twitter mining w mongoDB and NLTK sentiment

August 18, 2020

```
[ ]: """@author: BBE - Brian Hogan brian.p.hogan@alumni.harvard.edu"
     Objective: Generate New York State twitter traffic chatter building profile
                of good, bad, and ugly traffic pattern days.
     Method:
         Obtain: Mongodb grab tweets over month across 1 to n twitter handles.
         Scrub:  Pandas dataframe.
         Analyze: NLTK w Vadar for +/- neu and compound scoring
         Visualize: Wordcloud
         Predict: Naive Bayes Sentiment Analysis
     """
     import tweepy
     import json
     import pymongo
     import pandas as pd
     from bson.json_util import dumps  #from dn_fn.py for save & load to database

     CONSUMER_KEY = 'GFuEK46t.....'  #BBE twitter keys...
     CONSUMER_SECRET = 'sWsBF6S9EOPD.....'
     OAUTH_TOKEN = '989685004832792578-3....'
     OAUTH_SECRET = 'zRm1pwVBQOYX4b8...'
```

```
[ ]: """ Functions"""
     """=> twitter login        """
     def oauth_login():
       auth = tweepy.OAuthHandler(CONSUMER_KEY,CONSUMER_SECRET)
       auth.set_access_token(OAUTH_TOKEN,OAUTH_SECRET)
       tweepy_api = tweepy.API(auth)
       if (not tweepy_api):        #error out
           print ("Problem Connecting to API with OAuth")
       return tweepy_api  #api object to twitter functions
     def appauth_login(): #login to twitter w extended rate limiting
       auth = tweepy.AppAuthHandler(CONSUMER_KEY,CONSUMER_SECRET)
       #auth.set_access_token(OAUTH_TOKEN,OAUTH_SECRET) #needed for one test so put␣
      ↪back in
       tweepy_api = tweepy.API(auth, wait_on_rate_limit=True,␣
      ↪wait_on_rate_limit_notify=True)
       if (not tweepy_api):  #let user know if api error
           print ("Problem Connecting to API with AppAuth")
       return tweepy_api     #api object to twitter functions

     """=> connection test """
     if __name__ == '__main__':  #test connection
       tweepy_api = oauth_login()
       print ("Twitter  Authorization OK :", tweepy_api)
       tweepy_api = appauth_login()
       print ("Twitter  Authorization OK :", tweepy_api)

     def simple_search(api, query, max_results=20):  #ASYNCH 8.4
         # the first search initializes a cursor, stored in the metadata results,
       #   that allows next searches to return additional tweets
       search_results = [status for status in tweepy.Cursor(api.search, q=query).
     ↪items(max_results)]
       tweets = [tweet._json for tweet in search_results]
       return tweets

     """asynch dn_fn.py  """
     def save_to_DB(DBname, DBcollection, data):
         client = pymongo.MongoClient('localhost', 27017) #connect to server
         """change names to lowers case because they are not case senstitive
         and remove special characteers like hashtask and spaces   """
         DBname = DBname.lower()
         DBname = DBname.replace('#', '')
         DBname = DBname.replace(' ', '')
         DBcollection = DBcollection.lower()
         DBcollection = DBcollection.replace('#', '')
         DBcollection = DBcollection.replace(' ', '')
         db = client[DBname]
         collection = db[DBcollection]
         collection.insert_many(data)
         print("\nSaved", len(data), "documents to DB", DBname, DBcollection)

     """dn_fn.py  - used to get existing data; return as json objects"""
     def load_from_DB(DBname, DBcollection):
         client = pymongo.MongoClient('localhost', 27017)
         client.list_database_names   # ISSUE HERE W DEPRECTATION again...5-31-19
         db = client[DBname]
         collection = db[DBcollection]  #find collection and load docs
         docs = collection.find()
         docs_bson = list(docs)
         docs_json_str = [dumps(doc) for doc in docs_bson]
         docs_json = [json.loads(doc) for doc in docs_json_str]
         return docs_json
```

```
[ ]: """Get Tweets from MongoDB and store in Panda Frame"""
     """8.4 """
     if __name__ == '__main__':
         print("Program collects twitter tweets generating wordclouds,freqency, and␣
      ↪sentiment; requires a MongoDB.")
         """ask user for hashag, database and dbcollection so not hardcoded"""
         print(".............................................................................
