#!/usr/bin/env python

# coding: utf-8

# In[31]:

import tweepy,csv,re

from textblob import TextBlob

import matplotlib.pyplot as plt

# In[33]:

class SentimentAnalysis:

def \_\_init\_\_(self):

self.tweets = []

self.tweetText = []

def DownloadData(self):

# authenticating

consumerKey = 'Q4fPRhZLrtRjDMB1MGMR3UoZI'

consumerSecret = 'mE2mcFhO2wjlahpBooCKG9VeellzS90GVsqdmq5ZVaqczoPbat'

accessToken = '766671490385268736-bmCMC9BwBQllSnegoo08lmsj2Oqsy3a'

accessTokenSecret = '4ibEHYksRhKFW5fHQoD9bqHeLUB5e4wbeGlbMtibMWtGW'

auth = tweepy.OAuthHandler(consumerKey, consumerSecret)

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

api = tweepy.API(auth)

# input for term to be searched and how many tweets to search

searchTerm = input("Enter Keyword/Tag to search about: ")

NoOfTerms = int(input("Enter how many tweets to search: "))

# searching for tweets

self.tweets = tweepy.Cursor(api.search, q=searchTerm, lang = "en").items(NoOfTerms)

# Open/create a file to append data to

csvFile = open('result.csv', 'a')

# Use csv writer

csvWriter = csv.writer(csvFile)

# creating some variables to store info

polarity = 0

positive = 0

wpositive = 0

spositive = 0

negative = 0

wnegative = 0

snegative = 0

neutral = 0

# iterating through tweets fetched

for tweet in self.tweets:

#Append to temp so that we can store in csv later. I use encode UTF-8

self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))

# print (tweet.text.translate(non\_bmp\_map)) #print tweet's text

analysis = TextBlob(tweet.text)

# print(analysis.sentiment) # print tweet's polarity

polarity += analysis.sentiment.polarity # adding up polarities to find the average later

if (analysis.sentiment.polarity == 0): # adding reaction of how people are reacting to find average later

neutral += 1

elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity <= 0.3):

wpositive += 1

elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polarity <= 0.6):

positive += 1

elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polarity <= 1):

spositive += 1

elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polarity <= 0):

wnegative += 1

elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polarity <= -0.3):

negative += 1

elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarity <= -0.6):

snegative += 1

# Write to csv and close csv file

csvWriter.writerow(self.tweetText)

csvFile.close()

# finding average of how people are reacting

positive = self.percentage(positive, NoOfTerms)

wpositive = self.percentage(wpositive, NoOfTerms)

spositive = self.percentage(spositive, NoOfTerms)

negative = self.percentage(negative, NoOfTerms)

wnegative = self.percentage(wnegative, NoOfTerms)

snegative = self.percentage(snegative, NoOfTerms)

neutral = self.percentage(neutral, NoOfTerms)

# finding average reaction

polarity = polarity / NoOfTerms

# printing out data

#typetypeprint("How people are reacting on " + searchTerm + " by analyzing " + str(NoOfTerms) + " tweets.")

print()

print("General Report: ")

if (polarity == 0):

print("Neutral")

elif (polarity > 0 and polarity <= 0.3):

print("Weakly Positive")

elif (polarity > 0.3 and polarity <= 0.6):

print("Positive")

elif (polarity > 0.6 and polarity <= 1):

print("Strongly Positive")

elif (polarity > -0.3 and polarity <= 0):

print("Weakly Negative")

elif (polarity > -0.6 and polarity <= -0.3):

print("Negative")

elif (polarity > -1 and polarity <= -0.6):

print("Strongly Negative")

print()

print("Detailed Report: ")

print(str(positive) + "% people thought it was positive")

print(str(wpositive) + "% people thought it was weakly positive")

print(str(spositive) + "% people thought it was strongly positive")

print(str(negative) + "% people thought it was negative")

print(str(wnegative) + "% people thought it was weakly negative")

print(str(snegative) + "% people thought it was strongly negative")

print(str(neutral) + "% people thought it was neutral")

self.plotPieChart(positive, wpositive, spositive, negative, wnegative, snegative, neutral, searchTerm, NoOfTerms)

def cleanTweet(self, tweet):

# Remove Links, Special Characters etc from tweet

return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ / \S +)", " ", tweet).split())

# function to calculate percentage

def percentage(self, part, whole):

temp = 100 \* float(part) / float(whole)

return format(temp, '.2f')

def plotPieChart(self, positive, wpositive, spositive, negative, wnegative, snegative, neutral, searchTerm, noOfSearchTerms):

labels = ['Positive [' + str(positive) + '%]', 'Weakly Positive [' + str(wpositive) + '%]','Strongly Positive [' + str(spositive) + '%]', 'Neutral [' + str(neutral) + '%]',

'Negative [' + str(negative) + '%]', 'Weakly Negative [' + str(wnegative) + '%]', 'Strongly Negative [' + str(snegative) + '%]']

sizes = [positive, wpositive, spositive, neutral, negative, wnegative, snegative]

colors = ['yellowgreen','lightgreen','darkgreen', 'gold', 'red','lightsalmon','darkred']

patches, texts = plt.pie(sizes, colors=colors, startangle=90)

plt.legend(patches, labels, loc="best")

# plt.title('How people are reacting on ' + searchTerm + ' by analyzing ' + str(noOfSearchTerms) + ' Tweets.')

plt.axis('equal')

plt.tight\_layout()

plt.show()

if \_\_name\_\_== "\_\_main\_\_":

sa = SentimentAnalysis()

sa.DownloadData()