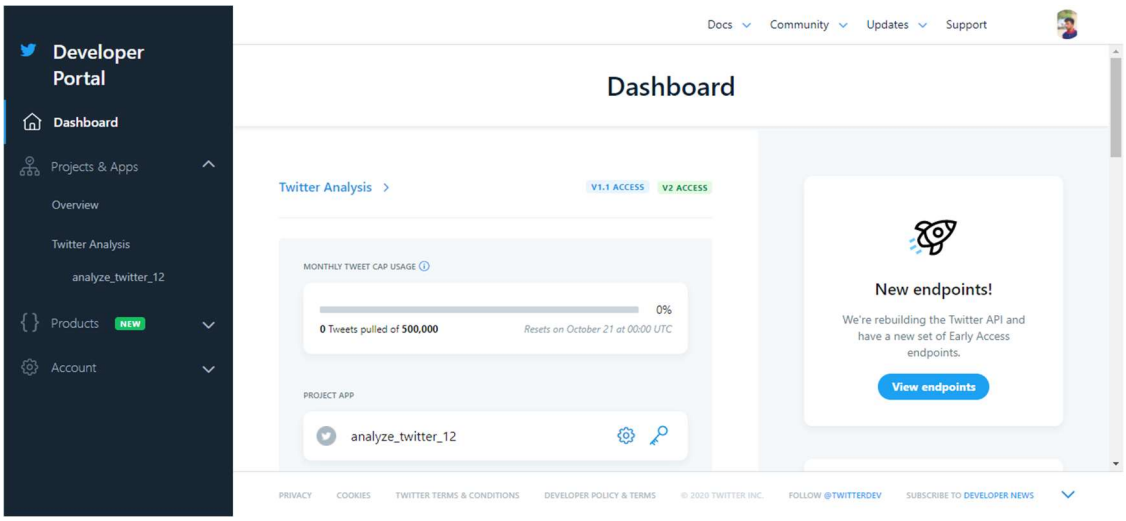
Libraries:

1. NumPy
2. Pandas
3. Matplotlib
4. Seaborn
5. Wordcloud
6. Pillow
7. Spacy
8. TextBlob

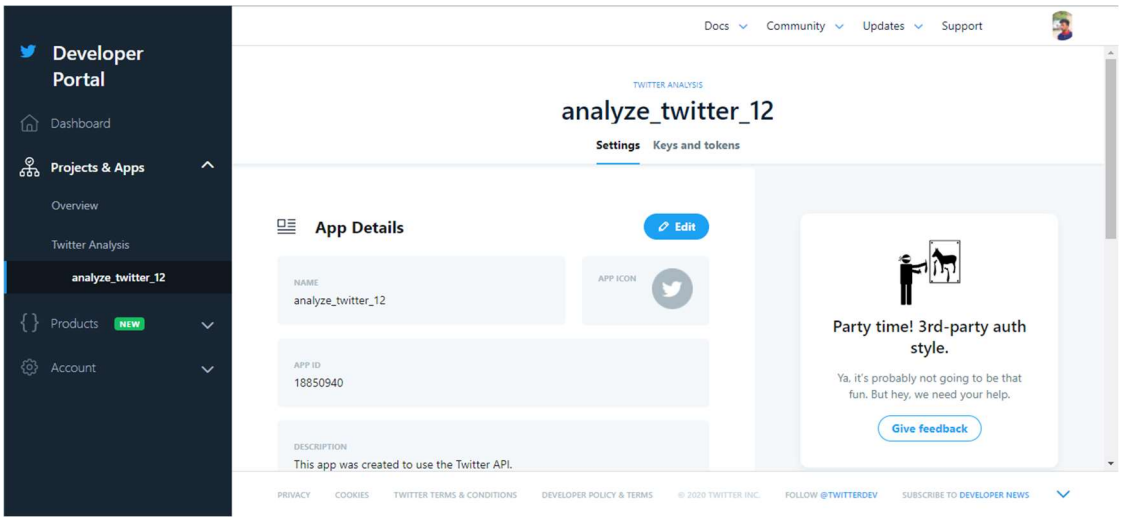
Twitter API Steps:

Step 1: Create a Twitter Account if you don't have one. (Link: https://twitter.com/)

Step 2: Sign up for a Developer Account. (Link: https://developer.twitter.com/en)



Step 3: Create an app in the Twitter Developer Console.



Step 4: Save the CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET keys in a secure file and do not share them with anyone.

