Twitter Tweet Mining w MongoDB, NLTK, & Naive Bayes

March 4, 2020

[ ]:
```
"""THANK YOU FOR REVIEWING THIS NOTEBOOK
   Please email me so I know your a real person and can help with further code.
   Remainder of this file builds a Panda DF, analyzes basic sentiment with␣
 ↪Vadar finishing with Naive Bayes Prediction
   Much apprecated ~BBE
   brian.p.hogan@alumni.harvard.edu"""
```

[ ]:
```
"""Created on Wed Jul 10 15:06:41 2019
@author: BBE - Brian Hogan
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
    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
    db = client[DBname]
    collection = db[DBcollection]  #find collection and load docs
```

```
    docs = collection.find()
    return docs_json
```