Web Science: Assignment #2

Alexander Nwala

Apurva Modi

Saturday, February 16, 2019

Contents

Problem 1

Write a Python program that extracts 1000 unique links from Twitter. Omit links from the Twitter domain (twitter.com)
Also note that you need to verify that the final target URI (i.e.,the one that responds with a 200) is unique. You could have many different shortened URIs for www.cnn.com (t.co, bit.ly, goo.gl,etc.).

SOLUTION :

1. The program requires one to create the Twitter developer keys and access Tokens. Find the below process to obtain the keys and access tokens

   1. Creating an user account at "https://www.twitter.com"
   2. Log in to "https://www.apps.twitter.com/" and create an application
   (note: The developer needs to fill the form found after clicking the "Create App" option)
   3. Finally, selecting the "Generate secret access tokens" option will create the tokens, which could be used to call twitter APIs

Extracting 1000 unique links from the twitter feed across all the users worldwide :
The below code in Listing 1; extracts the web links, captures the redirection and finally saves in an independent file.

Listing 1: assignment2_1.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Feb 16 00:41:30 2019
@Credits: https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/
@Credits: http://adilmoujahid.com/posts/2014/07/twitter-analytics/
@author: apurvamodi
"""
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import json
import requests

ckey = 'mtDSeNYtJUZkKfspxTFmk7Nn8'
csecret = 'iYg5kksoIQKGsXVwGZ7bYpH0cFlxonNPg9hyhDKdGoP89bic6G'
atoken = '1094978973245812738-msYR1atvDnyfTTO46shWdnp5SIJcAA'
asecret = 'EMdqw7fA4IfkDYzKBqNQfoe5sAwz7dgCcRTWkpZOteUKd'
count = 0
uniqueLink = set([])
txtFile = open("finally1000URIs.txt","w")
class listener(StreamListener):

    def on_data(self, data):
        global count;
        if(count == 2000):
```

```
            return False
        else:
            jsonTweets = json.loads(data)
            links = jsonTweets['entities']['urls']

        if( len(links) != 0 and jsonTweets['truncated'] == False ):
            links = self.getLinksFromTweet(links)

            for link in links:
                global uniqueLink
                if(link in uniqueLink):
                    pass
                else:
                    print(link)
                    count = count + 1
                    uniqueLink.add(link)
                    txtFile.write(link)
                    txtFile.write('\n')
            #print(count)
        return True

    def getLinksFromTweet(self, linksDict):

        links = []
        destUrl = ''
        for uri in linksDict:

            if("https://twitter.com" in uri['expanded_url']):
                pass
            else:
                destUrl = self.checkForRedirection(uri['expanded_url'][0:])
                links.append(destUrl)
            return links

    def checkForRedirection(self,link1):
        response = requests.get(link1, allow_redirects=False, timeout=5)
        return response.url



    def on_error(self, status):
        if status == 420:
            #returning False in on_data disconnects the stream
            return False
        return True


auth = OAuthHandler(ckey, csecret)
auth.set_access_token(atoken, asecret)
try:
  twitterStream = Stream(auth, listener())
  twitterStream.filter(track=['crypto'])
except:
```

```
  twitterStream.filter(track=['crypto'])
txtFile.close()
```

Extracted URIs :

Listing 2: Extracted Links