Code:

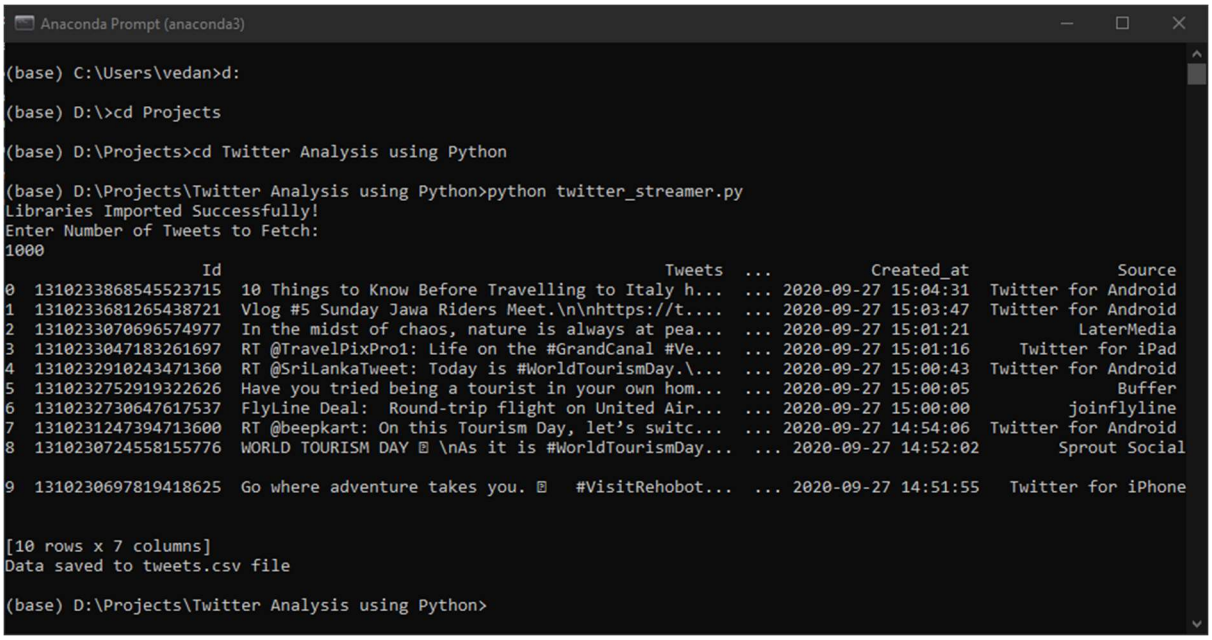
Filename: twitter_streamer.py

Source Code:

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

import numpy as np
import pandas as pd

import twitter_credentials

print("Libraries Imported Successfully!")

# Hashtags = ['Data Science', 'Artificial Intelligence', 'Machine Learning', 'Deep
Learning']

## Twitter AUTHENTICATION
class TwitterAuthenticator():

   def authenticate_twitter_app(self):
      auth = OAuthHandler(twitter_credentials.CONSUMER_KEY,
twitter_credentials.CONSUMER_SECRET)
      auth.set_access_token(twitter_credentials.ACCESS_TOKEN,
twitter_credentials.ACCESS_TOKEN_SECRET)
      return auth

## Twitter CLIENT
class TwitterClient():
   def __init__(self, twitter_user = None):
      self.auth = TwitterAuthenticator().authenticate_twitter_app()
      self.twitter_client = API(self.auth)
      self.twitter_user = twitter_user

   def get_twitter_client_api(self):
      return self.twitter_client

   def getTweets(self, count, hashtags):
      tweets = []
      last_id = -1
      if len(tweets) < count:
         while len(tweets) <= count:
```

```
            count_tweets = count - len(tweets)
            try:
                new_tweets = api.search(q = hashtags, lang = 'en', count = count_tweets,
                                        max_id = str(last_id - 1))
                if not new_tweets:
                    break
                tweets.extend(new_tweets)
                last_id = new_tweets[-1].id
            except tweepy.TweepError as e:
                break
        return tweets

class DataFrameGenerator():
    """
    """
    def tweets_to_data_frame(self, tweets):
        df = pd.DataFrame(data = [[tweet.id, tweet.text, len(tweet.text),
tweet.favorite_count, tweet.retweet_count, tweet.created_at, tweet.source]
                                  for tweet in tweets],
                          columns = ['Id', 'Tweets', 'Length', 'Likes', 'Retweets', 'Created_at',
'Source'])
        return df


if __name__ == "__main__":

    hashtags = ['Travel', 'Wanderlust']
    count = int(input("Enter Number of Tweets to Fetch:\n"))

