Web Science: Assignment #2

Alexander Nwala

Puneeth Bikkasandra

Thursday, February 15, 2018

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: twitterLinks.py

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from BeautifulSoup import BeautifulSoup
from requests import Request, Session
import json
import time
import requests

ckey = 'L9QTLPWp2CswcJWRRaNtrsxWO'
csecret = 'nKm7PFmtFAWQYupqffdLz6YWD23VzFlNV8myAei7BaFYDNIoZN'
atoken = '962415725104324608-gJ39MDzaSIlbj44ZBSIuhezb3QcgOAx'
asecret = 'SD9uFFWZH5zfrg4taMdjXkH3vefgmqpPne10EmPiXLijg'
count = 0
uniqueLinks = set([])
linksFile = open("1000TwitterLinks.txt","w")

class listener(StreamListener):

    def on_data(self, data):
        global count
        if(count == 1300):
            return False
        else:
            tweetJson = json.loads(data)
```

```
                username = tweetJson['user']['screen_name']
                links = tweetJson['entities']['urls']

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

                     for link in links:
                          global uniqueLinks,linksFile
                          if(link in uniqueLinks):
                               pass
                          else:
                               print(link)
                               count = count + 1
                               uniqueLinks.add(link)
                               linksFile.write(link)
                               linksFile.write('\n')
                     # time.sleep(1)
           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, verify=False, timeout=10)
           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=['football'])
except:
     twitterStream.filter(track=['football'])
```

Sample listing of links obtained after extraction:

Listing 2: Extracted Links

```
https://www.nytimes.com/2018/02/11/opinion/head-trauma-football.html?partner=IFTTT
https://www.football-italia.net/117066/ht-benevento-shock-roma
https://mobile.twitter.com/ProSoccerSF/status/962740869999706112
https://paper.li/TibsNews/1359398292?edition_id=7f545270-0f6b-11e8-94d7-0cc47a0d164b
https://www.instagram.com/p/BfEfI8Ollhj/
https://www.kingfut.com/2018/02/11/kotoko-cara-three-penalty-misses/
https://www.change.org/p/save-youth-football-in-california
https://sportgid.net/football/levante-real-22-obzor-matcha-i-video-golov-03-02-2018/
http://football-station.net/b/2018/02/104384.html
http://www.football-chairman.com/
```

Remaining links can be found in the attached file **"1000twitterLinks.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. **Passing the URL collected in a file "1000twitterLinks.txt" from previous code link in the following way:**

   `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 file.

Listing 3: timeMaps.py

```
import requests
import sys
import time

uri_t = "http://memgator.cs.odu.edu/timemap/link/"
mementoList = []
plotMementosDict = {}
count = 1
headers = {'user-agent': 'my-app/0.0.1'}
f = open('1000TwitterLinks.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)
```

```
25
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
35
print("plotMementosDict : ",plotMementosDict)

for mementoValue in plotMementosDict:
     print('{:>8}  {:>8}'.format(str(plotMementosDict[mementoValue]),mementoValue))
40   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 TimeMaps

```
Count,URI,Mementos
1,1,214
2,1,56
3,1,4183
5 4,1,36
5,1,131
6,1,136
7,1,6459
8,1,254
10 9,1,25
10,1,26
11,1,27
12,2,20
13,1,21
15 14,1,23
15,1,45
16,32,1
17,978,0
18,4,3
20 19,15,2
20,4,5
21,6,4
22,1,7
23,4,6
25 24,1,9
25,2,8
26,1,89917
27,1,4093
28,1,3488
```

```
29,1,77
30,1,32107
31,1,95
32,1,4778
33,1,13
34,1,12
35,1,2579
36,1,19
37,1,16
38,1,50
39,1,1472
40,1,49
41,1,295
```

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

Listing 5: histogram.py

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

data=pd.read_csv('mem.csv', sep=',', skiprows=0,header=None, index_col =0).dropna()
data.plot(kind='bar')
plt.ylim(0, 1000)
plt.ylabel("URIs")
plt.xlabel('mementos')
plt.title('Histogram')
L=plt.legend()
L.get_texts()[0].set_text('URIs')
plt.show()
```

The below graph shows the extracted content with **URI** at y-axis and **Mementos** at x-axis

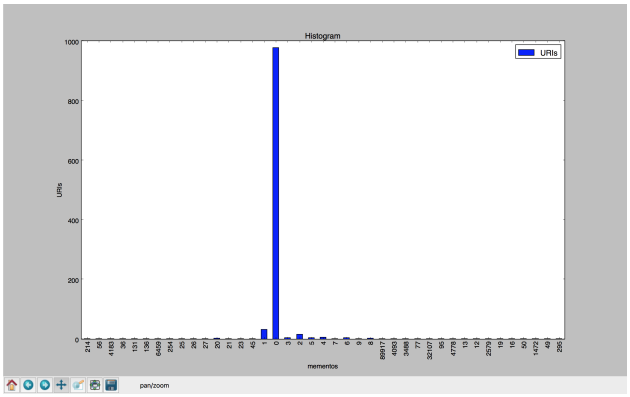

Figure 1: Barplot URI vs Mementos

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:1000
Number of Mementos:137
Number of Date Estimate:212
```

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: AgeMem.py

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

noAge = 0
noMementos = 0
plotMementosDict = {}
totalURI = 0
f = open('1000TwitterLinks.txt','r')
for link in f:
     if(link == ''):
          pass
     else:
          try:
          totalURI = totalURI + 1
          carbonDateResponse = requests.get("http://cd.cs.odu.edu/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', 'wb') as csv_file:
    fieldsnames = ['currentAge','Mementos']
    writer = csv.DictWriter(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])
```