Analyzing Tweets using Python and Hadoop Ecosystem

April 21, 2019

Jake Ardoin – jardo16@lsu.edu

Introduction

Motivation

People put much more information about themselves on social media than they realize. Someone who just uses twitter with the base settings would probably be surprised to learn they're than a stranger could use it to find out things such as: their usual schedule, where they work, where they live, when they are away from their home, and much more. Social media is a wealth of information about people, all given away for free, sometimes unknowingly. I wanted to familiarize myself with the various techniques for obtaining data from one of the largest and most accessible social networks: Twitter. I chose examples to see just how much data I could access easily. I that focused on places instead of people, but the techniques are the same and could easily be adjusted for a variety of uses.

Project Description

I acquired a twitter account and used python to mine it for tweets. I used up all of the paid tiers and end the end just had the free tiers running in the back group while I worked on the project during the week. Then I took them to Hadoop to set up a small, 2 node cluster using my pc and laptop. Since I couldn't get the internet to work between them and there wasn't any time left to fix it, I took all the data I had and used Panda's and Text Blob to do the sentiment analysis in Jupyter Notebooks instead of with map-reduce.

Division of Roles

Me. All of them.

System Design

Big Data Frameworks

I'm currently running a Python 3.5 Environment with Anaconda to pull data which is transferred to a Hadoop Cluster running on 2 VM's. Apache Pig is used to analyze the data. And

so was Spark, and a bunch of different VM setups. For some reason I couldn't keep it from freezing when I would try to scan these tweet json's in a VM.

Chosen Datasets

I applied for a twitter development account and once I got that I used it to access their API and to mine tweets from them. The data sets I got were as follows:

Full Archive Search : 100 tweets a month going back about 5 years and Full Archive searches were out.

Month Archive : A significant number of tweets about the term Computer Science. I searched it within a mile of LSU and a Mile of ULL to compare if people at those schools tend to speak well about the program or not.

Standard (1 week) archive: When I ran out of searches for the other archives I went and searched the standard archive for two different movies, and compared what people were saying about them to see which one was liked more.

Also, all tweets are unique, there are no retweets, every single one was posted as a status or in response to a status without being retweeted,

Detailed Description of Components

Create heading for each component

Twitter API – had to apply to twitter to gain access to their API., limited searches unless you are willing to pay.

Ubuntu, a lightweight Ubuntu distribution that I installed Hadoop and Java on when I was trying to make the cluster.

Also, the newer realse of the cloudera VM to try and use the newer Hue to do the analysis, it didn't work well enough for me though.

The rest is all python and Jupyter Notebooks.

Evaluation & Test

Software Manual

You will need to install Jupyter Notebooks to run the majority of the program. There are a few different once, all of them are well documented except for the one I used to do the final analysis. So, if you have something where you can pull up Jupyter Notebook you will have no trouble seeing how I acquired the data as its separated into cells for each search I did.

Test Environment and Test Cases

The testing environment was the same as much of the rest of the project. I used python on Jupyter Notebooks because of how well it handled the large files of tweets. Since they were pure text files, I was able to test all the files I had downloaded. The only ones that were left out of the final test were the full archive ones that I labeled badly, and so I just added more standard search experiments to make up for it.

Conclusion

I was surprised by just how much data and how quickly it could be accessed. I also never realized how lightweight json files are. I thought I'd have gigs worth of data, but by the time Iran out of the paid searches I barely had a couple 100 megs. I also found out that even thoughthese seem to be popular activities for programmer's, search tweets and using Hadoop ecosystemare both poorly documented. It's hard to figure out some simple things like getting two machinesto talk to each other, but the right answer is buried beneath so much junk. Anyway, this was agreat project and I can't wait until the summer rolls around so I can set up a physical cluster andbe done with it.

The next pages are some of my results and some of my calculations. The full files are included in the folder, but I believe my comments on the program portions I used for downloading the tweets are helpful for explaining how I approached this project.

