Assignment 2

CS 532: Introduction to Web Science

Spring 2018

Chandrasekhar Reddy Muthyala

Finished on February 13, 2018

1

Question

```
1.  Write a Python program that extracts 1000 unique links from
Twitter.  You might want to take a look at:

http://adilmoujahid.com/posts/2014/07/twitter-analytics/

see also:

http://docs.tweepy.org/en/v3.5.0/index.html
https://github.com/bear/python-twitter
https://dev.twitter.com/rest/public

But there are many other similar resources available on the web.
Note that only Twitter API 1.1 is currently available; version 1
code will no longer work.

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.).  For example:

$ curl -IL --silent https://t.co/DpO767Md1v | egrep -i "(HTTP/1.1|^location:)"
HTTP/1.1 301 Moved Permanently
location: https://goo.gl/40yQo2
HTTP/1.1 301 Moved Permanently
Location: https://soundcloud.com/roanoketimes
/ep-95-talking-hokies-recruiting-one-week-before-signing-day
HTTP/1.1 200 OK
You might want to use the search feature to find URIs, or you can
pull them from the feed of someone famous (e.g., Tim O'Reilly).  If
you find something inappropriate for any reason you see fit, just
discard it and get some more links.  We just want 1000 links that
were shared via Twitter.
Hold on to this collection and upload it to github -- we'll use it
later throughout the semester.
```

Solution

Extracting 1000 unique links from twitter excluding twitter domain links. Initially I do not know how to approach this problem, so I referred links given in the assignments and got to know that there is a twitter API which can fetch the latest links based on the key word provided. Steps I followed for this problem: 1) Created a twitter account. In order to fetch URLs from the twitter it requires access token and key, so I have created an Application developer account. 2) Used python as a programming language and these are the libraries used.

- from tweepy.streaming import StreamListener
- from tweepy import OAuthHandler
- from tweepy import Stream
- import json
- import requests
- from urllib.error import URLError
- from http.client import IncompleteRead

The main program for twitter streaming was **TwitterTweepyApi.py** . It first starts by listening to the stream of data from twitters streaming API. Then once a piece of data is received as a JSON format, parsed the JSON and extracted URLs from entities object. For each and every link I am checking whether that link has URLs and used truncated attribute to avoid the repetition of twitter status(for example : share status). After that I am sending that link to other function called **getLinksFromTweet** which will provide me an uncorrupted link as well as not twitter domain URL and then checking if it is a unique link or not, it goes on running this program until I get 1000 Unique URLs. All those links I am storing in a file called **finalurls.txt**.