    twitter_client = TwitterClient()
    api = twitter_client.get_twitter_client_api()
    tweets = twitter_client.getTweets(count, hashtags)
    #print(dir(tweets[0]))

    frame_generator = DataFrameGenerator()
    df = frame_generator.tweets_to_data_frame(tweets)

    print(df.head(10))
    df.to_csv("tweets.csv", index = False, header = True)
    print("Data saved to tweets.csv file")
```

Output:


Anaconda Prompt (anaconda3)
(base) C:\Users\vedan>d:
(base) D:\>cd Projects
(base) D:\Projects>cd Twitter Analysis using Python
(base) D:\Projects\Twitter Analysis using Python>python twitter_streamer.py
Libraries Imported Successfully!
Enter Number of Tweets to Fetch:
1000
Id Tweets ... Created_at Source
0 1310233868545523715 10 Things to Know Before Travelling to Italy h... ... 2020-09-27 15:04:31 Twitter for Android
1 1310233681265438721 Vlog #5 Sunday Jawa Riders Meet.\n\nhttps://t.... ... 2020-09-27 15:03:47 Twitter for Android
2 1310233070696574977 In the midst of chaos, nature is always at pea... ... 2020-09-27 15:01:21 LaterMedia
3 1310233047183261697 RT @TravelPixPro1: Life on the #GrandCanal #Ve... ... 2020-09-27 15:01:16 Twitter for iPad
4 1310232910243471360 RT @SriLankaTweet: Today is #WorldTourismDay.\... ... 2020-09-27 15:00:43 Twitter for Android
5 1310232752919322626 Have you tried being a tourist in your own hom... ... 2020-09-27 15:00:05 Buffer
6 1310232730647617537 FlyLine Deal: Round-trip flight on United Air... ... 2020-09-27 15:00:00 joinflyline
7 1310231247394713600 RT @beepkart: On this Tourism Day, let’s switc... ... 2020-09-27 14:54:06 Twitter for Android
8 1310230724558155776 WORLD TOURISM DAY \nAs it is #WorldTourismDay... ... 2020-09-27 14:52:02 Sprout Social
9 1310230697819418625 Go where adventure takes you. #VisitRehobot... ... 2020-09-27 14:51:55 Twitter for iPhone
[10 rows x 7 columns]
Data saved to tweets.csv file
(base) D:\Projects\Twitter Analysis using Python>

Filename: twitter_analysis.ipynb

Importing Libraries:

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
from textblob import TextBlob
import re
%matplotlib inline
plt.style.use("ggplot")
from IPython.display import Markdown, display
def printmd(string):
  display(Markdown(string))
print("Libraries imported successfully")
```

Output:

Loading DataFrame:

Code:

```
df = pd.read_csv("tweets.csv")
df['Created_at'] = pd.to_datetime(df['Created_at'], format = "%Y-%m-%d %H:%M:%S")
df['Date'] = df['Created_at'].dt.date
df.head(10)
```

Output:

Load DataFrame

```
In [2]: df = pd.read_csv("tweets.csv")
        df['Created_at'] = pd.to_datetime(df['Created_at'], format = "%Y-%m-%d %H:%M:%S")
        df['Date'] = df['Created_at'].dt.date
        df.head(10)
```

Out[2]:

	Id	Tweets	Length	Likes	Retweets	Created_at	Source	Date
0	1309913978076884992	RT @TravelPixPro1: #ArtNouveau #Paris #France ...	140	0	14	2020-09-26 17:53:24	Twitter for Android	2020-09-26
1	1309913950696468482	RT @TravelPixPro1: Grand Abbey atop the tidal ...	140	0	47	2020-09-26 17:53:17	Twitter for Android	2020-09-26
2	1309913791199621121	RT @UberFacts: Wanderlust (noun):\n\nA strong ...	61	0	1070	2020-09-26 17:52:39	Twitter for iPhone	2020-09-26
3	1309913572772909056	Top news today: @IATA: 'Here's why wearing a m...	135	0	0	2020-09-26 17:51:47	The Tweeted Times	2020-09-26
4	1309912475563970561	The new norm for family portraits! #roadtrip #...	140	0	0	2020-09-26 17:47:25	Instagram	2020-09-26
5	1309912153311371265	RT @TravelPixPro1: #ArtNouveau #Paris #France ...	140	0	14	2020-09-26 17:46:08	Twitter for Android	2020-09-26
6	1309912152820441089	"Wants and Needs and Everything in between"\n....	136	0	0	2020-09-26 17:46:08	Twitter for Android	2020-09-26
7	1309912141118550023	RT @TravelPixPro1: Grand Abbey atop the tidal ...	140	0	47	2020-09-26 17:46:06	Twitter for Android	2020-09-26
8	1309910699104178179	RT @TravelPixPro1: #ArtNouveau #Paris #France ...	140	0	14	2020-09-26 17:40:22	Twitter for Android	2020-09-26
9	1309909710469050376	• Whet your wanderlust. Thrilling tales of an ...	140	0	0	2020-09-26 17:36:26	AskDavid.com Services	2020-09-26

