Assignment 2

CS 532: Introduction to Web Science

Spring 2018

Hrishikesh Gadkari

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.
```

Answer

To solve the above problem, I first went through the two resources [1] and [2] as mentioned on the assignment page as well as from the discussion in the class, I got to know that in order to access the twitter API and fetch URIs I need to get keys and tokens by creating a developer account on twitter. Also, the program needed to install and import tweepy library which allows the developer to stream tweets based on keywords. The following dependencies were used:

- from tweepy.streaming import StreamListener
- from tweepy import OAuthHandler
- from tweepy import Stream
- import json
- import requests
- import urllib
- import tldextract
- from urllib.parse import urlparse, urljoin
- import sys

For this assignment I reused the code which was developed for assignment 2 which made implementation much easier and quicker. The program shown in Listing 1 was written in Python 3.5 for fetching 1000 unique links, omitting links from twitter domain. The following command was used to run and save the output in **1000ulinks.txt**.

```
python tweet_crawl.py > 1000ulinks.txt
```

```
1  #Import the necessary methods from tweepy library
2  from tweepy.streaming import StreamListener
3  from tweepy import OAuthHandler
4  from tweepy import Stream
5  import json
6  import requests
7  import urllib
8  import tldextract
9  from urllib.parse import urlparse, urljoin
10 import sys
```

```
11
12 #get keys from: https://apps.twitter.com/
13 #consumer key, consumer secret, access token, access secret.
14 ckey = 'bSeaweiw68Hma0VLyeEd9se9u'
15 csecret = 'jch6kXwJociEynIHDoC8OunYLsYeRDCCjkaz0EUf3CSHzrNpSd'
16 atoken = '958819771000205312-w7L1GrIudQbONzjpfMRbwD33ITfWxnB'
17 asecret = 'MKE2Au1XVZDg1xV1F4USZsuIETm7WxgEuLACbDiQooxHG'
18 #created two blank lists for comparison
19 links1 = []
20 links2 = []
21
22 #listen to the stream of data from twitter streaming API
23 class listener(StreamListener):
24
25         def on_data(self, data):
26
27 #data in JSON format is extracted
28                 tweetJson = json.loads(data)
29
30
31 #get links from tweets
32                 links = tweetJson['entities']['urls']
33
34 #constraints set
35                 if( len(links) != 0 and tweetJson['truncated']
                   == False ):
36
37 #links passed as dictionary
38                         self.getLinksFromTweet(links)
39
40                 return True
41
42         def getLinksFromTweet(self, linksDict):
43
44                 for uri in linksDict:
45 #establish http connection for head and get request
46                         urllib.request.urlopen(uri['expanded_url
                           '])
47                         req = requests.get(uri['expanded_url'])
48                         resp = requests.head(uri['expanded_url
                           '])
49
50
51
52                         if req.status_code == 200 and resp.
                           headers.get('content-type') is not None
                           :
53 #check if expanded_url has any redirections or short links
54                                 if req.history:
```

```
55                                          final2 = req.url
56  #extract domain name using tlde
57                                          extracted2 = tldextract.
                                            extract(final2)
58                                          if extracted2.domain !=
                                            'twitter':
59                                                  o = urlparse(
                                                    final2)
60  #separate uri path and query
61                                                  final3 = urljoin
                                                    (final2, o.path)
62                                                  finalresp =
                                                    requests.head(
                                                    final3)
63  #make comparison and then append
64                                                  if final3 not in
                                                     links1:
65                                                          links1.
                                                            append(
                                                            final3)
66                                                          links2.
                                                            append(
                                                            final2)
67                                                          print(
                                                            final2)
68  #similar steps for expanded_url
69                                      else:
70                                          extracted3 = tldextract.
                                            extract(uri['
                                            expanded_url'])
71                                          if extracted3.domain !=
                                            'twitter':
72                                                  o = urlparse(uri
                                                    ['expanded_url
                                                    '])
73                                                  urilink =
                                                    urljoin(uri['
                                                    expanded_url'],
                                                    o.path)
74                                                  if urilink not
                                                    in links1:
75                                                          links1.
                                                            append(
                                                            urilink
                                                            )
76                                                          links2.
                                                            append(
                                                            uri['
                                                            expanded_url
```