Run on the command line

```
1 python twitterTweepyApi.py> finalurls.txt
```

```
'''
  prerequisites:
  0. create a twitter account
  1. obtain your access tokens: https://apps.twitter.com/
    1.0 create new app
  2. install tweepy (pip install tweepy)

  credit:
  http://docs.tweepy.org/en/v3.4.0/streaming_how_to.html
  http://adilmoujahid.com/posts/2014/07/twitter-analytics/
  https://pythonprogramming.net/twitter-api-streaming-tweets-
  python-tutorial/

  Tweet JSON:, use http://jsonviewer.stack.hu/ to view object
  https://developer.twitter.com/en/docs/tweets/data-dictionary/
  overview/intro-to-tweet-json

  rate limiting:
  https://developer.twitter.com/en/docs/basics/rate-limiting

  streaming rate limiting:
  https://developer.twitter.com/en/docs/tweets/filter-realtime/
  guides/connecting.html
'''

#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import json
import time
import requests
from urllib.error import  URLError
from http.client import IncompleteRead
# import http.client

#get keys from: https://apps.twitter.com/
#consumer key, consumer secret, access token, access secret.
ckey="v5YfpnUYDpYFrm9hn4wfV6hCw"
csecret="2nOHkJyPqteSudHrlS8YWcsuK8wvX7AgIPVIK8ySHj3dU5TnF6"
atoken="956029983469776897-GUROfj1hmnuImgewZAZrKAog5dJ1q3O"
asecret="UipAVctbQJUQEczQ2s7Dd5J3teLa6ZwEAWWbaceTSSfca"
MyUniqueLinks = []
class listener(StreamListener):
  global MyUniqueLinks
  def on_data(self, data):
    #learn about tweet json structure: https://developer.twitter
    .com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-
```

```
    json
45      tweetJson = json.loads(data)
46      links = tweetJson['entities']['urls']
47      if( len(links) != 0 and tweetJson['truncated'] == False ):
48        links = self.getLinksFromTweet(links)
49        for l in links:
50          if l not in MyUniqueLinks:
51            if len(MyUniqueLinks)<1001:
52              MyUniqueLinks.append(l)
53              print(l)
54      return True
55
56    def getLinksFromTweet(self, linksDict):
57      links = []
58      for uri in linksDict:
59        try:
60          responsed = requests.get(uri['expanded_url'], stream=
            True,timeout=5,allow_redirects=True,headers={'User-Agent
            ':'Mozilla/5.0'})
61          intialStatus = responsed.status_code
62          if intialStatus==200 and responsed.headers['content-type
            '] != None:
63            if(responsed.history):
64              if "twitter.com" not in responsed.url:
65                links.append(responsed.url )
66            else:
67              if "twitter.com" not in uri['expanded_url']:
68                links.append( uri['expanded_url'] )
69          if(responsed.history):
70            if "twitter.com" not in responsed.url:
71              links.append(responsed.url )
72        except KeyboardInterrupt:
73          exit()
74        except:
75          pass
76
77      return links
78
79    def on_error(self, status):
80      print( status )
81
82      if status_code == 420:
83        #returning False in on_data disconnects the stream
84        return False
85      return True
86
87
88  auth = OAuthHandler(ckey, csecret)
89  auth.set_access_token(atoken, asecret)
```

```
90
91 twitterStream = Stream(auth, listener())
92 twitterStream.filter(track=['football','cricket'])
```

Listing 1: Python script for twitter streaming

2

Question

2. Download the TimeMaps for each of the target URIs. We'll use the ODU Memento Aggregator, so 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/
(depending on which format you'd prefer to parse)
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 occurence.
* = https://en.wikipedia.org/wiki/Histogram
What's a TimeMap?
See: http://www.mementoweb.org/guide/quick-intro/
And the week 4 lecture.

Solution

Libraries used for extracting mementos:

- import requests
- import json
- import os, errno
- import csv

To solve above problem I used `http://memgator.cs.odu.edu/` and wrote **extractMemento.py** to retrieve a JSON response for time map of my 1000 URLs. First I collected mementos from the response header and I am storing it to a dictionary data structure, here key as mementos count and value as number of URLs with those mementos. Here I found interesting for the links which do not have mementos. Status code will gives me 404 and with mementos greater than zero results 200 status. Based on that time maps are storing in **urlMementos-1** folder. For each URL I am creating a text file. If the URL have zero mementos it wont giive JSON, so I am writing URL and 404 not found. If the mementos greater than zero I am storing time map JSON to text file. After looping through all the 1000 links I have a complete list of mementos in dictionary and those values are writing into **testMementos.csv**.

Run on the command line

```
python extractMemento.py
```

```
import requests
import csv
import json
import os, errno
mementos = {}
lineNo=1
class extractMementos:
  def collectmementos(self):
    global mementos
    global lineNo
    with open('finalurls.txt') as fp:
      for line in fp:
        try:
```

```
14            url = 'http://memgator.cs.odu.edu/timemap/json/'+
              line.strip()
15            responsed = requests.get(url, stream=True,headers
              ={'User-Agent':'Mozilla/5.0'})
16            mementoCount = responsed.headers['X-Memento-Count
              ']
17            print("status:",responsed.status_code,"m:",
              mementoCount)
18            if not os.path.exists('urlMementos-1'):
19              try:
20                os.makedirs('urlMementos-1')
21              except OSError as e:
22                if e.errno != errno.EEXIST:
23                  raise
24
25            if responsed.status_code==200:
26              with open('urlMementos-1/timemap%s.txt' %
                lineNo, 'w+') as outfile:
27                json.dump(responsed.json(), outfile,indent=4,
                  sort_keys=True,ensure_ascii=False)
28                lineNo = lineNo+1
29            else:
30              print('else-404')
31              with open('urlMementos-1/timemap%s.txt' %
                lineNo, 'w+') as outfile:
32                print('else-file',lineNo)
33                #print(responsed.headers)
34                outfile.write(line.strip()+": 404 not found
                  .")
35                lineNo = lineNo+1
36                print('l-no',lineNo)
37
38            if mementoCount in mementos:
39              mementos[mementoCount] = mementos[mementoCount
                ]+1
40            else:
41              mementos[mementoCount] = 1
42            print("memento:",mementos)
43          except KeyboardInterrupt:
44            exit()
45          except:
46
47            pass
48  obj=extractMementos()
49  obj.collectmementos()
50  with open('testMementos.csv', 'w') as csvfile:
51    fieldnames = ['mementos', 'count']
52    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
53    writer.writeheader()
```

```
54    for key,value in mementos.items():
55      writer.writerow({'mementos': key, 'count': value})
56      print (key, value)
```