Average Length of Tweets:

Code:

```
printmd("The average length of tweets that were retrieved in **%.0f** characters" %
df['Length'].mean())
print("Top 5 most tweeted character lengths")
df['Length'].value_counts().head()
```

Output:

Building WordCloud for 100 most frequent words:

Code:

```
tweets_text = ' '.join(df['Tweets'].values)
stopwords = {'http', 'https', 'co', 'com', 'in', 'to'}
logomask = np.array(Image.open("twittermask.png"))
plt.figure(figsize = (16,14))

wordcloud = WordCloud(
stopwords=STOPWORDS.union(stopwords),
background_color='black',
mask = logomask,
max_words=100,
width=1800,
height=1400).generate(tweets_text)

plt.imshow(wordcloud)
plt.title("WordCloud of Tweets")
plt.axis('off')
plt.show()
```

Output:

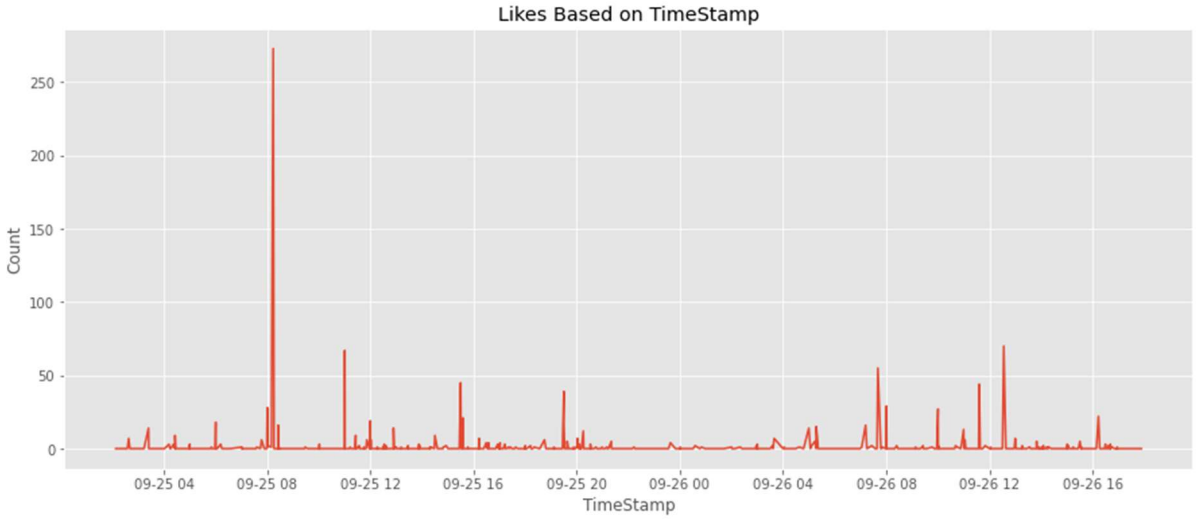

WordCloud of Tweets
to travel
tourism travel
the tidal
helps you
while Hiking
FrenchHistory RT
FlySafe ReadyToFly
RT Cyprus4Holidays
Travel Photography
UberFacts Wanderlust
MontSaintMichel off
nnAoAUpfzU RT
Hiking to
Here why
RT IATA
strong desire
Abbey atop
Tourism Day
the coast
travel wanderlust
desire to
noun strong
rock island
in GrandCanyon
RT UberFacts
wanderlust https
ReadyToFly travel
Wanderlust noun
the bottom
Travel FrenchHistory
tidal rock
travel RT
why wearing
IATA Here
RT TravelPixProl
wearing mask
Northern France
of Northern
mask helps
co nnAoAUpfzU
France Travel
Happy World
Day tourism
island of
World Tourism
to FlySafe
coast of
Photography Adventure
Grand Abbey

Analysing Likes Trend according to Timestamp:

Code:

```
print(df['Likes'].max())
fig, ax = plt.subplots(figsize = (15,6))
sns.lineplot(x = df['Created_at'], y = df['Likes'].values)
plt.title("Likes Based on TimeStamp")
plt.xlabel("TimeStamp")
plt.ylabel("Count")
plt.show()
```