```
                                                        '] )
77                                                      print(
                                                        uri['
                                                        expanded_url
                                                        '])
78
79                          if req.history:
80                                  final = req.url
81                                  extracted = tldextract.extract(
                                    final)
82                                  if extracted.domain != 'twitter
                                    ':
83                                          o1 = urlparse(final)
84                                          final4 = urljoin(final,
                                            o1.path)
85                                          finalresp = requests.
                                            head(final4)
86
87                                          if final4 not in links1:
88                                                  links1.append(
                                                    final4)
89                                                  links2.append(
                                                    final)
90                                                  print(final)
91
92
93
94
95
96
97                  return True
98
99          def on_error(self, status):
100                 print( status )
101
102                 if status_code == 420:
103 #returning False in on_data disconnects the stream
104                         return False
105                 return True
106
107
108 auth = OAuthHandler(ckey, csecret)
109 auth.set_access_token(atoken, asecret)
110
111 #for getting just 1000 links
112 while len(links2) < 1001:
113         try:
114                 twitterStream = Stream(auth, listener())
115 #multiple keywords used
```

```
116                     twitterStream.filter(track=['trump', 'football',
                         'olympics', 'jenner', 'kardashian', 'politics
                        '])
117
118             except KeyboardInterrupt:
119                     print()
120                     sys.exit (1)
121             except:
122                     pass
```

Listing 1: Python script for twitter streaming

For my keywords I chose the ones which were currently trending on twitter. It first starts by listening to the stream of data from twitters streaming API, from which data is received in JSON format. The data was parsed using entities and urls object of a tweet and also filtered them for lengths and maximum characters [3]. These links one by one were sent to getLinksFromTweet for further filtration. In the next step, HTTP head and get requests [8] were made for each of the links fetched from the previous step and checked if expandedUrl entity had any redirects or were shortened for status 200 as well as status 301. Next I used the tldextract [4] library for extracting the domain name as twitter since it had to be excluded. For checking the links to be unique I maintained two lists where in List 1 I stored all the URIs by formatting the query and parameters in links and List 2 where new links after comparing with List 1 , the links were again formed as original URIs. For this I used the urlparse and urljoin libraries [7].

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.

Answer

To solve the above problem, `http://memgator.cs.odu.edu/` was used as mentioned in the question for getting the time maps for each of the 1000 URIs. I chose the JSON format to parse the data. The following dependencies were used:

- import requests
- import json
- import os
- import csv

First I checked the response code for first few URIs. From that I came to know about the links which had zero or no mementos, returned a response code of 404. So I made HTTP get and head requests to get the json data of timemaps and saved in .json file for URIs which returned 200 response using json.dump [5] function and in .txt file as No-Mementos for those which returned a 404 response. Secondly to store the number of mementos and no of URIs in a csv file [6] named **timemaps.csv**, I used a dictionary to arrange them with respect to Key as MementoCount and URICount as value. The following program was written to implement the above problem in Python 3.5:

```
1  import requests
2  import json
3  import os
4  import csv
5
6  #directory for storing timemaps for each uri
7  os.mkdir("timemaps")
8  count = 1
9
10 #dictionary for sorting
11 dict={}
12 with open('1000ulinks.txt') as fp:
13         for line in fp:
14 #get request made to memgatorin as json
15                 url = 'http://memgator.cs.odu.edu/timemap/json/'
                       + line.strip()
16
17
18                 try:
19                         req = requests.get(url, stream=True,
                               headers={'User-Agent':'Mozilla/5.0'})
```

```
20                          resp = requests.head(url, stream=True,
                                headers={'User-Agent':'Mozilla/5.0'})
21
22
23
24                          if req.status_code == 200 :
25
26  #dumping json data into respective timemap files for URI
27                                  with open('timemaps/%s.json' %
                                        count, 'w+') as outfile:
28                                      json.dump(req.json(),
                                            outfile, sort_keys =
                                            True, indent = 4,
                                            ensure_ascii = False)
29  #getting mementocount
30                                      mementocount = resp.
                                            headers.get('X-
                                            Memento-Count')
31  #dictionary for sorting the number of mementos with      URIs
32                                      if mementocount in dict:
33                                              dict[
                                                    mementocount]
                                                     += 1
34                                      else:
35                                              dict[
                                                    mementocount]
                                                     = 1
36
37
38  #if 404 storing no mementos in for respective URI
39                          else :
40                                  with open('timemaps/%s.txt' %
                                        count, 'w+') as outfile1:
41                                      outfile1.write('No
                                            Momentos')
42                                      mementocount = 0
43                                      if mementocount in dict:
44                                              dict[
                                                    mementocount]
                                                     += 1
45                                      else:
46                                              dict[
                                                    mementocount]
                                                     = 1
47
48                          count = count + 1
49
50
51                  except:
```

```
52                          pass
53
54
55 #Dumping data into csv file in Memento-Count and URI count
       columns
56 with open('timemaps.csv', 'w', newline='') as csvfile:
57         fieldnames = ['Memento_Count', 'URI_Count']
58         writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
59         writer.writeheader()
60
61         for key, value in dict.items():
62                 writer.writerow({'Memento_Count': key, '
                       URI_Count': value})
63         csvfile.close()
```

Listing 2: Python Script for downloading Timemaps

There were few errors while plotting the histogram, for which I had to write a small piece of code shown below, for converting the **timemaps.csv** file into a single column file named **counts.csv**.

```
1  import csv
2
3  with open('counts.csv', 'a+', newline='') as csvfile:
4          spamwriter = csv.writer(csvfile, delimiter=' ',
               quotechar='|', quoting=csv.QUOTE_MINIMAL)
5          spamwriter.writerow('Memento_Count')
6          csvfile.close()
7
8  with open('names.csv', newline='') as csvfile:
9          field = ['Memento_Count', 'URI_Count']
10         reader = csv.DictReader(csvfile, fieldnames=field)
11         for row in reader:
12                 #count = row['Memento']
13                 if row['URI_Count'] == "URI_Count":
14                         continue
15         #               print(row['Memento_Count'])
16                 count = row['URI_Count']
17                 count = int(count)
18
19                 for row1 in range(0,count):
20
21                         with open('counts.csv', 'a+', newline
                               ='') as csvfile:
22                                 spamwriter = csv.writer(csvfile,
                                       delimiter=' ', quotechar
                                       ='|', quoting=csv.
                                       QUOTE_MINIMAL)
```

```
23                                  spamwriter.writerow(row['
                                        Memento_Count'])
24                                  csvfile.close()
```

Listing 3: Conversion

I then created a simple histogram , using R [9] shown in Listing 4, of URIs vs. Number of Mementos as shown in Figure 1. From the histogram observation we come to know that majority of URIs had zero or low counts of mementos.

```
1  setwd(getwd())
2  # csv dataframe of URI, number of mementos
3  numMementos <- read.table(header = TRUE,sep = ",",'counting.csv
       ')
4  # histogram
5
6  hgram <- hist(numMementos$Memento_Count, col = "gray", breaks =
       20, main="URIs vs. Number of Mementos",xlab = "Number of
       Mementos",ylab = "Number of URIs")
7  # add count labels
8  text(hgram$mids,hgram$counts, adj=c(0.5, -0.5), labels=
       hgram$counts)
```

Listing 4: R script for creating Histogram

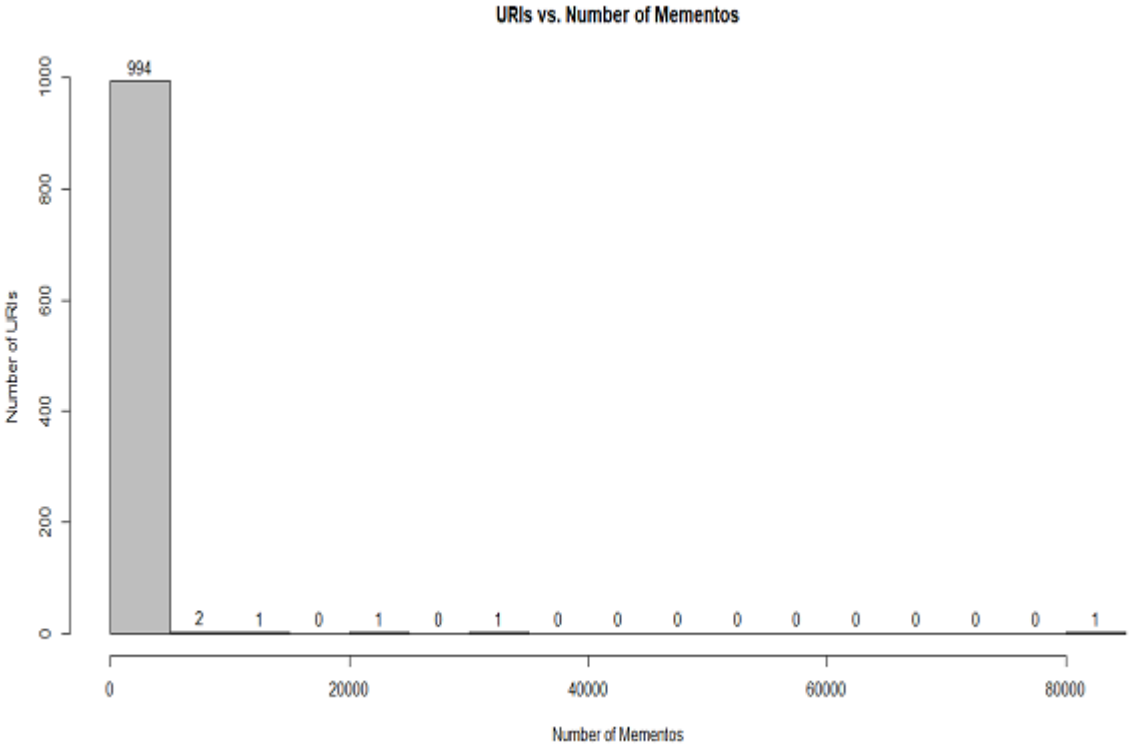

Figure 1: Histogram of Number of URIs vs. Number of Mementos

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
```

Answer

To solve the above problem, used the carbon dating tool [11] 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 saved **1000ulinks.txt** file. I first checked first few links and came to know that links with zero mementos either no creation date or had a creation date whereas URIs that had no date estimates didn't contain any mementos. The following dependencies were used:

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

The program shown in Listing 5 was written in Python 3.5 and the output was saved in **dates.csv** while with column names as Age and MementoCount.

```
1  import requests
2  import json
3  from datetime import datetime
4  import csv
5
6
7  #counts initialized
8
9  count = 1
10 count_nocd = 0
11 count_nomem = 0
12
13 #write first row as headers in csv file
14 with open('dates.csv', 'a+', newline='') as csvfile:
15         spamwriter = csv.writer(csvfile, delimiter=',',
               quotechar='|')
16         spamwriter.writerow(['Age','Memento-Count'])
17         csvfile.close()
18
19 with open('1000ulinks.txt') as fp:
20         for line in fp:
21                 try:
22 #requests made to memgator as well as carbon date url
23                         urlmem = 'http://memgator.cs.odu.edu/
                               timemap/json/' + line.strip()
```

```
24                          urlcd = 'http://cd.cs.odu.edu/cd/' +
                                line.strip()
25                          reqmem = requests.head(urlmem, stream=
                                True, headers={'User-Agent':'Mozilla
                                /5.0'})
26
27  #if no mementos
28                          if reqmem.status_code == 404:
29                                  reqcd = requests.get(urlcd,
                                        stream=True, headers={'User-
                                        Agent':'Mozilla/5.0'})
30                                  data = reqcd.json()
31  #get the count for no date estimates if any
32                                  if data['estimated-creation-date
                                        '] == "":
33                                          count_nocd = count_nocd
                                                + 1
34                                          count_nomem =
                                                count_nomem +1
35
36                                  else:
37                                          count_nomem =
                                                count_nomem +1
38
39
40
41                          else:
42                                  reqcd = requests.get(urlcd,
                                        stream=True, headers={'User-
                                        Agent':'Mozilla/5.0'})
43                                  data = reqcd.json()
44  #check if any uri having memento count has no estimate date
        creation
45                                  if data['estimated-creation-date
                                        '] == "":
46                                          count_nocd = count_nocd
                                                + 1
47  #else get the age in days using datetime
48                                  else:
49                                          memento_count = reqmem.
                                                headers.get('X-
                                                Memento-Count')
50                                          birthday = data['
                                                estimated-creation-
                                                date']
51                                          birthday = datetime.
                                                strptime(birthday, '%
                                                Y-%m-%dT%X')
```

```
52                                                  age = datetime.now() -
                                                        birthday
53  #dump data into csv file
54                                                  with open('dates.csv', '
                                                        a+', newline='') as
                                                        csvfile:
55                                                      spamwriter = csv
                                                            .writer(
                                                            csvfile,
                                                            delimiter
                                                            =',',
                                                            quotechar
                                                            ='|')
56                                                      spamwriter.
                                                            writerow([age
                                                            .days,
                                                            memento_count
                                                            ])
57                                                      csvfile.close()
58
59  #the other data required dumped in calc.csv file
60                              with open('calc.csv', 'w', newline='')
                                    as csvfile:
61                                      spamwriter = csv.writer(csvfile,
                                            delimiter=',', quotechar
                                            ='|')
62                                      spamwriter.writerow(['Total URIs
                                            ','No Mementos','No Date
                                            Estimate'])
63                                      spamwriter.writerow(['1000',
                                            count_nomem,count_nocd])
64                                      csvfile.close()
65
66
67
68                  except:
69                          pass
```

Listing 5: Python script for finding the carbon date

Also the program in Listing 5, calculated the following values and saved in **calc.csv** file.

- total URIs: 1000
- no mementos: 533
- no date estimate: 82

The following R [10] program was used to create the scatterplot for Age in Days vs number of Mementos, shown in Figure 2

```
setwd(getwd())
# Merged Dataset with memento count and days
mementoDays <- read.table(header = TRUE, sep = ",",'thirdq.csv')
# filtered set w/ atleast 1 memento
#withMementos <- subset(mementoDays, mementoDays$V2 > 0)
#emptyDate <- subset(mementoDays, is.na(mementoDays$V3))
plot(mementoDays$Age,mementoDays$Memento_Count,xlab="Age in Days
    ",ylab="Number of Mementos",main="Scatter Plot for Days vs.
    Mementos")
```

Listing 6: R Script to create scatterplot

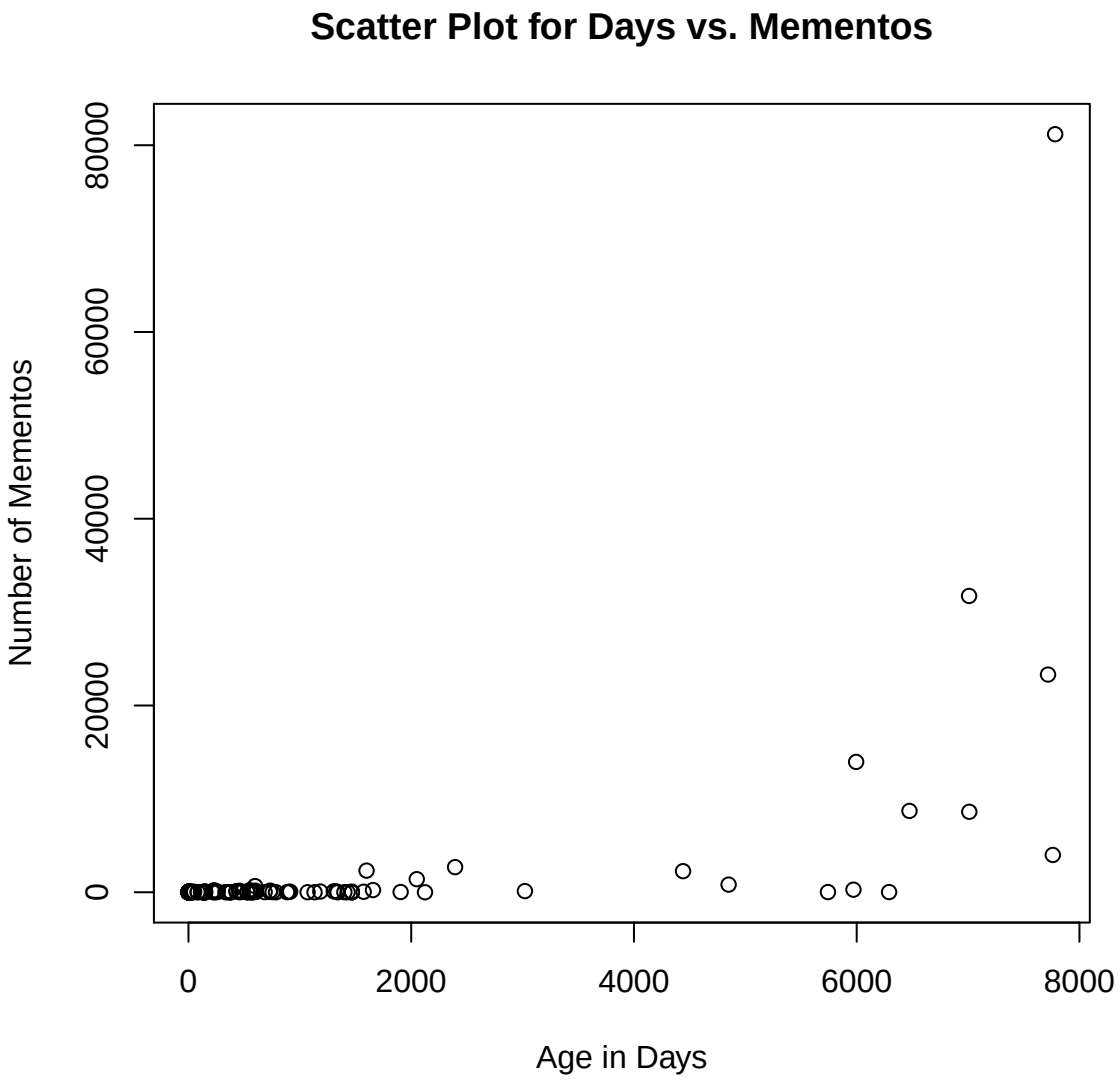

Figure 2: Scatter plot of age in days and number of mementos

References

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

[2] Moujahid, Adil. "An Introduction to Text Mining Using Twitter Streaming API and Python." An Introduction to Text Mining Using Twitter Streaming API and Python // Adil Moujahid // Data Analytics and More. N.p., 21 July 2014. Web. 13 Feb. 2018. http://adilmoujahid.com/posts/2014/07/twitter-analytics/

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

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

[5] 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

[6] 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

[7] 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

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

[9] R Documentation."function — R Documentation graphics", Web. 13 Feb, 2018. https://www.rdocumentation.org/packages/graphics/versions/3.4.3/topics/hist

[10] R Documentation."function — R Documentation lessR", Web. 13 Feb, 2018. https://www.rdocumentation.org/packages/lessR/versions/2.5/topics/ScatterPlot

[11] Zetan, Li. "2016-09-20: Carbon Dating the Web, Version 3.0." 2016-09-20: Carbon Dating the Web, Version 3.0. N.p., 20 Sept. 2016. Web. 08 Feb. 2017. http://ws-dl.blogspot.com/2016/09/2016-09-20-carbon-dating-web-version-30.html