Listing 2: Python script for extracting mementos

Now I have complete list of mementos in **testMementos.csv** and I drew a histogram chart using python language. Libraries used for ploting histogram:

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

Here matplotlib library is used for ploting the histogram gram, pandas is used to get the data from csv in to dataframes.

Run on the command line

```
1  python histogram.py
```

```
1  import matplotlib.pyplot as plt
2  import pandas as pd
3  import numpy as np
4  data = pd.read_csv('mementosCollection.csv', sep=',',
   skiprows=0,header=None, index_col =0).dropna()
5  # data = pd.read_csv('testMementos.csv', sep=',', index_col
    =0)
6
7  data.plot(kind='bar')
8  plt.ylabel("URL's count")
9  plt.xlabel('mementos')
10 plt.title('Histogram')
11 L=plt.legend()
12 L.get_texts()[0].set_text('URLs count')
13 plt.show()
```

Listing 3: Python script for histogram

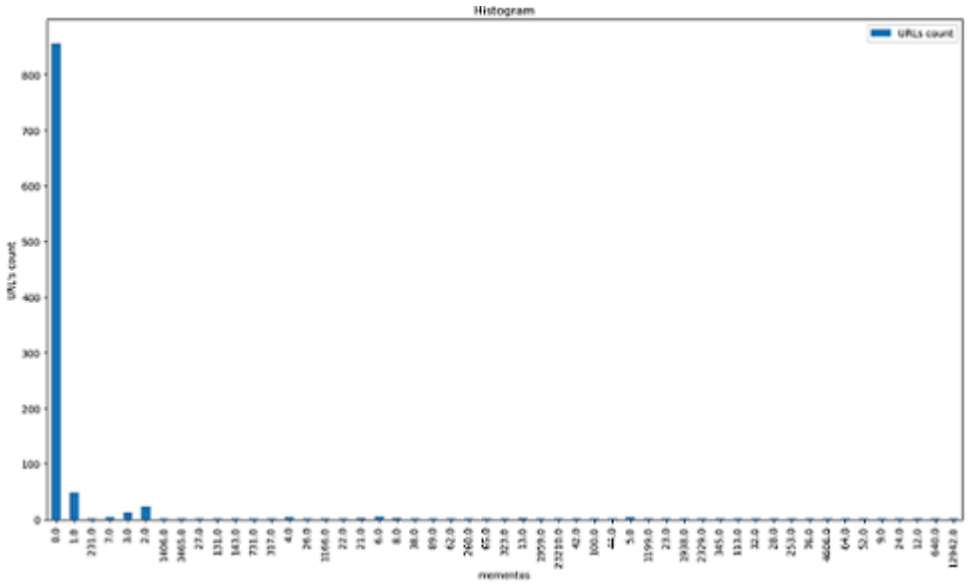

Figure 1: Histogram of Number of URIs vs. Number of Mementos

3

Question

```
3.  Estimate the age of each of the 1000 URIs using the "Carbon
Date" tool:
http://ws-dl.blogspot.com/2016/09/2016-09-20-carbon-dating-web-version-30.html
Note: you should use "docker" and install it locally.  You can do
it like this:
http://cd.cs.odu.edu/cd?url=http://www.cs.odu.edu/
But it will inevitably crash when everyone tries to use it at the
last minute.
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
no mementos:     137
no date estimate:  212
```

Solution

To solve the above problem, used the carbon dating tool [**?**] provided in the question to retrieve the estimated creation dates and memgator to fetch the number of memtos for each of the URIs from the existing **finalurls.txt** file.In this I am getting mementos and age of the URLs. Libraries used to find age and mementos are:

- import requests
- import json
- from datetime import datetime
- import csv
- import errno

I am passing each URL from **finalurls.txt** to **carbonDateAgeCalculation** function, in which carbon date and mentos are finding and writing all URLs of age and mementos whose status$_{c}ode$200in**ageAndMementos.csv**.$Theprogramshowninlisti$

Run on the command line

```
python carbonDate.py
```

```
1
2  import requests
3  import json
4  import csv
5  from datetime import datetime
6  import errno
7  totalUrls=0
8  noMementoCount = 0
9  noEst = 0
10 def carbonDateAgeCalculation(url):
11     print('----------------------------')
12     global ageAndMementos
13     global noMementoCount
14     global noEst
15     global totalUrls
16     totalUrls = totalUrls+1
17     try:
18         cdUrl = 'http://cd.cs.odu.edu/cd/'+url
19         memUrl = 'http://memgator.cs.odu.edu/timemap/json/'+url
20         cdres = requests.get(cdUrl)
```

```
21          memres =requests.get(memUrl, stream=True, headers={'User
                -Agent': 'Mozilla/5.0'})
22          print('cdredst:',cdres.status_code,'memst:',memres.
                status_code)
23          data = cdres.json()
24          memcount = memres.headers['X-Memento-Count']
25          estDate = data['estimated-creation-date']
26
27          if estDate=="":
28              noEst = noEst+1
29          if memcount=='0':
30              noMementoCount = noMementoCount+1
31          print('noMementoCount: ', noMementoCount)
32          print('noEst: ', noEst)
33          if cdres.status_code==200 and memres.status_code==200:
34              estDate = datetime.strptime(estDate, '%Y-%m-%dT%X')
35              age = datetime.now() - estDate
36              print('age: ', age.days, 'Memento: ',memcount)
37
38              print('------------------------')
39              return([age.days,memcount])
40      except KeyboardInterrupt:
41          exit()
42      except:
43          pass
44      return('fail')
45
46  with open('finalurls.txt') as fp:
47      with open('ageAndMementos.csv','w') as csvfile:
48          fieldsnames = ['age','mementos']
49          writer = csv.DictWriter(csvfile, fieldnames=fieldsnames)
50          writer.writeheader()
51          for line in fp:
52              values = carbonDateAgeCalculation(line.strip())
53              if values!='fail':
54                  writer.writerow({'age': values[0], 'mementos':
                        values[1]})
55              else:
56                  print(values)
57
58  print('totalUrls: ', totalUrls)
59  print('no mementos: ', noMementoCount)
60  print('no date estimate: ', noEst)
61
62  '''
63  ('totalUrls: ', 1001)
64  ('no mementos: ', 851)
65  ('no date estimate: ', 139)
66
```

```
67
68 '''
```

Listing 4: Python script for finding the carbon date

Libraries used for plotting scatter plot:

- import pandas as pd
- import matplotlib.pyplot as plt

Run on the command line

```
1 python ageVSmementosScatterPlot.py
```

All the values of URLs of age and mementos are stored in **ageAndMementos.csv**. Used Python language to plot scatter graph.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 df  = pd.read_csv("ageAndMementos.csv")
4 # df.plot()  # plots all columns against index
5 df.plot(kind='scatter',x='age',y='mementos') # scatter plot
6 plt.show()
```

Listing 5: Python script for scatter graph

Also the program in Listing, calculated the following values and saved in **ageOfGoodLinks.txt** file.

- total URIs: 1001
- no mementos: 846
- no date estimate: 139

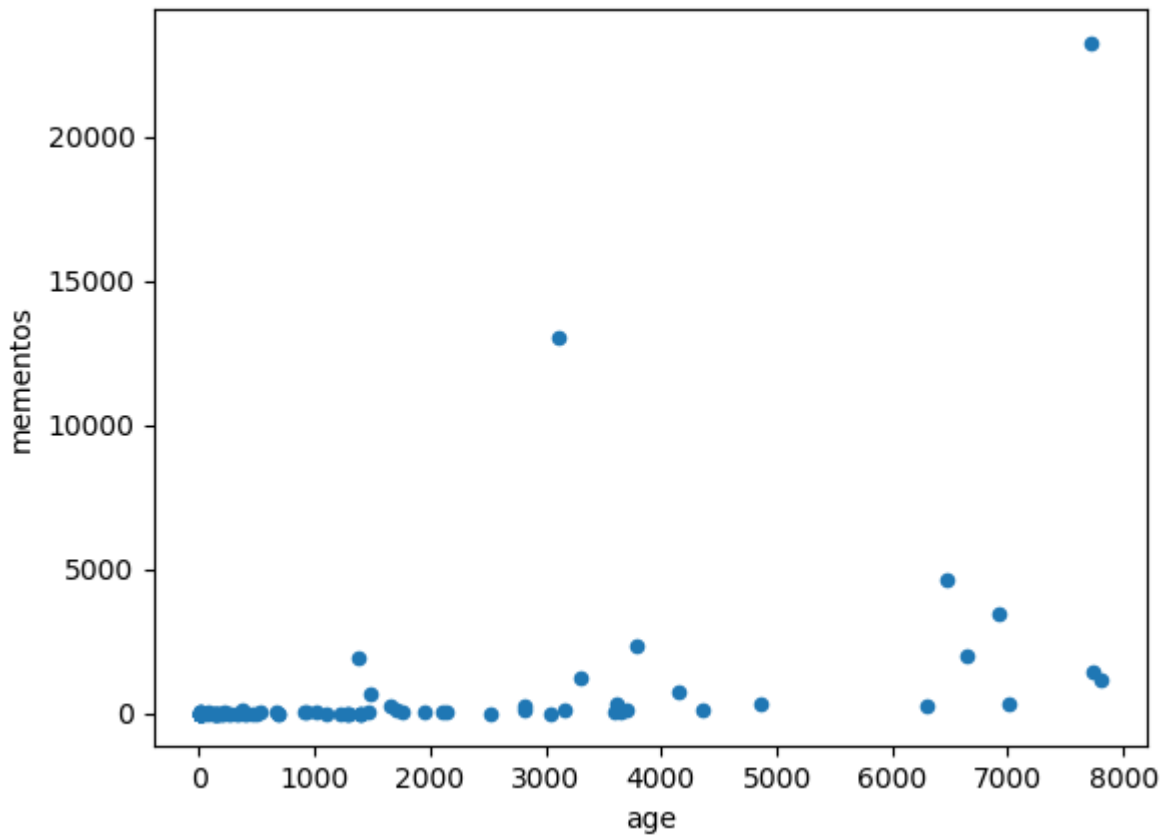

Figure 2: Scatter plot of age in days and number of mementos

References

[1] Python Programmng. “Twitter API Streaming.” N.p., 13 Feb. 2018.https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/

[2] Tweepy Documentation. “Tweepy Documentation - tweepy 3.5.0 .” N.p., 13 Feb. 2018. http://docs.tweepy.org/en/v3.5.0/index.html

[3] Python Software Foundation. “tldextract 2.2.0 : Python Package Index .” N.p., 13 Feb. 2018. https://pypi.python.org/pypi/tldextract

[4] Python Software Foundation. “19.2. json - JSON encoder and decoder - Python 3.6.4 documentation .” N.p., 13 Feb. 2018. https://docs.python.org/3/library/json.html

[5] Python Software Foundation. “14.1. csv - CSV File Reading and Writing - Python 3.6.4 documentation.” N.p., 13 Feb. 2018. https://docs.python.org/3/library/csv.html

[6] Urllib.parse Documentation. “21.8. urllib.parse Parse URLs into components Python 3.6.4 documentation, Web. 13 Feb. 2018. https://docs.python.org/3/library/urllib.parse.html

[7] Urllib.requests Documentation.“Developer Interface Requests 2.18.4 documentation”, Web. 13 Feb, 2018. https://docs.python.org/3.0/library/urllib.request.html