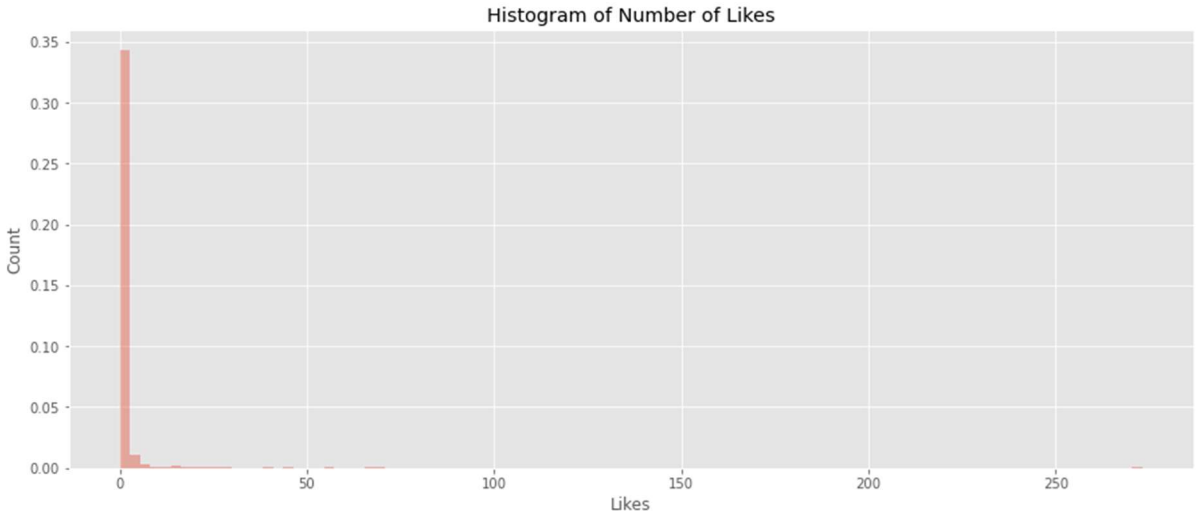
Output:



Code:

```
fig, ax = plt.subplots(figsize = (15,6))
sns.distplot(df['Likes'].values, bins = 100)
plt.title("Histogram of Number of Likes")
plt.xlabel("Likes")
plt.ylabel("Count")
plt.show()
```

Output:

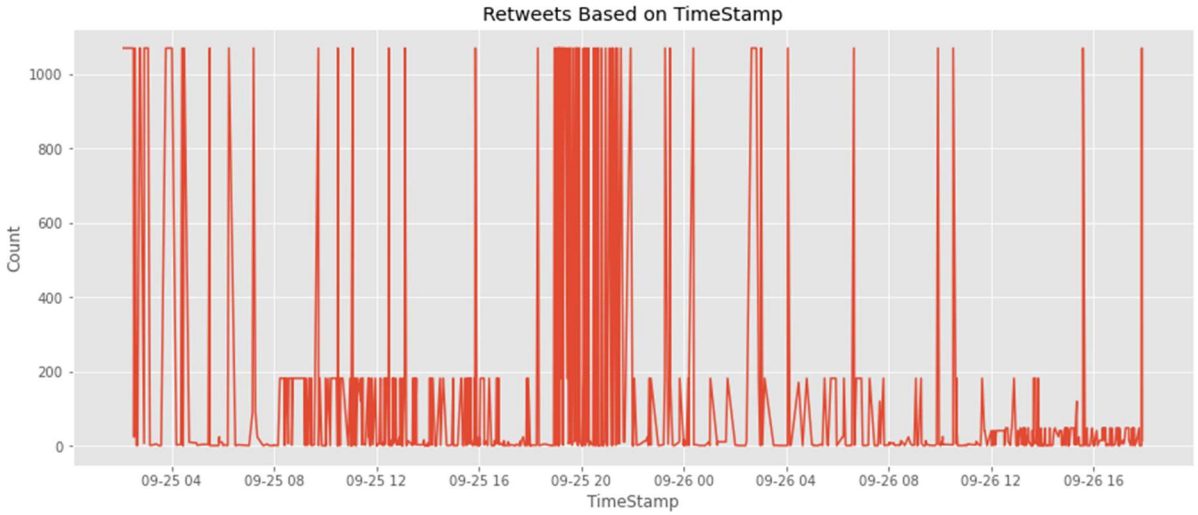


Analysing Retweets Trend according to Timestamp:

Code:

```
print(df['Retweets'].max())
fig, ax = plt.subplots(figsize = (15,6))
sns.lineplot(x = df['Created_at'], y = df['Retweets'].values)
plt.title("Retweets Based on TimeStamp")
plt.xlabel("TimeStamp")
plt.ylabel("Count")
plt.show()
```

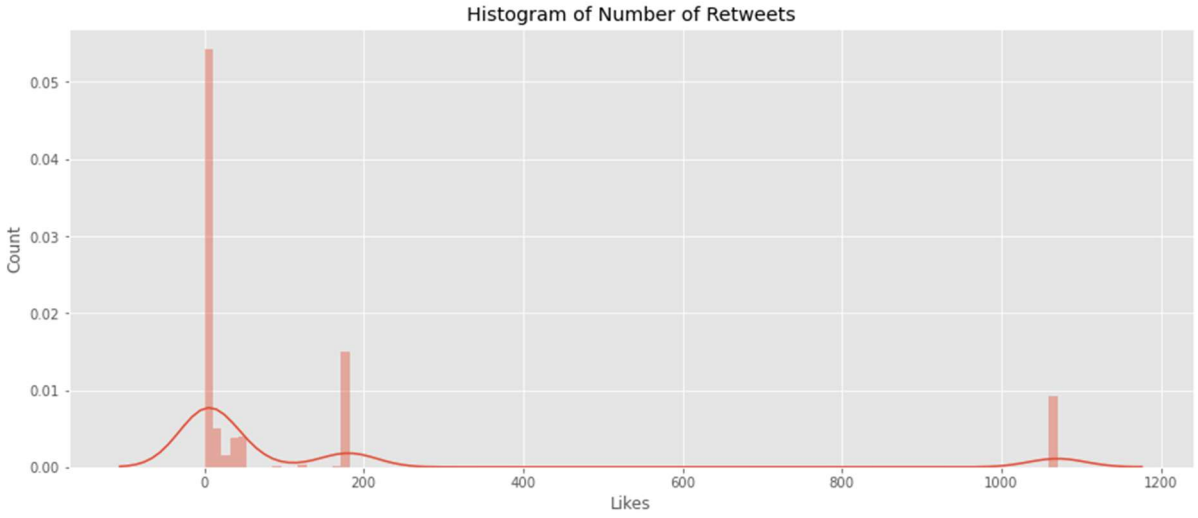
Output:



Code:

```
fig, ax = plt.subplots(figsize = (15,6))
sns.distplot(df['Retweets'].values, bins = 100)
plt.title("Histogram of Number of Retweets")
plt.xlabel("Retweets")
plt.ylabel("Count")
plt.show()
```

Output:

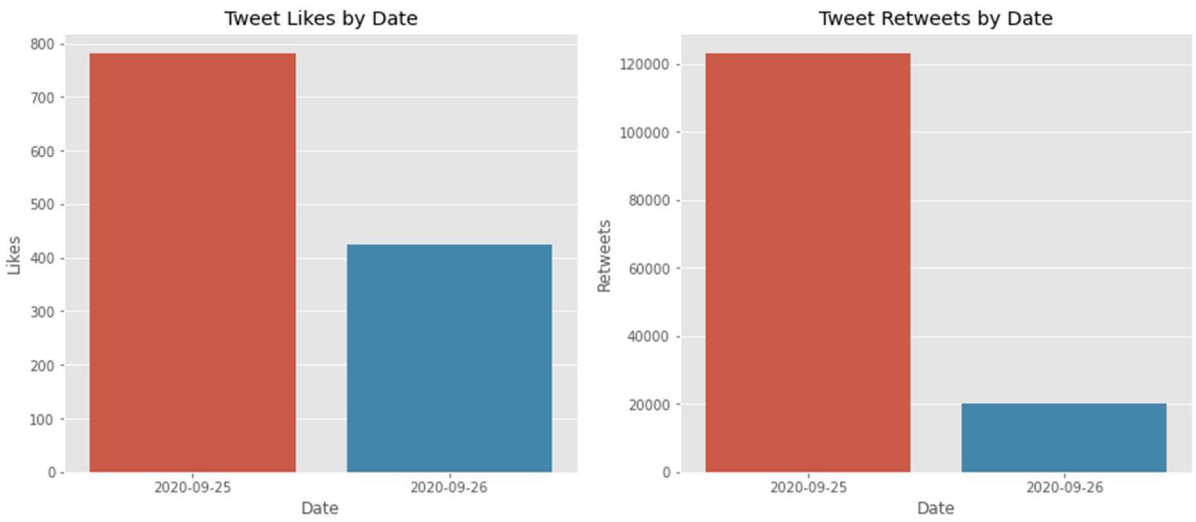


Comparative Analysis of Tweets by Date (Likes & Retweets):

