Introduction to Web Science 101: Assignment #2

Dr. Nelson

Alexander Nwala

Thursday, September 25, 2014

Contents

Problem 1

Write a Python program that extracts 1000 unique links from Twitter. You might want to take a look at:
http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/
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 different shortened URIs for www.cnn.com. For example,
http://cnn.it/1cTNZ3V http://t.co/BiYdsGotTd
Both ultimately redirect to cnn.com, so they count as only 1 unique URI. Also note the second URI redirects twice – don't stop at the first redirect.
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).
Hold on to this collection – we'll use it later throughout the semester.

Listing 1: Main Block

```
#Main block retrieves live tweets (sinceIDValue = 0) from twitter
searchQuery = 0
sinceIDValue = 'www%2E-filter:link'
for tweet in tweepy.Cursor(api.search, q=searchQuery).items(30):
    #print tweet.id, tweet.created_at
    if( tweet.id > sinceIDValue ):
        sinceIDValue = tweet.id

    expandedUrlsList = getUrisArray(tweet.entities['urls'])

    if(len(expandedUrlsList) > 0):
        for u in expandedUrlsList:

            if(len(u) > 0):
                u = u.lower().strip()

                #if u not in modifiedUrlsDataFileLongString:
                if isInsideList(modifiedUrlsDataFileLongString, u) == False:
                    print "...adding: ", u
                    urlsDataFile.write( u + ', ' + str(tweet.id) + ', ' + str(
                        datetime.datetime.now()) + '\n' )
                    modifiedUrlsDataFileLongString.append(u)
```

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

1. **Establish connection between application and Twitter:** This was achieved through tweepy [1] which provides a python wrapper for the Twitter api.

2. **Authenticate application:** This was achieved by registering the application with a Twitter account and generating OAuth [2] access tokens

3. **Fetch tweets with links:** This was achived through the Twitter api search utility. With this, I was able to search for tweets containing "www.". However, since we are only interested in tweets with links, I applied the link filter to the search query "www.-filter:link". This query was encoded as outlined by Listing 2:

```
www%2E-filter:link
```

Listing 2: Query Twitter

```
#Query Twitter Code Snipped
searchQuery = 'www%2E-filter:link'
for tweet in tweepy.Cursor(api.search, q=searchQuery).items(30):
    ...
```

4. **Expand urls:** Given that Twitter shortens urls embedded in tweets, the application expands the urls by accessing the "expanded_url" attribute from the tweet json. However, since the expanded url could yet again be a short url from a third party url shortening service such as Bitly, the url is yet again expanded as shown in Listing 3.

Listing 3: Expand Url

```
#Expand url
def followTheRedirectCurl(url):
    if(len(url) > 0):
        try:
            r = requests.head(url, allow_redirects=True)
            return r.url
        except:
            return ''
    else:
        return ''
```

5. **Send request continuously in a rate friendly manner:** Given that Twitter imposes a search rate limit [3] of 180 request per 15 minute window, the application honors the limit by sending request continuously at a rate limit of 2 requests (implemented by retrieving 2 pages for every request) per 15 seconds. This means the application sends 120 requests per 15 minute window.

Listing 4: Query Twitter

```
#Send requests in a rate-friendly manner
lengthOfLinksFile = 0
while(lengthOfLinksFile < 1500):
    lengthOfLinksFile = fetchResultsFromTwitter(searchQuery='www%2E-filter:link',
        numberOfPullRequestsToInitiate=1, sinceIDValue=0)
    print "...lengthOfLinksFile: ", lengthOfLinksFile
    time.sleep(15)
```

Problem 2

Download the TimeMaps for each of the target URIs. We'll use the mementoweb.org Aggregator, so for example:

URI-R = http://www.cs.odu.edu/

URI-T = http://mementoweb.org/timemap/link/http://www.cs.odu.edu/

```
You could use the cs.odu.edu aggregator:

URI-T = http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/http://www.cs.odu.edu/

But be sure to say which aggregator you use -- they are likely to give
different answers.

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.

See: http://en.wikipedia.org/wiki/Histogram

Note that the TimeMaps can span multiple pages.  Look for links like:

<http://mementoweb.org/timemap/link/1000/http://www.cnn.com/>;rel="timemap";
type="application/link-format"; from ="Sun, 08 Jul 2001 21:30:54 GMT"

This indicates another page of the TimeMap is available.  There can be
many pages to a TimeMap.
```

SOLUTION

With the use of the cs.odu.edu aggregator, the application downloads timemaps for each URI, taking into account URIs which span multiple pages by considering this signature:

```
>;rel="timemap"
```

This is outlined by Listing 5. The memento count for each URI is derived by counting the number lines in the page list. This distribution is summarized by Chart 1.

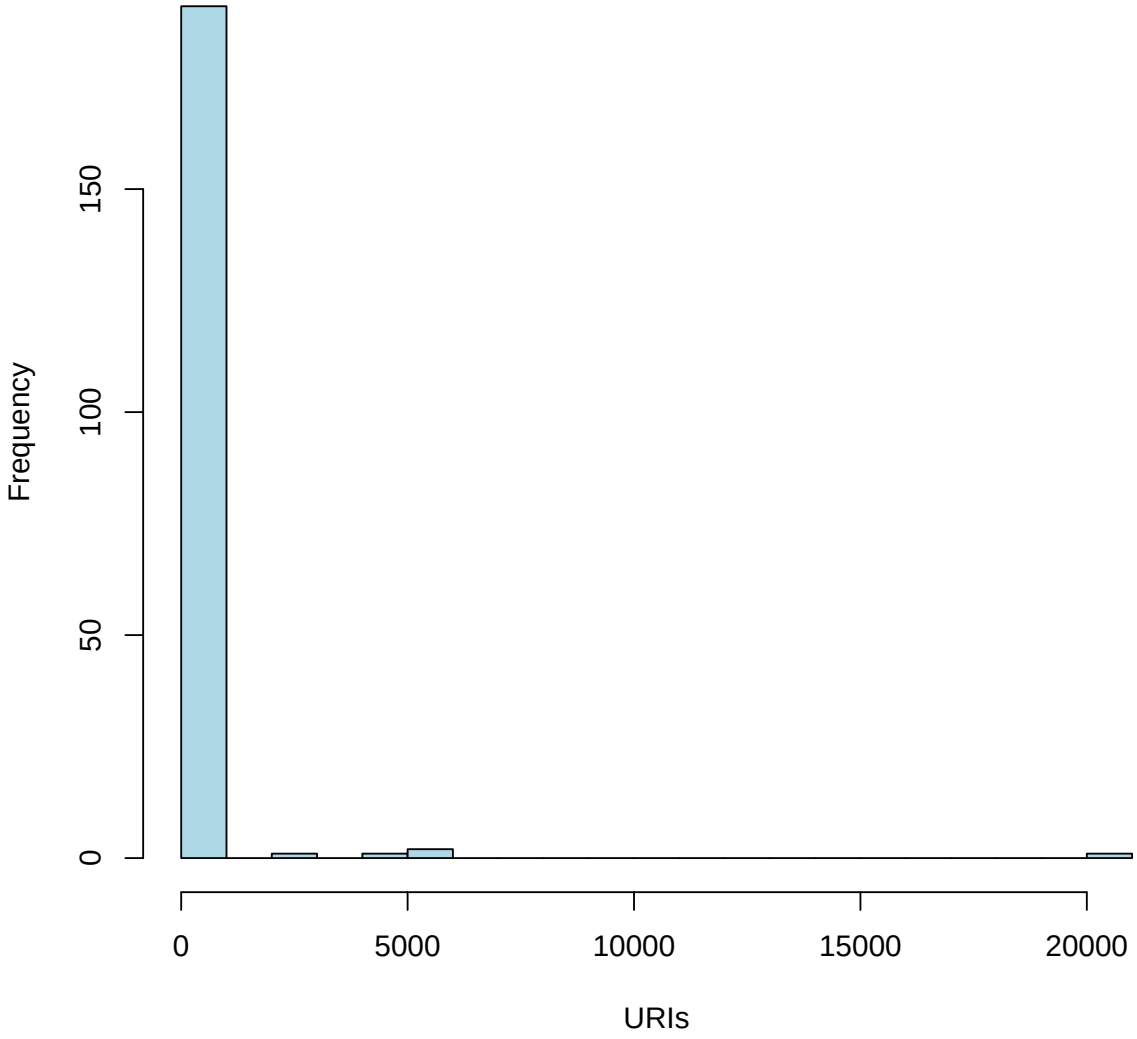
```
The file URI-MEM-DATE-AGE.txt contains tuples of form:
    <URI, MementoCount, Estimated Created Date, Age>
```

Listing 5: Query Twitter

```
#Get mementos
def getMementosPages(url):
     if(len(url)>0):
          pages = []
          timemapCount = 0
          timemapPrefix = 'http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/' + url
          while( True ):
               co = 'curl --silent ' + timemapPrefix
               page = commands.getoutput(co)
               pages.append(page)

               indexOfRelTimemapMarker = page.rfind('>;rel="timemap"')

               if( indexOfRelTimemapMarker == -1 ):
                    break
               else:
                    #retrieve next timemap for next page of mementos e.g retrieve url
                        from <http://mementoproxy.cs.odu.edu/aggr/timemap/link/10001/
                        http://www.cnn.com>;rel="timemap"
```

```
                    i = indexOfRelTimemapMarker -1
                    timemapPrefix = ''
                    while( i > -1 ):
                        if(page[i] != '<'):
                            timemapPrefix = page[i] + timemapPrefix
                        else:
                            break
                        i = i - 1

        return pages
    else:
        print "url length = 0"
```

Chart 1: Distribution of URIs



Problem 3

```
Estimate the age of each of the 1000 URIs using the "Carbon Date" tool:

http://ws-dl.blogspot.com/2013/04/2013-04-19-carbon-dating-web.html

Note: you'll have better luck downloading and installing the tool
rather than using the web service (which will run slowly and likely
be unreliable).

For URIs that have > 0 Mementos and an estimated creation date,
create a graph with age (in days) on one axis and number of mementos
on the other.
```

SOLUTION

Based on the criteria of considering only URIs with an estimated creation date and memento count which exceeds 0, of 1000 instances, 196 URIs fulfilled this condition. This shortage is not surprising since the tweets were "born" about the same time, and most likely focus upon current events. As seen from the data in Table 1 and Chart 2, there is a positive correlation between Age and the Memento count.

```
The file AGE-MEMENTOCOUNT.txt contains tuples of form:
    <Age, MementoCount>
```

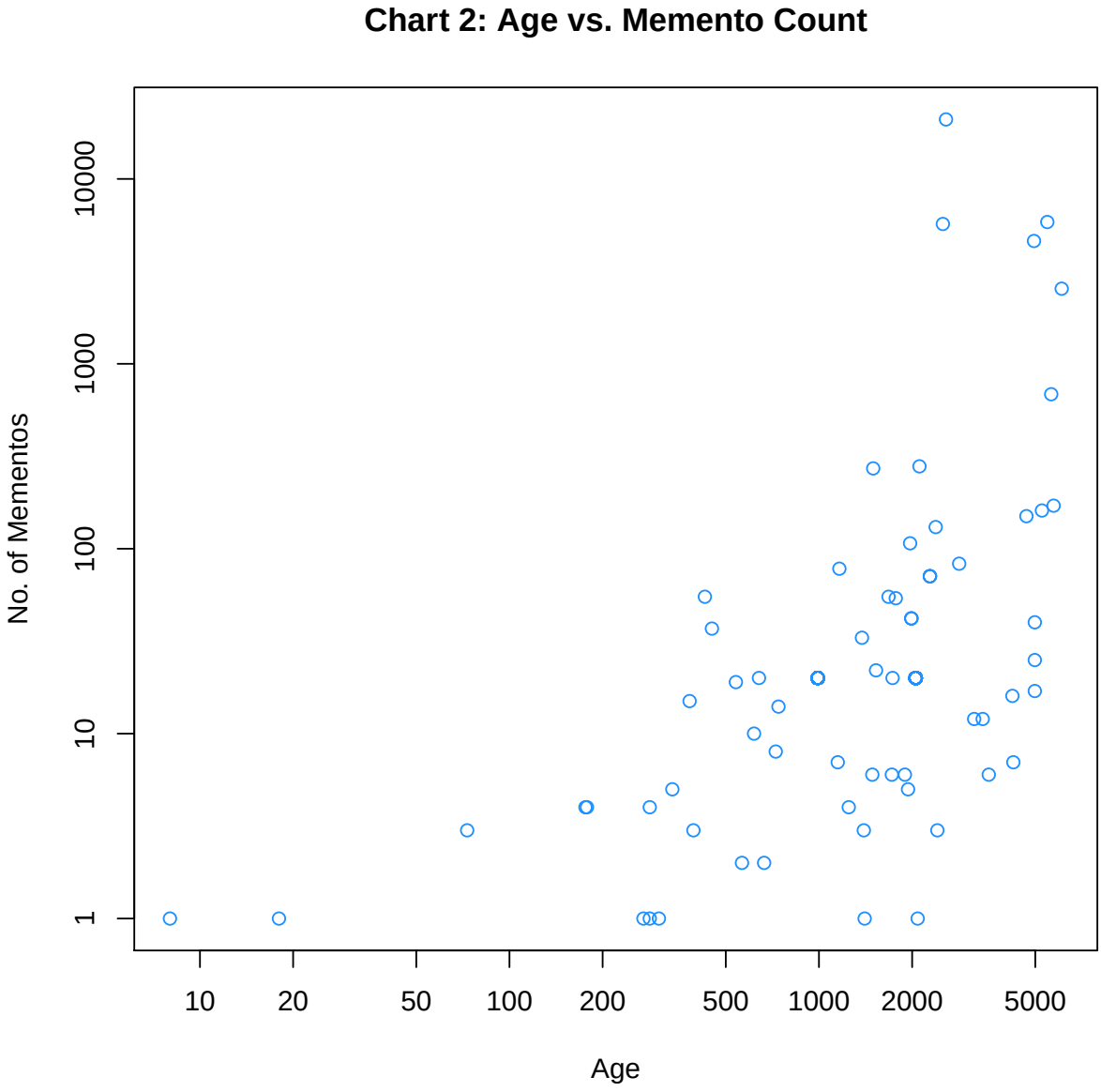


Chart 2: Age vs. Memento Count
No. of Mementos
10000
1000
100
10
1
10
20
50
100
200
500
1000
2000
5000
Age

Table 1: Age vs. Memento

Item	Age	Memento Count
1	6085	2549
2	5731	171
3	5628	685
4	5467	5848
5	5253	161
6	4982	25
7	4982	17
8	4982	40
9	4952	4618
10	4680	150
11	4246	7
12	4217	16
13	3537	6
14	3381	12
15	3173	12
16	2838	83
17	2573	20990
18	2514	5708
19	2413	3
20	2383	131
21	2283	71
22	2283	71
23	2283	71
24	2282	71
25	2112	279
26	2084	1
27	2054	20
...	...	...
88	2054	20
89	2043	20
90	1988	42
91	1988	42
92	1988	42
93	1970	107
94	1940	5
...	...	...
187	336	5
188	304	1
189	284	4
190	284	1
191	271	1
192	178	4
193	176	4
194	73	3
195	18	1
196	8	1

References

[1] http://www.tweepy.org/

[2] https://dev.twitter.com/oauth

[3] https://dev.twitter.com/rest/public/rate-limiting