# Description: This is a sentiment analysis that parses the tweets from Twitter using Python

# Importing the Libraries

import tweepy

from textblob import TextBlob

from wordcloud import WordCloud

import pandas as pd

import numpy as np

import re

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

# Twitter API Credentials

consumerKey = "DnaXPshCp7DmC9oj6JDS6Vi2C"

consumerSecret = "e9m5e8cjLgrgH3wl0Bv3g8ZyeOPHsrZYeuV0cLANAYyE3Y5Atp"

accessToken = "1036644696620118016-GdFu7cpIGPxHhiTf6Y0vVUAKNifd5n"

accessTokenSecret = "LXrdlmDgJaeVStqerk2xo4EC8u3Zcl1qBVEYslaFKnHEp"

# Create the authentication object

authenticate = tweepy.OAuthHandler(consumerKey, consumerSecret)

# Set the access token and access token secret

authenticate.set\_access\_token(accessToken, accessTokenSecret)

# Create the API obhect while passing in the auth information

api = tweepy.API(authenticate, wait\_on\_rate\_limit=True)

# Extracting 100 tweets from the twitter user

posts = api.user\_timeline(screen\_name = "BillGates", count= 100,lang = "en", tweet\_mode ="extended")

# Print the last 10 tweets from the account

print("Show the 10 recent tweets: \n")

i = 1

for tweet in posts[0:10]:

print(str(i) + ')' + tweet.full\_text + '\n')

i +=1

#Create a dataframe with a column called Tweets

df = pd.DataFrame([tweet.full\_text for tweet in posts], columns = ['Tweets'])

#Show the first 5 rows of data

df.head(10)

#Clean the text

#Create a function to clean the tweets

def cleanTxt(text):

text = re.sub(r'@[A-Za-z0-9]+','', text) #Removed @mentions

text = re.sub(r'#', '', text) #Removed # symbols

text = re.sub(r'RT[\s]+', '', text) #Removed RT

text = re.sub(r'https?:\/\/S+','', text) #Removed the hyper link

text = re.sub(r'https://+', '', text)

return text

df['Tweets']=df['Tweets'].apply(cleanTxt)

#Show the cleaned text

df

# Create a function to get the subjectivity

def getSubjectivity(text):

return TextBlob(text).sentiment.subjectivity

# Create a function to get the polarity

def getPolarity(text):

return TextBlob(text).sentiment.polarity

# Create two new columns

df['Subjectivity'] = df['Tweets'].apply(getSubjectivity)

df['Polarity'] = df['Tweets'].apply(getPolarity)

# Show the new dataframe with new columns

df

#how wel the sentiments are distributed

# Plot Word cloud

allWords= ' '.join([twts for twts in df['Tweets']])

wordCloud = WordCloud(width = 500,height= 300, random\_state =21,max\_font\_size=110).generate(allWords)

plt.imshow(wordCloud, interpolation= "bilinear")

plt.axis('off')

plt.show()

#Create a function to compute the negative, neutral and positive analysis

def getAnalysis(score):

if score<0:

return 'Negative'

elif score == 0:

return 'Neutral'

else:

return 'Positive'

df['Analysis'] =df['Polarity'].apply(getAnalysis)

#Show the dataframe

df

# Print all of the positive tweets

j=1

sortedDF =df.sort\_values(by=['Polarity'])

for i in range(0,sortedDF.shape[0]):

if (sortedDF['Analysis'][i]== 'Positive'):

print(str(j)+ ')' +sortedDF['Tweets'][i])

print()

j+=1

#print the negative tweets

j=1

sortedDF = df.sort\_values(by=['Polarity'],ascending='False')

for i in range(0,sortedDF.shape[0]):

if(sortedDF['Analysis'][i] == 'Negative'):

print(str(j)+')'+sortedDF['Tweets'][i])

print()

j+=1

# Plot the polarity and subjectivity

plt.figure(figsize=(8,6))

for i in range(0,df.shape[0]):

plt.scatter(df['Polarity'][i],df['Subjectivity'][i],color='Blue')

plt.title('Sentimenal Analysis')

plt.xlabel('Polarity')

plt.ylabel('Subjectivity')

plt.show()

# Get the percentage of positive tweets

ptweets = df[df.Analysis=='Positive']

ptweets = ptweets['Tweets']

round( (ptweets.shape[0] / df.shape[0]) \*100,1)

# Get the percentage of negative tweets

ntweets = df[df.Analysis == 'Negative']

ntweets = ntweets['Tweets']

round( (ntweets.shape[0]/ df.shape[0]) \*100,1)

# Show the value counts

df['Analysis'].value\_counts()

# plot and visualize the counts

plt.title('Sentiment Analysis')

plt.xlabel('Sentiment')

plt.ylabel('Counts')

df['Analysis'].value\_counts().plot(kind='bar')

plt.show()