Code:

```
data = df.groupby(df['Date']).sum()
data.drop(['Id', 'Length'], axis = 1, inplace = True)
data.reset_index(inplace = True)
print(data.head())
fig, ax = plt.subplots(1, 2, figsize = (15,6))
#sns.barplot(x='Date', y='value', hue='variable',
#            data=pd.melt(data, ['Date']))
sns.barplot(x = "Date", y = "Likes", data = data, ax = ax[0])
ax[0].set_title("Tweet Likes by Date")
sns.barplot(x = "Date", y = "Retweets", data = data, ax = ax[1])
ax[1].set_title("Tweet Retweets by Date")
plt.show()
```

Output:

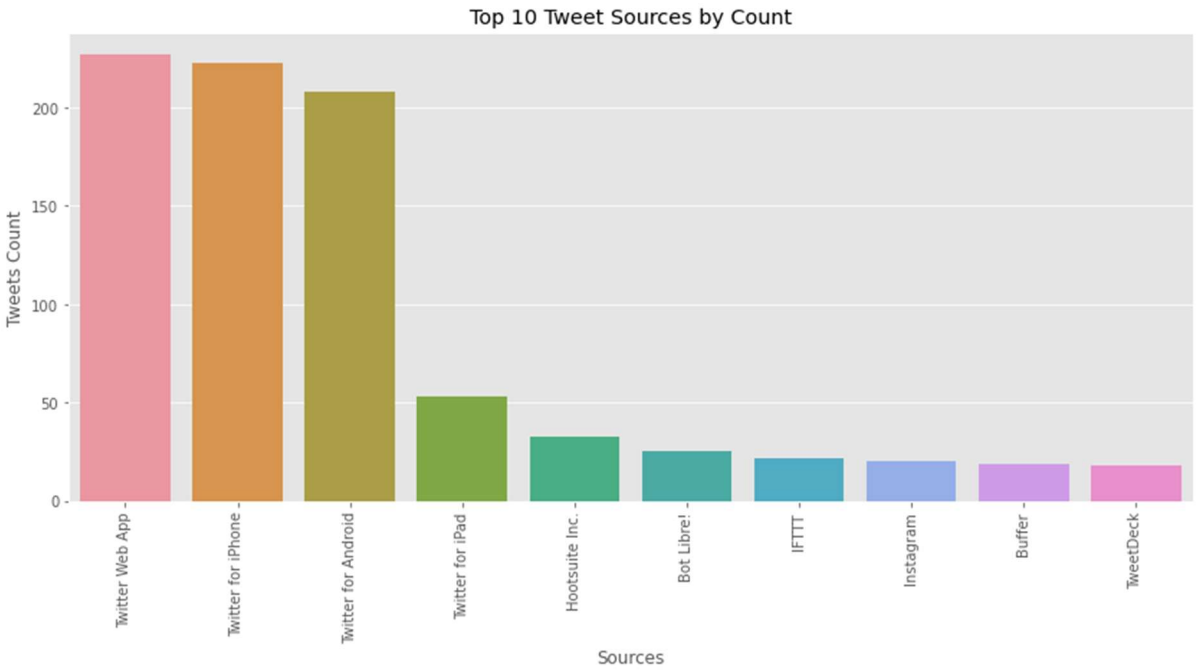


Analysing Top 10 Tweet Sources:

Code:

```
data = df['Source'].value_counts().to_frame().reset_index()
data = data.head(10)
fig, ax = plt.subplots(figsize = (14,6))
sns.barplot(x = data['index'], y = data['Source'])
plt.title("Top 10 Tweet Sources by Count")
plt.xlabel("Sources")
plt.ylabel("Tweets Count")
plt.xticks(rotation = 90)
plt.show()
```

Output:



Named Entity Recognition:

Code:

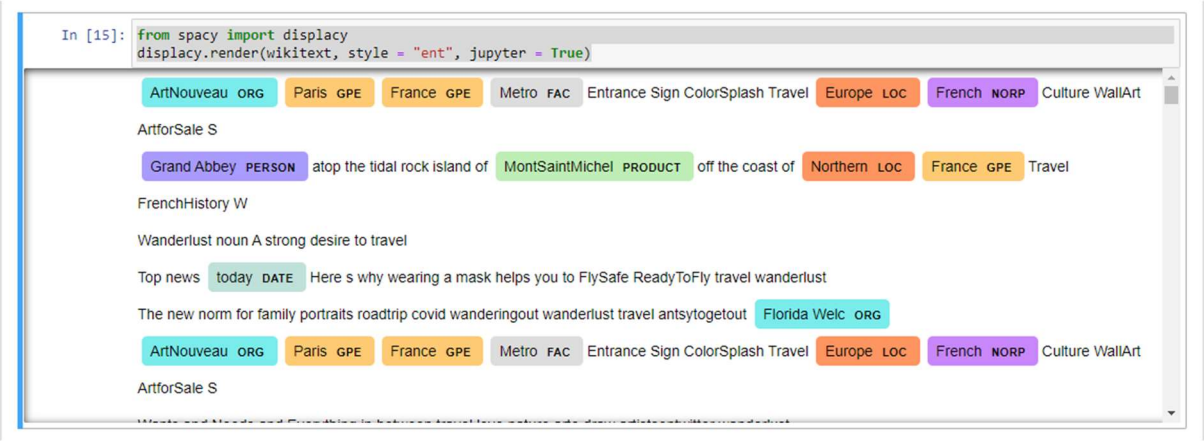
```
import spacy
nlp = spacy.load('en_core_web_sm')

def process_text(tweets):
    """Remove emoticons, numbers etc. and returns list of cleaned tweets."""
    data = tweets
    regex_remove = "(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^RT|http.+?"
    stripped_text = [
        re.sub(regex_remove, ' ',
               tweets).strip() for tweets in data
    ]
    return '\n'.join(stripped_text)
#tweets = '\n'.join(df['Tweets'].values)
tweets = process_text([tweet for tweet in df['Tweets'].values])
wikitext = nlp(tweets)

for word in wikitext.ents:
    print(word.text, word.label_)
```

Output:

Code:

```
from spacy import displacy
displacy.render(wikitext, style = "ent", jupyter = True)
```

Output:



Analysing Sentiments of Tweets (Positive, Negative or Neutral):

Code:

```
def clean_tweet(tweets):
   return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^RT|http.+?",
" ", tweets).split())
def analyze_sentiment(tweets):
   analysis = TextBlob(clean_tweet(tweets))

   if analysis.sentiment.polarity > 0:
      return "Positive"
   elif analysis.sentiment.polarity == 0:
      return "Neutral"
   else:
      return "Negative"
df['Sentiment'] = np.array([analyze_sentiment(tweet) for tweet in df['Tweets']])
df.head()
```

Output:

```
In [16]:
def clean_tweet(tweets):
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^RT|http.+?", " ", tweets).split())
def analyze_sentiment(tweets):
    analysis = TextBlob(clean_tweet(tweets))

    if analysis.sentiment.polarity > 0:
        return "Positive"
    elif analysis.sentiment.polarity == 0:
        return "Neutral"
    else:
        return "Negative"
df['Sentiment'] = np.array([analyze_sentiment(tweet) for tweet in df['Tweets']])
df.head()
```

Out[16]:

	Id	Tweets	Length	Likes	Retweets	Created_at	Source	Date	Sentiment
0	1309913978076884992	RT @TravelPixPro1: #ArtNouveau #Paris #France ...	140	0	14	2020-09-26 17:53:24	Twitter for Android	2020-09-26	Neutral
1	1309913950696468482	RT @TravelPixPro1: Grand Abbey atop the tidal ...	140	0	47	2020-09-26 17:53:17	Twitter for Android	2020-09-26	Positive
2	1309913791199621121	RT @UberFacts: Wanderlust (noun):\n\nA strong ...	61	0	1070	2020-09-26 17:52:39	Twitter for iPhone	2020-09-26	Positive
3	1309913572772909056	Top news today: @IATA: 'Here's why wearing a m...	135	0	0	2020-09-26 17:51:47	The Tweeted Times	2020-09-26	Positive
4	1309912475563970561	The new norm for family portraits! #roadtrip #...	140	0	0	2020-09-26 17:47:25	Instagram	2020-09-26	Positive

Code:

```
data = df['Sentiment'].value_counts().to_frame().reset_index()
data
```

Output:

```
In [17]:
data = df['Sentiment'].value_counts().to_frame().reset_index()
data
```

Out[17]:

	index	Sentiment
0	Positive	476
1	Neutral	459
2	Negative	80

Code:

```
fig, ax = plt.subplots(figsize = (14,6))
sns.barplot(x = "index", y = "Sentiment", data = data)
plt.title("Analysing Sentiment of Tweet Data")
plt.xlabel("Sentiments")
plt.ylabel("Sentiment Count")
plt.show()
```

Output: