

# 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](http://www.cnn.com) (t.co, bit.ly, goo.gl, etc.). For example:

```
$ curl -IL --silent https://t.co/Dp0767Md1v | 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
```

```

1  '''
2  prerequisites:
3      0. create a twitter account
4      1. obtain your access tokens: https://apps.twitter.com/
5          1.0 create new app
6      2. install tweepy (pip install tweepy)
7
8  credit:
9      http://docs.tweepy.org/en/v3.4.0/streaming-how_to.html
10     http://adilmoujahid.com/posts/2014/07/twitter-analytics/
11     https://pythonprogramming.net/twitter-api-streaming-tweets-
python-tutorial/
12
13     Tweet JSON:, use http://jsonviewer.stack.hu/ to view object
14     https://developer.twitter.com/en/docs/tweets/data-dictionary/
overview/intro-to-tweet-json
15
16     rate limiting:
17     https://developer.twitter.com/en/docs/basics/rate-limiting
18
19     streaming rate limiting:
20     https://developer.twitter.com/en/docs/tweets/filter-realtime/
guides/connecting.html
21  '''
22
23  #Import the necessary methods from tweepy library
24  from tweepy.streaming import StreamListener
25  from tweepy import OAuthHandler
26  from tweepy import Stream
27  import json
28  import time
29  import requests
30  from urllib.error import URLError
31  from http.client import IncompleteRead
32  # import http.client
33
34  #get keys from: https://apps.twitter.com/
35  #consumer key, consumer secret, access token, access secret.
36  ckey="v5YfpnUYDpYFrm9hn4wfV6hCw"
37  csecret="2nOHkJyPqteSudHrLS8YWcsuK8wvX7AgIPVIK8ySHj3dU5TnF6"
38  atoken="956029983469776897-GUOfj1hmnuImgewZAZrKAog5dJ1q3O"
39  asecret="UipAVctbQJUQEczQ2s7Dd5J3teLa6ZwEAWWbaceTSSfca"
40  MyUniqueLinks = []
41  class listener(StreamListener):
42      global MyUniqueLinks
43      def on_data(self, data):
44          #learn about tweet json structure: https://developer.twitter
.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-

```

```

45     json
46     tweetJson = json.loads(data)
47     links = tweetJson['entities']['urls']
48     if( len(links) != 0 and tweetJson['truncated'] == False ):
49         links = self.getLinksFromTweet(links)
50         for l in links:
51             if l not in MyUniqueLinks:
52                 if len(MyUniqueLinks) < 1001:
53                     MyUniqueLinks.append(l)
54                     print(l)
55         return True
56
57 def getLinksFromTweet(self, linksDict):
58     links = []
59     for uri in linksDict:
60         try:
61             responded = requests.get(uri['expanded_url'], stream=
62             True, timeout=5, allow_redirects=True, headers={'User-Agent
63             ': 'Mozilla/5.0'})
64             intialStatus = responded.status_code
65             if intialStatus == 200 and responded.headers['content-type
66             '] != None:
67                 if(responded.history):
68                     if "twitter.com" not in responded.url:
69                         links.append(responded.url )
70                 else:
71                     if "twitter.com" not in uri['expanded_url']:
72                         links.append( uri['expanded_url'] )
73             if(responded.history):
74                 if "twitter.com" not in responded.url:
75                     links.append(responded.url )
76         except KeyboardInterrupt:
77             exit()
78         except:
79             pass
80
81     return links
82
83 def on_error(self, status):
84     print( status )
85
86     if status_code == 420:
87         #returning False in on_data disconnects the stream
88         return False
89     return True
90
91 auth = OAuthHandler(ckey, csecret)
92 auth.set_access_token(accessToken, accessSecret)

```

```
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 occurrence.

\* = <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 command line

```
1 python extractMemento.py
```

```
1 import requests
2 import csv
3 import json
4 import os, errno
5 mementos = {}
6 lineNo=1
7 class extractMementos:
8     def collectmementos(self):
9         global mementos
10        global lineNo
11        with open('finalurls.txt') as fp:
12            for line in fp:
13                try:
```



```

14         url = 'http://memgator.cs.odu.edu/timemap/json/' +
        line.strip()
15         responded = requests.get(url, stream=True, headers
        ={'User-Agent': 'Mozilla/5.0'})
16         mementoCount = responded.headers['X-Memento-Count
        ']
17         print("status:", responded.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 responded.status_code==200:
26             with open('urlMementos-1/timemap%s.txt' %
        lineNo, 'w+') as outfile:
27                 json.dump(responded.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(responded.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 plotting histogram:

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

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

### Run on command line

```

1 | python histogram.py

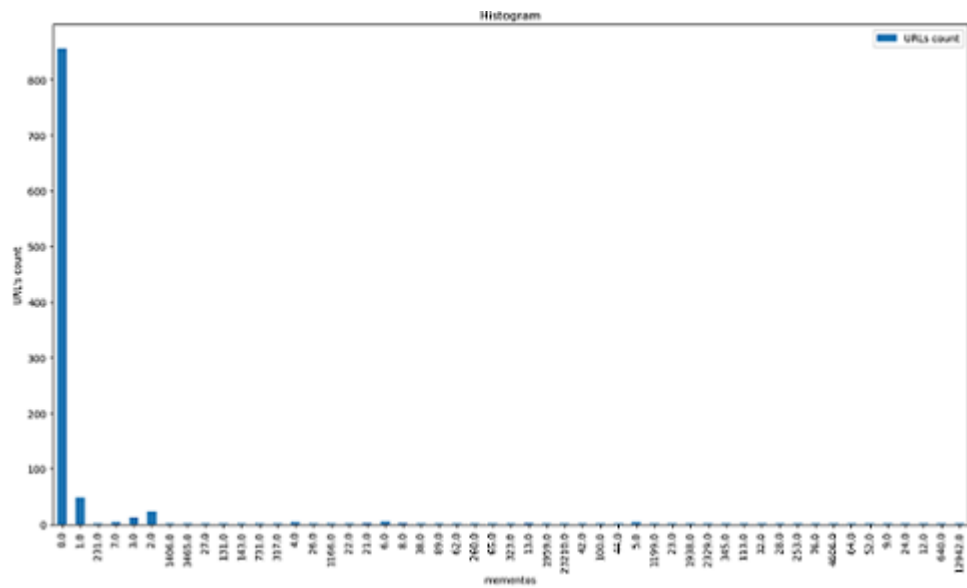
```

```

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

```

Listing 3: Python script for histogram



### 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 mementos are finding and writing all URLs of age and mementos whose status\_code 200 in **ageAndMementos.csv**. *The program shown in Listing 1.*

## Run on command line

```
1 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 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

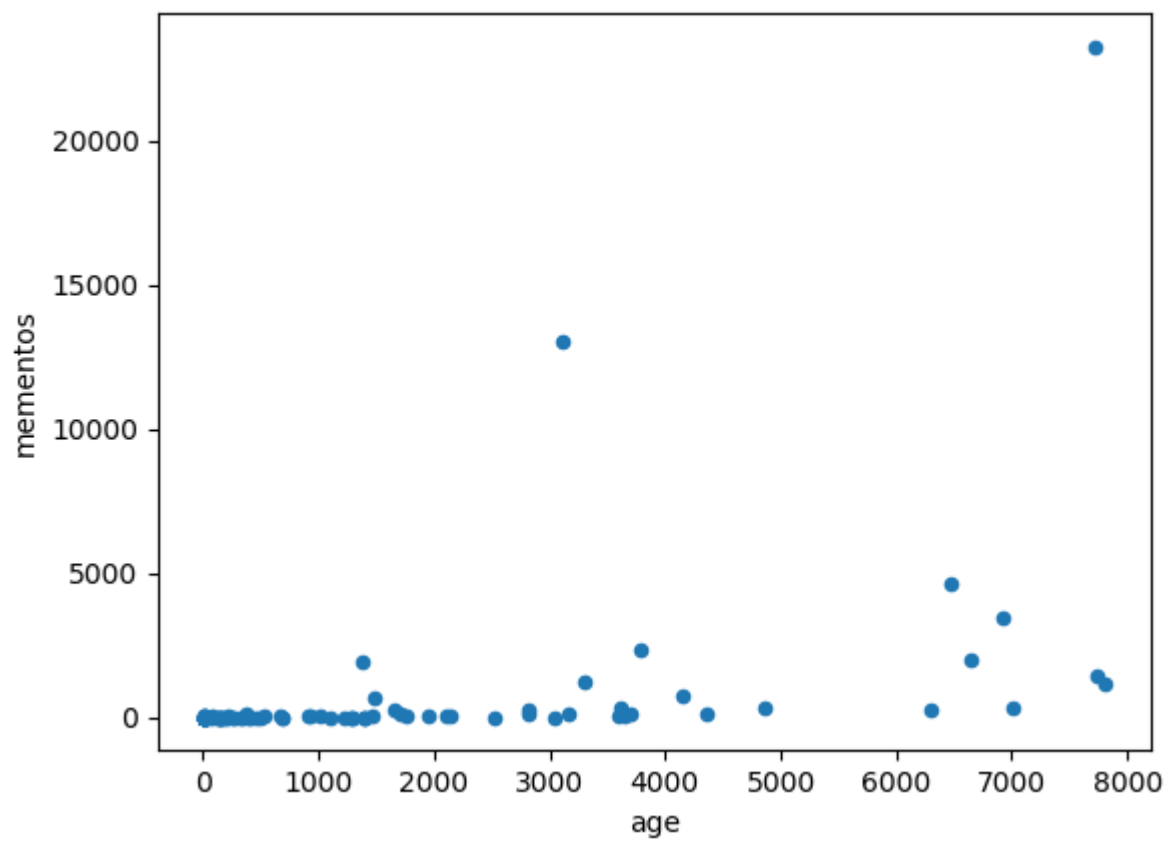


Figure 2: Scatter plot of age in days and number of mementos



## 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>