In [1]:
```
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, f1_score
from sklearn.metrics import classification_report
import json
import pandas as pd
from textblob import TextBlob
```

In [2]:
```
hellboylist = []
shazamlist = []
lsucsclist = []
ullcsclist = []
hellboy = 'standard_hellboy_tweets.txt'
shazam = 'standard_shazam_tweets.txt'
lsucsc = '30day_lsucsc_tweets.txt'
ullcsc = '30day_ullcsc_tweets.txt'
blob = TextBlob("")
```

In [3]:
```
hellpol = 0
n = 0;
with open('standard_hellboy_tweets.txt', encoding='utf-8') as file:
    all_data = json.load(file)
    for each_dictionary in all_data:
        tweet_id = each_dictionary['id']
        whole_tweet = each_dictionary['text']
        for x in whole_tweet:
            hellpol = TextBlob(x).sentiment.polarity
            n = n+1

        hellboylist.append({'tweet_id': tweet_id,
                            'tweet text': whole_tweet,
                           })
        hellboy_df = pd.DataFrame(hellboylist, columns = ['tweet_id', 'wh
ole_tweet'])
```

In [4]:
```
shapol = 0
n = 0;
with open(shazam, encoding='utf-8') as file:
    all_data = json.load(file)
    for each_dictionary in all_data:
        tweet_id = each_dictionary['id']
        whole_tweet = each_dictionary['text']
        for x in whole_tweet:
            shapol = TextBlob(x).sentiment.polarity
            n = n+1

        shazamlist.append({'tweet_id': tweet_id,
                            'tweet text': whole_tweet,
                           })
        shazam_df = pd.DataFrame(shazamlist, columns = ['tweet_id', 'whol
e_tweet'])
print(f'The polarity of these tweets averages {shapol / n}')
```

```
The polarity of these tweets averages 0.0
```

In [5]:
```
ullpol = 0
n = 0;
with open(ullcsc, encoding='utf-8') as file:
    all_data = json.load(file)
    for each_dictionary in all_data:
        tweet_id = each_dictionary['id']
        whole_tweet = each_dictionary['text']
        for x in whole_tweet:
            ullpol = TextBlob(x).sentiment.polarity
            n = n+1

        ullcsclist.append({'tweet_id': tweet_id,
                           'tweet text': whole_tweet,
                          })
        ullcsc_df = pd.DataFrame(ullcsclist, columns = ['tweet_id', 'whol
e_tweet'])
print(f'The polarity of these tweets averages {ullpol / n}')
```

```
The polarity of these tweets averages 0.0
```

In [6]:
```
lsupol = 0
n = 0;
with open(lsucsc, encoding='utf-8') as file:
    all_data = json.load(file)
    for each_dictionary in all_data:
        tweet_id = each_dictionary['id']
        whole_tweet = each_dictionary['text']
        for x in whole_tweet:
            lsucpol = TextBlob(x).sentiment.polarity
            n = n+1

        lsucsclist.append({'tweet_id': tweet_id,
                           'tweet text': whole_tweet,
                          })
        lsucsc_df = pd.DataFrame(lsucsclist, columns = ['tweet_id', 'whol
e_tweet'])

print(f'The polarity of these tweets averages {lsupol / n}')
```

```
The polarity of these tweets averages 0.0
```

In [ ]:

In [ ]:

In [ ]:

In [6]:

```
-0.2
```

In [ ]:

In [1]:
```
from TwitterAPI import TwitterAPI, TwitterPager
import numpy as np
import pandas as pd
import re
import json
import time
c_key = "gYNH4hRCzGvAtgVig2VjJgpnY"
c_key_secret = "5IPHN6S713jbtTtmHcFnpWGDOXnoLGqi3GMVVkLzSoECyHHw6B"
access_token = "1109532827891032065-zOsINvtJBaZ6HV0XiXoPp1C68fIIn2"
access_token_secret = "VlApfqJcwbfTmUWe4d3NA3w1epQOUUkGek3Oze11nOysy"
month_endpoint = "https://api.twitter.com/1.1/tweets/search/30day/monthDe
v.json"
full_endpoint = "https://api.twitter.com/1.1/tweets/search/fullarchive/fu
llDev.json"
api = TwitterAPI(c_key,
                 c_key_secret,
                 access_token,
                 access_token_secret)
```