      ↪.......................")
         print("Please select one of the Following Artificial Intelligence Twitter␣
      ↪experts for this program.")
         print(".............................................................................
      ↪.......................")
         print("====>@mfordfuture<===========")␣
      ↪#,@romanyam,@cynthiabreazel,@petitegeek,@erickorvitz,@FLIxrisk,@FHIOxford")
         print(".............................................................................
      ↪.......................")
         query = input("Enter Twitter hashtag (#, @ etc): ")
         print(".............................................................................
      ↪......................")
         num_tweets = input("Enter max # of tweets to grab: ")
         num_tweets = int(num_tweets)
         print(".............................................................................
      ↪......................")
         DBname = input("Enter mongodb name (this query doesnt overwrite old data):␣
      ↪")
         print(".............................................................................
      ↪......................")
         DBcollection = input("Please Mongo filename to store within your database:␣
      ↪")

         api = appauth_login()  #login to thr api
         #api = oauth_login() <--uncomment if sutich to appauth to avoid rate limit

         result_tweets = simple_search(api, query, max_results=num_tweets)
         print ('Number of result tweets imported: ', len(result_tweets)) #let user␣
      ↪know success

         save_to_DB(DBname, DBcollection, result_tweets)  #save to database

         """OK now that we have the tweets were going to do some counting"""
         print('Tweet summary statistics are next. Refer to the tweet-datatable.txt␣
      ↪'\
               'output file in the folder run for full tweet dataset collected.')
         #get results from mongo db
         tweet_results = load_from_DB(DBname.lower(), DBcollection.lower())
         tweet_df = pd.DataFrame() #initiate an empty dataframe to fill
```

```
    tweet_df['id']=[tweet['id'] for tweet in tweet_results] #collect data
    tweet_df['language']=[tweet['lang'] for tweet in tweet_results]
    tweet_df['location']=[tweet['user']['location'] for tweet in tweet_results]
    tweet_df['screen_name']=[tweet['user']['screen_name'] for tweet in␣
↪tweet_results]
    tweet_df['followers']=[tweet['user']['followers_count']for tweet in␣
↪tweet_results]
    tweet_df['tweet']=[tweet['text']for tweet in tweet_results]
    df2 = pd.DataFrame(tweet_df)
    #what data is provided to customer
    print("Tweet columns in the csv output reports include: ",df2.columns)

    #import summary statistics
    #print("What are unique total counts, unique values, top values of tweets? :
↪{}".format(df2.describe(include=['object'])))
    #this meta data could be parsed - need to learn how to execute
    print("..........................................................................
↪.......................")
    print("Tweet import metadata :{}".format(df2.sum()))  #metadata of all␣
↪tweets
    print("..........................................................................
↪.......................")
    output_tweet_data = df2.describe(include=['object']) #output detail to csv
    output_tweet_data.to_csv("Final_project_Tweets_Dataframe_BBE.txt",␣
↪index=True)
    #average followers
    #print("What are the average total tweet followers :{}".format(df2.
↪describe()))
    output_tweet_data = df2  #output the total tweet datatable
    output_tweet_data.to_csv("Final_project_Tweets_BBE.txt", index=True)

    """===WORD FREQUENCY=================="""
    import nltk  #for natural language modeling
    nltk.download('stopwords')
    client = pymongo.MongoClient('localhost', 27017)
    client.list_database_names()   # ISSUE HERE W DEPRECTATION again...5-31-19
    #client.list_database_names()
    #project is 652(cant use - made bk, bkf the file)
    """============="""
    db = client.DBname #client.bk
    db.collection_names()  #get the collection name
    collection = DBcollection       #db.bk_f  #find collection and load docs
    """================"""
    """ THE FOLLOWING is what you use to go get the tweets and carry on"""
    docs = load_from_DB(DBname, DBcollection)
    doclist = [tweet for tweet in docs]
    #len(doclist)
    def print_tweet_data(tweets):    #sample loop to read through tweets
        for tweet in tweets:
            print('\nDate: ',tweet['text'])
            #print_tweet_data(doclist[:1])
    """important to build the message list"""
    msglist = [doc['text'] for doc in doclist if 'text' in doc.keys()]
    #len(msglist)
    """tokens are a summary of individual words"""
    all_tokens = [tok for msg in msglist for tok in nltk.word_tokenize(msg)]
    #len(all_tokens)
    #all_tokens[:10]
    msgtweet = nltk.FreqDist(all_tokens) #build the frequency of tokenized words
    #msgtweet.most_common(15)
    all_tokens = [tok.lower() for msg in msglist for tok in nltk.
 ↪word_tokenize(msg)]
    #all_tokens[:10]
    nltk_stopwords = nltk.corpus.stopwords.words('english')#remove nonvalue add␣
 ↪words
    #len(nltk_stopwords)
    import re
    def alpha_filter(w):
        pattern = re.compile('^[^a-z]+S')  #need to expand on filter for more
        if (pattern.match(w)):              #symbols
            return True
        else:
            return False
    token_list = [tok for tok in all_tokens if not alpha_filter(tok)]
    #token_list[:30]
    msgtweet = nltk.FreqDist(token_list)
    top_words=msgtweet.most_common(20) #words used most in the tweets
    #words={} #make a dictionary  ====>move to dictionary in future
#    print("Twitter Traffic Chatter Most Common Words/Frequency")
#    for word, freq in top_words:   #print the most commone words
#        print("Word:",word,freq)
    # close the database connection
    client.close()
```

[ ]:
```
"""=> TWEET PUll =>STRIP & CLEAN =>STOPWORDS =>SENTIMENT => TOKENIZE GRID?
    1) Tweet Pull from Mongo DB:    ( 652 )
        (tweet pull manual from top 10 AI twitter hashtags from newsarticle)
    2) Strip & Clean:
    3) Stopwords Remove:            ibid
    4) Sentiment:                   ibid
    5) Tokenize Pos/Neg grid        ibid
    6) POS/NEG Word Clouds                              """
```

```
"""=============================================================================
==> 1 ) Tweet Pull (from Mongo Database)
============================================================================="""
#Getting the msglist from Mongo database - raw with uncleaned garbage!
sentences = msglist  #msglist = [doc['text'] for doc in doclist if 'text' in␣
 ↪doc.keys()]
mytweets = []
for sentence in sentences:  #from Dr. GAtes
    mytweets.append(sentence)
#print(mytweets)
"""=============================================================================
==> 2 ) Strip & Clean  (w emojies)
============================================================================="""
import re
#ONline search to remove emojois as don't know how to handle all that yet
emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags=re.UNICODE)
emoji_pattern2 = re.compile(
        u"(\ud83d[\ude00-\ude4f])|"  # emoticons
        u"(\ud83c[\udf00-\uffff])|"  # symbols & pictographs (1 of 2)
        u"(\ud83d[\u0000-\uddff])|"  # symbols & pictographs (2 of 2)
        u"(\ud83d[\ude80-\udeff])|"  # transport & map symbols
        u"(\ud83c[\udde0-\uddff])"  # flags (iOS)
        "+", flags=re.UNICODE)
single_parenthsis = re.sub("'","","A single ' char")
myfinaltweets=[]
for line in mytweets:
    #print("The next line is, ",line)
    line = line.rstrip()  #strip whitespace from the end
    line = re.sub('[/:?;!@#$-.]','',line) #adding colons and question mark
    line = re.sub('[...]','',line) #adding ellipsis
    # line = re.sub('[\\']','',line) #adding ellipsis
    line = re.sub("'","",line)

    line = re.sub('\s+',' ',line).strip() #remove extra whitespace
    line = line.strip("\n") #remove new line
    line = line.lower()
    line = emoji_pattern.sub(r'', line) #emoji_pattern.sub(r'', text)) # no␣
 ↪emoji
    line = emoji_pattern2.sub(r'', line)  #stil have some trouble icons
    #now remove for other characters not being pulled out!
    line = re.sub('https','',line)
    line = re.sub('mfordfuture','',line)
    line = re.sub('kdnuggets','',line)
    #print("Now the line is: ",line)
    myfinaltweets.append(line)
print("...........................................................................
 ↪...................")
print(".......Total tweets in analysis :",len(myfinaltweets) )
print("...........................................................................
 ↪...................")
#myfinaltweets[:1]
#myfinaltweets
#print(myfinaltweets)
finaltweetsjoined = "".join(myfinaltweets)
```

[ ]:
```
"""=================================================================
==> 3) StopWords & Wordcloud
================================================================="""
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
#text="This is any sentence of text. It can have punctuation, CAPS!, etc."
tokenized_word=word_tokenize(finaltweetsjoined)
#len(tokenized_word)
stop_words=set(stopwords.words("english"))
filtered_tweet_words=[]
for w in tokenized_word:
    #print(w)
    if w not in stop_words:
        filtered_tweet_words.append(w)

#print("Tokenized text:",tokenized_word)
#print(filtered_tweet_words)
print("...........................................................................
 ↪...................")
print("# of Unfiltered Word Tokens in Tweets Pulled:",len(tokenized_word))
print("...........................................................................
 ↪...................")
print("# of Filtered Word Token, w No StopWords, in Tweets Pulled:
 ↪",len(filtered_tweet_words))
print("...........................................................................
 ↪...................")
"""still have bad characters that cant export ! """
    #tweetfile= open('HW1_tweets.txt','r')
    #with open('HW1_tweets.txt',"w") as f:
    #    for item in filtered_text:
    #        f.write("%s\n" %item)
    #tweetfile.close()
```

```
"""=================================================================
==> 3) word Frequency
=================================================================="""
mostfrequentwords = nltk.FreqDist(filtered_tweet_words)
top_words=mostfrequentwords.most_common(50) #words used most in the tweets
#top_words
#len(top_words)
#well can make one but still not helping with getting homework done!
import pandas as pd
DF_topwords = pd.DataFrame(top_words)
print("..........................................................................
↪...................")
print("................50 Top Words from Tweets............... \n",DF_topwords)␣
↪        #print("HW1-736- AI Most Frequent Words... \n")
print("..........................................................................
↪...................")
output_tweet_data = DF_topwords  #output the total tweet datatable
print("Top Words Stored to Documents as : BBE_HW1_ist736_Tweet_Word_Frequency.
↪txt")
output_tweet_data.to_csv("BBE_HW1_736_Tweet_Frequency.txt", index=True)

"""=================================================================
==> 3) WordCloud (join tweets back together to create a wordcloud from grp)
=================================================================="""
wordcloud_items=[] #make a dictionary  ====>move to dictionary in future
for word, freq in top_words:   #print the most commone words
        #print("Word:",word,freq)
        wordcloud_items.append(word)
#print(wordcloud_items)
import numpy as np
import pandas as pd
from PIL import Image
#>conda install -c conda-forge wordcloud
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from wordcloud import WordCloud, ImageColorGenerator

import matplotlib.pyplot as plt
joinedfilteredtweets = " ".join(filtered_tweet_words)  ##  join
#print(joinedfilteredtweets)  # lower max_font_size, change the maximum number␣
↪of word and
    #lighten the background:"""                              #white, purple,␣
↪etc
wordcloud = WordCloud(max_font_size=50, max_words=100,␣
↪background_color="lightblue").generate(joinedfilteredtweets)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

[ ]:
```
"""=====================================================================
    SENTIMMENT ANALYUSIS => VADAR
======================================================================="""
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#this analyzer expects a list of text sentences
# provides positive, negative, and neutral. It also gives a compound
#score, which should be the overall sentiment, ranging -1 to +1 (positive).

sentences = myfinaltweets
#myfinaltweets
#len(myfinaltweets)
sid=SentimentIntensityAnalyzer()
score = []
for sentence in sentences:
    #print(sentence)
    ss=sid.polarity_scores(sentence)
    for k in sorted(ss):
        #print('{0}:{1},'.format(k,ss[k]),end=' ') #single quote not a double
        score.append(ss)
#creteing a data frame with all the sentiment scores so I can aggregate them
import pandas as pd
score_result = pd.DataFrame(score)
score_result  #uncomment it want to see the dataframe
    #      compound    neg    neu    pos
    #0       0.8104  0.076  0.557  0.368
"""score the whole dataframe for a net pos/neg result"""
#https://cmdlinetips.com/2018/12/how-to-loop-through-pandas-rows-or \
 #                                  -how-to-iterate-over-pandas-rows/
pos_value=0
neg_value=0
no_count=0
pos_count=0
neg_count=0
for index, row in score_result.iterrows():
    pos_value=0
    neg_value=0
    pos_value = (index,row['pos'])
    neg_value = (index,row['neg'])
    if pos_value != neg_value:
        if (pos_value > neg_value):
            pos_count = pos_count+1
        if (pos_value <= neg_value):
            neg_count = neg_count+1
    if pos_value == neg_value:
            no_count=no_count+1
no_count
pos_count
neg_count
print(".........Vadar Sentiment Score: Positive, Negative, Neutral.........")
print(pos_count/len(score_result)),print(neg_count/
 ↪len(score_result)),print(no_count/len(score_result))
if((pos_count/len(score_result))>neg_count/len(score_result)):
    print(".....................OVERALL VADAR SENTIMENT POSITIVE ! ")
if((pos_count/len(score_result))<=neg_count/len(score_result)):
    print(".....................OVERALL VADAR SENTIMENT NEGATIVE ! ")
"""===================================================================
    SENTIMMENT ANALYUSIS => naiveBayes
 ==================================================================== """
import operator
split_docs_in_half = int(round(len(myfinaltweets)/2,0))
split_docs_in_half
a=operator.__index__(split_docs_in_half)
#a
bbe_subj_docs=[] #need a list to put the tuples in to run nB sentiment
mydict = {}
for line in myfinaltweets[0:a]:  #this will print eacdh individual line
    #text = line
    words = line.split()
    mydict=(words,'subj')
    bbe_tuple = tuple(mydict)
    bbe_subj_docs.append(bbe_tuple)
    #t=tuple(words,)
    #print(words) #words
   #=> ['last', 'day', 'of', 'cfiminds', 'is', 'just', 'kicking', 'off',␣
 ↪'today's', 'theme', 'is', 'ai', 'and', 'intelligence', 'augmentation',␣
 ↪'we're', 'starting', 'with…', 'tconizrsbb0fj']
#print(bbe_subj_docs[0])
n=bbe_subj_docs[0]
a=a+1
end_doc_row_pointer=int(len(myfinaltweets))
b=operator.__index__(end_doc_row_pointer)
bbe_obj_docs=[] #need a list to put the tuples in to run nB sentiment
mydict = {}
for line in myfinaltweets[a:b]:  #this will print eacdh individual line
    #text = line
    words = line.split()
    mydict=(words,'obj')
    bbe_tuple = tuple(mydict)
    bbe_obj_docs.append(bbe_tuple)
#len(bbe_subj_docs)#len(bbe_obj_docs)
```

```
#bbe_subj_docs#bbe_obj_docs
subj_docs =bbe_subj_docs
obj_docs = bbe_obj_docs
```

[ ]:
```
"""======================================================================
    SENTIMMENT ANALYUSIS => VADAR
========================================================================"""
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#this analyzer expects a list of text sentences
# provides positive, negative, and neutral. It also gives a compound
#score, which should be the overall sentiment, ranging -1 to +1 (positive).

sentences = myfinaltweets
#myfinaltweets
#len(myfinaltweets)
sid=SentimentIntensityAnalyzer()
score = []
for sentence in sentences:
    #print(sentence)
    ss=sid.polarity_scores(sentence)
    for k in sorted(ss):
        #print('{0}:{1},'.format(k,ss[k]),end=' ') #single quote not a double
        score.append(ss)
#creteing a data frame with all the sentiment scores so I can aggregate them
import pandas as pd
score_result = pd.DataFrame(score)
score_result  #uncomment it want to see the dataframe
    #      compound    neg     neu     pos
    #0       0.8104  0.076  0.557  0.368
"""score the whole dataframe for a net pos/neg result"""
#https://cmdlinetips.com/2018/12/how-to-loop-through-pandas-rows-or \
 #                                    -how-to-iterate-over-pandas-rows/
pos_value=0
neg_value=0
no_count=0
pos_count=0
neg_count=0
for index, row in score_result.iterrows():
    pos_value=0
    neg_value=0
    pos_value = (index,row['pos'])
    neg_value = (index,row['neg'])
    if pos_value != neg_value:
        if (pos_value > neg_value):
            pos_count = pos_count+1
        if (pos_value <= neg_value):
            neg_count = neg_count+1
    if pos_value == neg_value:
            no_count=no_count+1
no_count
pos_count
neg_count
print(".........Vadar Sentiment Score: Positive, Negative, Neutral.........")
print(pos_count/len(score_result)),print(neg_count/
 ↪len(score_result)),print(no_count/len(score_result))
if((pos_count/len(score_result))>neg_count/len(score_result)):
    print("......................OVERALL VADAR SENTIMENT POSITIVE ! ")
if((pos_count/len(score_result))<=neg_count/len(score_result)):
    print("......................OVERALL VADAR SENTIMENT NEGATIVE ! ")
```