```
http://bit.ly/2ApQAH3#1
http://bit.ly/2ApQAH3#13
http://bit.ly/2ApQAH3#8
http://bit.ly/2ApQAH3#9
http://bit.ly/2ASEDvo
http://bit.ly/2BBvwy0
http://bit.ly/2BEtzkt
http://bit.ly/2BFeq2b
http://bit.ly/2BGgSWf
http://bit.ly/2BIczcZ
http://bit.ly/2BsMM77
http://bit.ly/2CNZDoc
http://bit.ly/2DIbxxU
http://bit.ly/2DJ4EMT
http://bit.ly/2DLDwNf
http://bit.ly/2DmlWPC
http://bit.ly/2Ed9of4
http://bit.ly/2EdnDjU
http://bit.ly/2Efx9mS
http://bit.ly/2F90npP
http://bit.ly/2GcxybS
http://bit.ly/2GGVfbA
http://bit.ly/2GK3OCG
http://bit.ly/2GK8AzX
http://bit.ly/2GL9ITT
http://bit.ly/2Gm7Txj
http://bit.ly/2GNiVeD
http://bit.ly/2GtxZyI
http://bit.ly/2Guc7Dm
http://bit.ly/2Gux0OO
http://bit.ly/2GuzoFa
http://bit.ly/2Gz6ZNa
http://bit.ly/2IfbAa4
http://bit.ly/2IhAqWP
http://bit.ly/2Ihg7Zj
http://bit.ly/2Irjq0q
http://bit.ly/2KQ3lSN
http://bit.ly/2N1GPE4
http://bit.ly/2NafgbR
http://bit.ly/2NejIqc
http://bit.ly/2Qn76wD
http://bit.ly/2QSPbOx
http://bit.ly/2RUhWuN
http://bit.ly/2RW9B9T
http://bit.ly/2RXWjtx
```

For more links visit the following file **"finally1000URIs.txt"**

Problem 2

Download the TimeMaps for each of the target URIs. We'll use the ODU Memento Aggregator,.
For example:
URI-R = http://www.cs.odu.edu/
URI-T = http://memgator.cs.odu.edu/timemap/link/http://www.cs.odu.edu/
OR
URI-T = http://memgator.cs.odu.edu/timemap/json/http://www.cs.odu.edu/

Create a histogram* of URIs vs. number of Mementos (as computed from the TimeMaps). For example, 100 URIs with 0 Mementos, 300 URIs with 1 Memento, 400 URIs with 2 Mementos, etc. The x-axis will have the number of mementos, and the y-axis will have the frequency of occurance.

SOLUTION

The solution for this problem is outlined by the following steps:

1. **I passed the extracted URIs stored in the file "finally1000URIs.txt" using the below mentioned program to get the Mementos**

   `http://memgator.cs.odu.edu/timemap/link/http://www.cs.odu.edu`

2. **Extracting the timemap for each URI :** The below code in Listing 1; extracts the time maps; Saves URI Index,URI Count and Mementos in an independent text file.

Listing 3: assignment2_2.py

```
import requests
import sys
import time

uri_t = "http://memgator.cs.odu.edu/timemap/json/"
mementoList = []
plotMementosDict = {}
count = 1
headers = {'user-agent': 'my-app/0.0.1'}
f = open('finally1000URIs.txt','r')
fw = open('MemeFile.txt','w')
fw.write("Count,URI,Mementos")
fw.write('\n')
for line in f:
     if(line == ''):
          pass
     else:
          response = requests.get(uri_t + line.strip(),headers=headers)
          print("...",response.status_code)
          if(response.status_code == 200):
               memento = response.headers['X-Memento-Count']
               mementoList.append(memento)
          else:
               mementoList.append(0)

```

```
for value in mementoList:
     if(str(value) in plotMementosDict):
          uriValue = plotMementosDict.get(str(value))
          plotMementosDict[str(value)] = uriValue + 1
     else:
          uriValue = 0
          plotMementosDict[str(value)] = uriValue + 1

print("plotMementosDict : ",plotMementosDict)

for mementoValue in plotMementosDict:
     print('{:>8}  {:>8}'.format(str(plotMementosDict[mementoValue]),mementoValue))
     fw.write(str(count)+","+str(plotMementosDict[mementoValue])+","+
     str(mementoValue))
     fw.write("\n")
     count = count + 1
```

TimeMaps obtained are saved in the format "Count,URI,Mementos" in to an independent text file as attached **"MemeFile.txt":**

Listing 4: Extracted with Count URIs and Mementos

```
Count,URI,Mementos
1,814,0
2,57,1
3,2,21
4,1,15
5,2,60
6,1,35
7,16,4
8,5,5
9,1,157
10,1,38
11,1,105
12,1,7
13,4,14
14,3,28
15,1,503
16,7,6
17,17,3
18,1,109
19,2,18
20,1,9
21,2,20
22,1,2737
23,1,85
24,37,2
25,3,37
26,4,16
27,1,46
28,1,270
29,1,3426
```

```
30,1,39
31,1,44
32,3,10
33,1,48
34,1,1681
35,1,40
36,1,45
37,1,111
38,1,30
39,1,378
40,1,107
41,1,25
42,1,42
43,1,3442
44,1,13
45,1,8
46,1,23
47,1,1308
48,2,33
49,1,1263
50,1,12
51,1,72
52,1,53
53,1,29
54,1,34
```

3. The TimeMaps obtained are saved in CSV file and plotted as Bar Chart . The below python code in Listing 5 creates a bargraph

Listing 5: BarChart-ScatterPlot.ipynb

```
import matplotlib.pyplot as plt
import csv
x = []
y = []
with open('MemeFile-txt.csv','r') as csvfile:
    next(csvfile, None)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        y.append((row[1]))
        x.append((row[2]))
y.sort()
x.sort()
plt.bar(x,y)
plt.ylabel("Number of URIs")
plt.xlabel("Mementos")
plt.title("Bar Graph Number of URIs vs Mementos")
```

Problem 3

Estimate the age of each of the 1000 URIs using the "Carbon Date" tool:
For URIs that have > 0 Mementos and an estimated creation date, create a graph with age (in days) on the x-axis and number of mementos on the y-axis.

Not all URIs will have Mementos, and not all URIs will have an estimated creation date.
Show how many fall into either categories. For example,

```
Total URIs: 1017
Number of Mementos: 812
Number of Date Estimate: 476
```

SOLUTION
Carbon Date for a site is calculated using the below URI pattern:

`http://cd.cs.odu.edu/cd?url=http://www.cs.odu.edu/`

Where the above URL calculates the carbon date for "http://www.cs.odu.edu/"

Listing 6: assignment2_3.py

```
import requests
import csv
import json
import sys
from datetime import datetime

noAge = 0
noMementos = 0
plotMementosDict = {}
totalURI = 0
f = open('finally1000URIs.txt','r')
for link in f:
     if(link == ''):
          pass
     else:
          try:
               totalURI = totalURI + 1
               carbonDateResponse = requests.get("http://localhost:8888/cd/"+link)
               mementoResponse = requests.get("http://memgator.cs.odu.edu/timemap/json/"
               +link,stream=True,headers={'User-Agent': 'Mozilla/5.0'})
               print('Carbon Date status :',carbonDateResponse.status_code)
               print('Mementos status :',mementoResponse.status_code)
               carbonDateResponseJSON = carbonDateResponse.json()
               totalMementos = mementoResponse.headers["X-Memento-Count"]
               ageDate = carbonDateResponseJSON["estimated-creation-date"]

               if(ageDate == ""):
                    noAge = noAge + 1
               if(totalMementos == '0'):
                    noMementos = noMementos + 1

               print('No mementos: ', noMementos)
               print('No Carbon Date: ', noAge)
```

```
                if carbonDateResponse.status_code==200 and
                   mementoResponse.status_code==200:
                     now = datetime.now()
                     createdDate = datetime.strptime(ageDate, '%Y-%m-%dT%H:%M:%S')
                     currentAge = (now - createdDate)
                     print('age: ', currentAge.days)
                     print('Memento: ',totalMementos)
                     plotMementosDict[str(currentAge.days)] = totalMementos

          except KeyboardInterrupt:
               exit()
          except:
               print("An exception")
               pass

print('Total URIs',totalURI)
print('No Mementos',noMementos)
print('no date estimate',noAge)

with open('carbonDate.csv', 'w') as in_csvFile:
     fieldsnames = ['currentAge','Mementos']
     writer = csv.DictWriter(in_csvFile, fieldnames=fieldsnames)
     writer.writeheader()
     # writer = csv.writer(csv_file)
     for age, MementoValue in plotMementosDict.items():
          writer.writerow({'currentAge': age, 'Mementos': MementoValue})
          #writer.writerow([age, MementoValue])
```

Listing 7: BarChart-ScatterPlot.ipynb

```
import matplotlib.pyplot as plt
import numpy as np
import csv

x = []
y = []

with open('carbonDate.csv','r') as csvfile:
    next(csvfile, None)
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        x.append((row[0]))
        y.append(int(row[1]))


plt.yscale('log')
plt.scatter(x, y)
plt.xlabel("Age(s) in Days")
plt.ylabel("Number of Mementos")
plt.title("Scatter Plot Number of Mementos vs their Age in Days")
```