In [2]:
```
# time format 201712220000


DATES_DICT = {'FEB19' : '201903010000',
              'JAN19' : '201902010000',
              'DEC18' : '201811010000',
              'NOV18' : '201812010000',
              'OCT18' : '201811010000',
              'SEP18' : '201810010000',
              'AUG18' : '201809010000',
              'JUL18' : '201808010000',
              'JUN18' : '201807010000',
              'MAY18' : '201806010000',
              'APR18' : '201805010000',
              'MAR18' : '201804010000',
              'FEB18' : '201803010000',
              'JAN18' : '201802010000',
              'DEC17' : '201701010000',
              'NOV17' : '201712010000',
              'OCT17' : '201711010000',
              'SEP17' : '201710010000',
              'AUG17' : '201709010000',
              'JUL17' : '201708010000',
              'JUN17' : '201707010000',
              'MAY17' : '201706010000',
              'APR17' : '201705010000',
              'MAR17' : '201704010000',
              'FEB17' : '201703010000',
              'JAN17' : '201702010000',
              'DEC16' : '201601010000',
              'NOV16' : '201612010000',
              'OCT16' : '201611010000',
              'SEP16' : '201610010000',
              'AUG16' : '201609010000',
              'JUL16' : '201608010000',
              'JUN16' : '201607010000',
```

```
                'MAY16' : '201606010000',
                'APR16' : '201605010000',
                'MAR16' : '201604010000',
                'FEB16' : '201603010000',
                'JAN16' : '201602010000',
                'DEC15' : '201501010000',
                'NOV15' : '201512010000',
                'OCT15' : '201511010000',
                'SEP15' : '201510010000',
                'AUG15' : '201509010000',
                'JUL15' : '201508010000',
                'JUN15' : '201507010000',
                'MAY15' : '201506010000',
                'APR15' : '201505010000',}
MONTH = ['0',
          'JAN',
          'FEB',
          'MAR',
          'APR',
          'MAY',
          'JUN',
          'JUL',
          'AUG',
          'SEP',
          'OCT',
          'NOV',
          'DEC']
```

In [4]:

```
#full archive searches
PRODUCT = 'fullarchive'
LABEL = 'fullDev'
QUERY = '(computer OR Computer) (science OR Science)  lang:en'
TAG = '-is:retweet place_country:us '

request_dict = {'query' : QUERY,
                'tag' : TAG,
                'toDate' : TO_DATE}
```

```
---------------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-4-9be2178a9a65> in <module>
      7 request_dict = {'query' : QUERY,
      8                 'tag' : TAG,
----> 9                 'toDate' : TO_DATE}

NameError: name 'TO_DATE' is not defined
```

In [5]:

```
PRODUCT = '30day'
LABEL = 'monthDev'
SEARCH_TERM = 'computer science OR Computer Science -filter:retweets'

#used this to grab tweets for full  tweets database a year at a time by j
ust changing the year
#it grabbed 100 tweets from each month
#for as many months as I was able with 50 requests
i = 1
```

```
TWEETS = []
YEAR = '15'
while i < 5000:
    month = MONTH[i]+YEAR
    TO_DATE = DATES_DICT[month]
    request_dict = {'query' : QUERY,
                    'tag' : TAG,
                    'toDate' : TO_DATE}

    r = api.request('tweets/search/%s/:%s' % (PRODUCT, LABEL),
                    request_dict)
    for item in r:
        TWEETS.append(item)
    with open('tweets_'+month+'.txt', 'w') as file:
        file.write(json.dumps(TWEETS, indent=4))
    print(month)
    i = i + 1
    time.sleep(60)
```

```
---------------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
<ipython-input-5-11fa8f87671e> in <module>
     11 while i < 5000:
     12     month = MONTH[i]+YEAR
---> 13     TO_DATE = DATES_DICT[month]
     14     request_dict = {'query' : QUERY,
     15                     'tag' : TAG,

KeyError: 'JAN15'
```

In [ ]:
```
PFT_LONG = -91.179858
PFT_LAT = 30.407750

ULL_LONG = -92.018258
ULL_LAT = 30.224502

PRODUCT = '30day'
LABEL = 'monthDev'
QUERY = '(computer OR Computer) (science OR Science)  lang:en'
TAG = '-is:retweet point_radius:[-92.018258,30.224502]'
```

In [ ]:
```
 def iterate_request(request):
        it = request.get_iterator()
        for item in it:
            yield item
```

In [ ]:
```
PRODUCT = '30day'
LABEL = 'monthDev'
QUERY = '(computer OR Computer) (science OR Science)  lang:en'
TAG = '-is:retweet point_radius:[-92.018258,30.224502,1]'

#this is to use to get the most recent tweets from the past 30 days withi
n a mile of the center of LSU
i = 1
```

```
tweets = []
while(True):
    wait = 5
    start = time.time()
    request_dict = {'query' : QUERY,
                    'tag' : TAG}


    #make request
    r = api.request('tweets/search/%s/:%s' % (PRODUCT, LABEL),
                    request_dict)
    print (i)
    i = i + 1

    for item in iterate_request(r):
        tweets.append(item)

    #get token
    cursor = -1
    r_json = r.json()
    if 'next' in r_json:
        cursor = r_json['next']
        elapsed = time.time() - start
        pause = wait - elapsed if elapsed < wait else 0
        time.sleep(pause)

    #check for next page of tweets if token is valid, if token is -1 quit
 running
    while cursor != -1 and i < 100:
        start = time.time()
        #add token to request dict
        request_dict['next'] = cursor

        #make request
        r = api.request('tweets/search/%s/:%s' % (PRODUCT, LABEL),
                    request_dict)
        print(i)
        i = i + 1

        #add items to tweet list
        for item in iterate_request(r):
            tweets.append(item)

        #get token
        cursor = -1
        r_json = r.json()
        if 'next' in r_json:
            cursor = r_json['next']

        #pause to ensure new request works
        elapsed = time.time() - start
        pause = wait - elapsed if elapsed < wait else 0
        time.sleep(pause)



    break
```

```
#write results to file
with open('30day_ullcsc_tweets.txt', 'w') as file:
        file.write(json.dumps(tweets, indent=4))
print('done')
```

In [ ]:
```
#used this to grab tweets for basic search, which is done by # of tweets
 not length of time, so
#Twitter estimates it at 6-9 days worth of tweets included in basic searc
h
SEARCH_TERM = '\"Computer Science" OR "computer science\" -filter:retweet
s'
print(SEARCH_TERM)
n = 80000
tweets = []

pager = TwitterPager(api, 'search/tweets', {'q': SEARCH_TERM})

try:
    for item in pager.get_iterator():
        tweets.append(item)
        if(len(tweets) % 100 == 0):
            print(len(tweets))
        if(len(tweets) > n):
            break
except:
    print('exception happened')
finally:
    with open('standard_csc_tweets', 'w') as file:
        file.write(json.dumps(tweets, indent=4))
        print('Done')
```

In [5]:
```
#Basic search for the hastag hellboy in english
# no retweets only posts and replies.
MIDDLE_BR_LAT= 30.440975
MIDDLE_BR_LONG = -91.105336

SEARCH_TERM = '(#shazam OR #shazammovie) lang:en -filter:retweets'
print(SEARCH_TERM)
n = 5400
tweets = []

pager = TwitterPager(api, 'search/tweets', {'q': SEARCH_TERM})

try:
    for item in pager.get_iterator():
        tweets.append(item)
        if(len(tweets) % 100 == 0):
            print(len(tweets))
        if(len(tweets) > n):
            break
except:
    print('exception happened')
finally:
    with open('standard_shazam_tweets', 'w') as file:
        file.write(json.dumps(tweets, indent=4))
        print('Done')
```