Introduction to Web Science: Assignment #2

Due on Thursday, February 9, 2017

Dr. Nelson

Udochukwu Nweke

Contents

Problem 1 **3**

Problem 2 **5**

Problem 3 **7**

Problem 1

Write a python program that extracts 1000 unique links from twitter.

Listing 1: Code for extracting links from Twitter

```
import tweepy
import commands
import time

import os, sys

consumer_key = '01oEm0j9Y52TFtjA5wgRHt9z6'
consumer_secret = 'MWPYnHvrezvhuuFQkOQgnBMYwXZD6SJpnoI8QlE7ZajgQMUKKs'
access_token = '154076252-uK6XnhweIkuc0qIvsNmGiiRebLqvYHbtWDgA5PBi'
access_token_secret = 'LDA5Qel3UQtIwUhvAZLffCGZ9pmmc7wkFOL5k0xx5Yt9O'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

for tweet in tweepy.Cursor(api.search, q=searchKey,
since_id=sinceID[searchKey]).items(30):
                        if( tweet.truncated == True ):
                             continue

                        sinceID[searchKey] = tweet.id
                        for url in tweet.entities['urls']:
                             strToWrite = str(tweet.id) +
                        ' <**> ' + url['expanded_url'] + '\n'
                             print '\t', tweet.id, url['expanded_url']
                        outfile.write(strToWrite)


                    outfile.close()

                    print '...sleeping for 15 seconds'
                    time.sleep(15)


def unshortenURL(url):

     curlRequest = 'curl -IL --silent ' + url + ' | egrep -i "(HTTP/1.1|^location:)"'
     curlOutput = commands.getoutput(curlRequest)
     curlOutput = curlOutput.splitlines()

     if( len(curlOutput) < 2 ):
          return ''


     if( curlOutput[-1].lower() == 'http/1.1 200 ok' ):
          return curlOutput[-2].split(' ')[-1].strip()
```

Problem 1 continued on next page...

```
50        return ''

def removeDups():

        infile = open('allTweets.txt', 'r')
55      lines = infile.readlines()
        infile.close()

        dedupDict = {}

60      print 'len(lines):', len(lines)

        for line in lines:
                line = line.strip()

65              line = line.split(' <**> ')

                if( len(line) < 1 ):
                        continue

70              id = line[0]
                url = line[1]
```

1. I used Twitter search API to periodically search for tweets with search terms: "Trump", "Obama", "Sports", and "Music". I only searched for tweets that were not truncated. If the tweet's truncated flag, was set, I skipped the tweet (Listing 1 line 19 demonstrates how I avoided truncated Tweets).

2. For each of these tweets, I extracted links (expanded links to remove t.co) from tweets that have not been truncated, because truncated tweets (which have a self reference) will require another API call with *tweet_mode=extended* option in other to get a link that does not reference itself.

Listing 2 is a snippet of the extracted raw file. The complete file is in *allLinks.txt* file.

Listing 2: A snippet of allLinks.txt containing 6 links extracted from tweets

```
1. http://route.overnewser.com/nascarworldnews/?url=http://www.foxsport
s.com/nascar/story/nascar-monster-energy-to-replace-drivers-names-on-fro
nt-windshield-012717%3fcmpid=feed:-sports-CQ-RSS-Feed&utm_source=twitter
-tools&utm_medium=NascarWorldNews&utm_campaign=article
5
2. https://www.twitch.tv/giiggly

3. http://www.jsonline.com/story/sports/nba/bucks/2017/01/29/trumps-ban
-sparks-rebuke-bucks-official/97091812/?hootPostID=5d701bb5e334d49
10 896aa8678def1d1f1&from=global&sessionKey=&autologin=

4. https://www.washingtonpost.com/sports/redskins/the-nfl-needs-a-great-
super-bowl-to-rescue-a-snooze-of-a-season/2017/01/29/c86677ec-e657-11
e6-b82f-687d6e6a3e7c_story.html?tid=sm_tw_ps
15
5. http://ktla.com/2017/01/29/7-countries-targeted-in-trumps-executive-
```

```
order-initially-identified-as-countries-of-concern-under-obama-administration/

6. https://www.buzzfeed.com/leticiamiranda/mexicans-are-boycotting-us-
products-to-protest-trumps-wall-t?utm_term=4ldqpfp&bftw#4ldqpfp
```

3. I grabbed 5,220 links and removed duplicate links by using a dictionary to store key value pairs where (url-key and id-value)since dictionaries do not permit duplicates. I unshortened the urls to get the final links.

Listing 1, line 52 shows how I removed duplicates in other to get unique urls with *removeDups()*. After removing duplicate links the 5,220 urls dropped to 1,900 urls. Hence, I grabbed only a 1000 url and saved in *1000UniqueUrls.txt* file. .

Problem 2

2. Download the Timemaps for each of the target URLs. 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 would prefer to parse)
Create a histogram* of URIs vs. number of mementos (as computed from the TimeMaps). For example, 100 URIs with 0 Memntos, 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.
* = `http://en.wikipedia.org/wiki/Histogram`
What's a TimeMap? See:`http://www.mementoweb.org/Wiki/Histogram/guide/quick-intro/`
And the Week 4 lecture.

Listing 3: Code Snippet For Problem 2

```
import commands
from P1 import errorMessage, readFromFile
import json
import os

def downloadTimemap(url):

     try:
          curlRequest = 'curl --silent http://memgator.cs.odu.edu/timemap/json/'
          + url
          curlOutput = commands.getoutput(curlRequest)
          return curlOutput
     except:
          errorMessage()
          return ''

def downloadTimemapsAndSave(urls):

     for i in range(0, len(lines)):
          url = lines[i].strip()
          print i, url
```

```
            timemapText = downloadTimemap(url)

            try:
                json.loads(timemapText)

                filename = './Timemaps/' + str(i) + '.json'
                saveText(filename, timemapText)

                print '\tsaved filename:', filename
            except:
                errorMessage()

            print ''

def countMementos(urls):

    outfile = open('./MementoCounts.csv', 'w')
    outfile.write('ID,MementoCount\n')

    for i in range(0, len(lines)):
        url = lines[i].strip()
        print i, url

        try:

            filename = './Timemaps/' + str(i) + '.json'
            if( os.path.exists(filename) != True ):
                continue

            timemapText = readFromFile(filename)
            timemapText = json.loads(timemapText)

            memCount = len(timemapText['mementos']['list'])
            print '\tmemCount:', memCount
            outfile.write(str(i) + ', ' + str(memCount) + '\n')
        except:
            errorMessage()

        print ''

    outfile.close()
```

1. I downloaded TimeMaps for the 1000 unique URLs by using curl to dereference the concatenation of the ODU Memento Aggregator and each url - Listing 3 (*function downloadTimemap*)

2. After downloading the TimeMaps, I saved only TimeMaps that existed for URLs (because some URLs did not have TimeMaps) - Listing 3 (*function downloadTimemapsAndSave*)

3. I wrote a function called *countMementos* which extracted and counted mementos for each link - Listing 3

4. I wrote the result into a CSV file as shown in listing 5. The creationDate.csv file has everything.

5. I wrote an R program to plot the histogram.

The R code snippet in listing 4 demonstrates how I used R to plot the histogram

Listing 4: R code For The Histogram

```
histData <-  read.csv('./MementoCounts.csv', head=TRUE, sep=",")
hist(histData$MementoCount, xlab='Number of Mementos', ylab='Frequency
of Occurence',
main='Number of Ements vs. Frequency of Occurence')
```

Listing 5: creationDate.csv

```
                                    Links                MementoCount
1. http://route.overnewser.com/                                 4
nascarworldnews/?url=http://
www.foxsports.com/nascar/story/n
ascar-monster-energy-to-replace-drivers-names
-on-front-windshield-012717%3fcmpid
=feed:-sports-CQ-RSS-Feed&utm_source=twitter-
tools&utm_medium=NascarWorldNews&utm_campaign=article

2. https://www.twitch.tv/giiggly                                 2

3. http://www.jsonline.com/story/sp                             2
orts/nba/bucks/2017/01/
29/trumps-ban-sparks-rebuke-bucks-o
fficial/97091812/
?hootPostID=5d701bb5e334d49896aa8678d
ef1d1f1&from=global&sessionKey=&autologin=

4. https://www.washingtonpost.com/sports/r                     38
edskins/the-nfl-needs-a-great-super-
bowl-to-rescue-a-snooze-of-a-season/2017/0
1/29/c86677ec-e657-11e6-b82f-687d6e6
a3e7c_story.html?tid=sm_tw_ps

5.http://ktla.com/2017/01/29/7-countries-ta                     9
rgeted-in-trumps-executive-order-
initially-identified-as-countries-of-concer
n-under-obama-administration/
```

Problem 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

But it will inevitably crash when everyone tries to use it at the last minute.

For URIs that have > 0 Mementos and 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, not all URIs will have an estimated creation date. Show how many fall into either categories. For example,

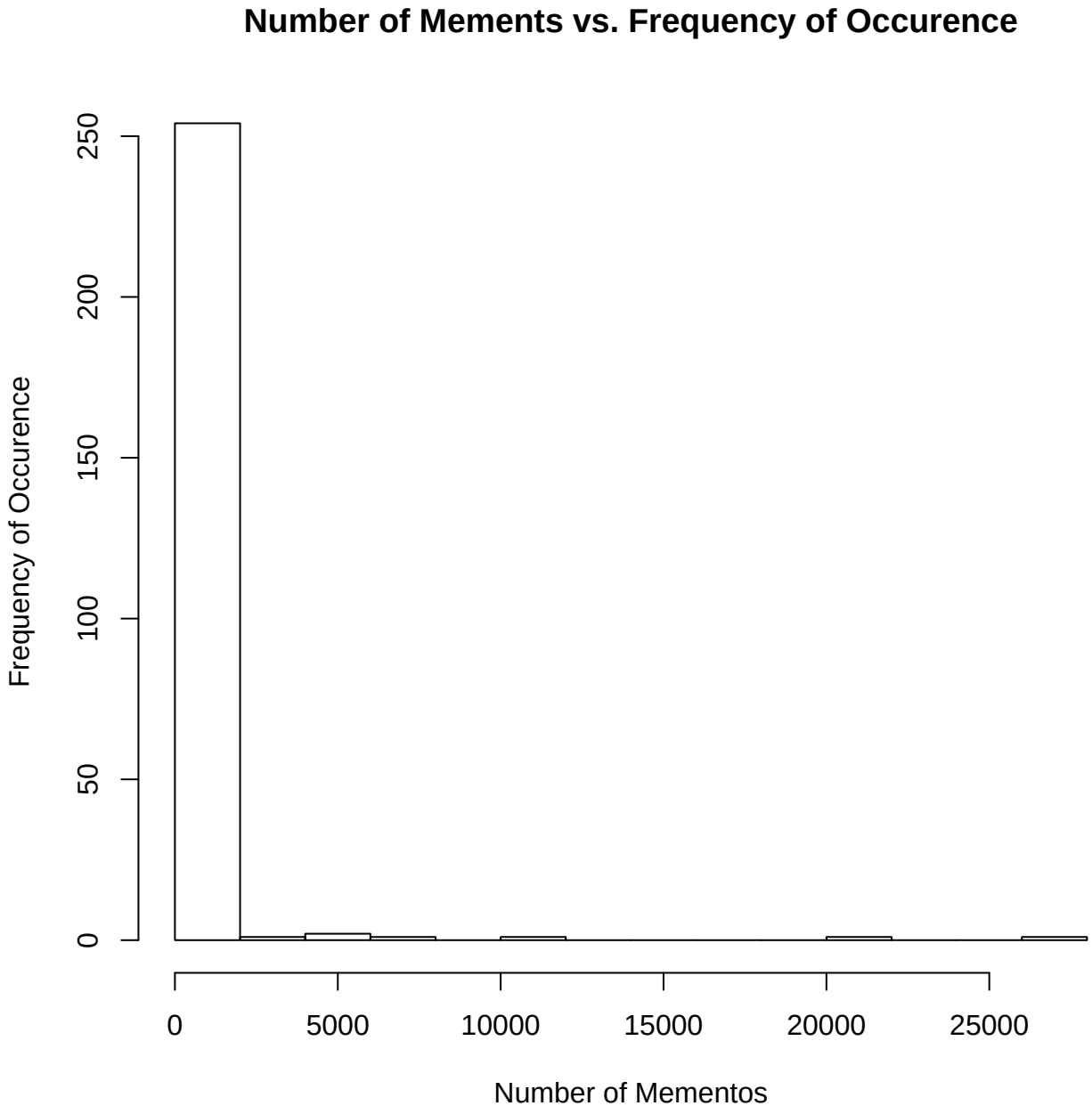
total URIs: 1000
no mementos: 137
no date estimate: 212

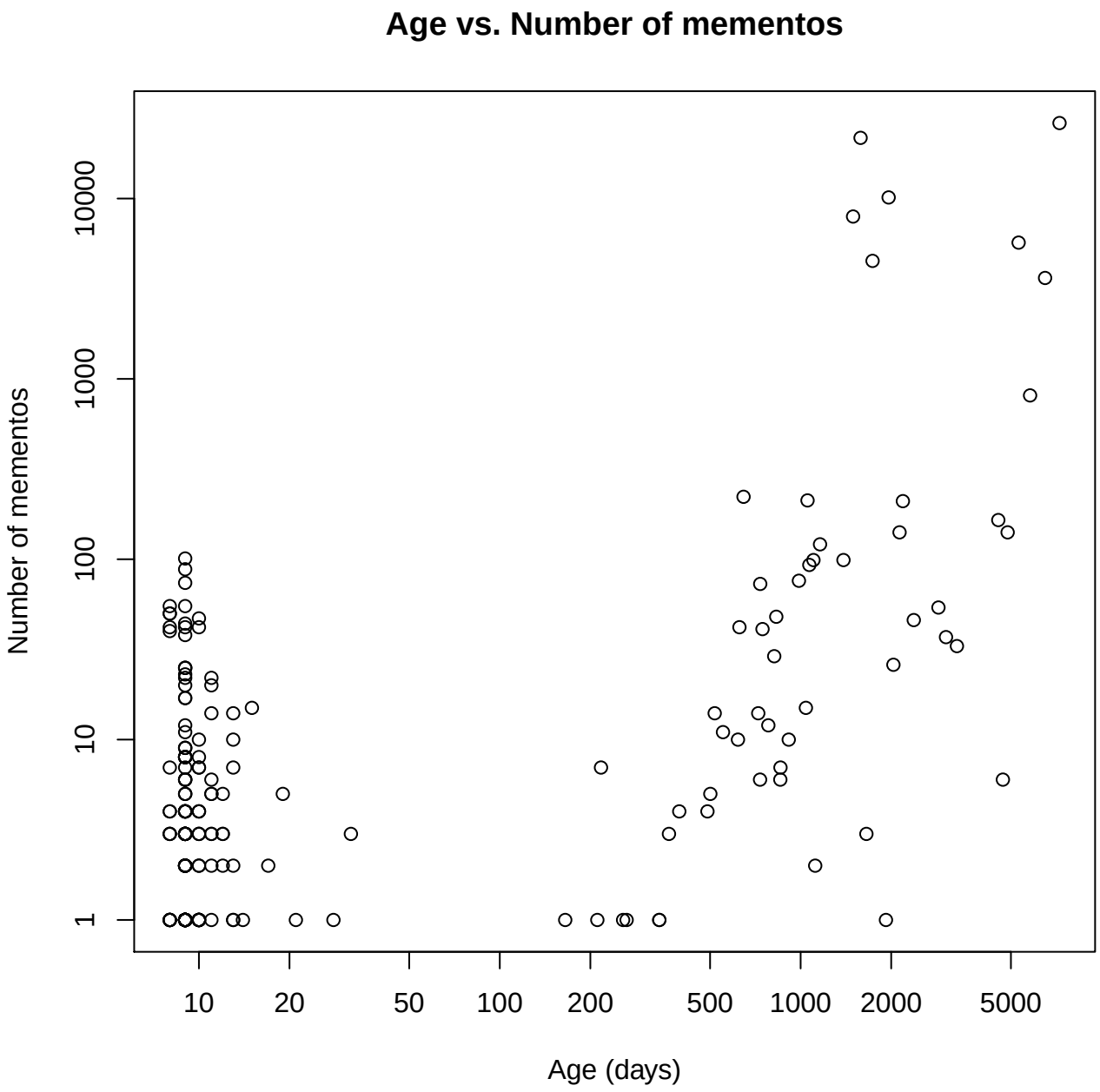
Listing 6: Extract Memento count and Age (CarbonDate)

```
import commands
import json
import os, sys
from datetime import datetime

def getCreationDateForURI(url):

    try:
        dockerRequest = 'docker run --rm -it carbon ./main.py -l search ' + url
        dockerOutput = commands.getoutput(dockerRequest)

        indexOfBrace = dockerOutput.find('{')
        if( indexOfBrace == -1 ):
            return ''

        dockerOutput = dockerOutput[indexOfBrace:]
        dockerOutput = json.loads(dockerOutput)

        return dockerOutput['Estimated Creation Date']
    except:
        errorMessage()
        return ''

def getCreationDates(urls):

    outfile = open('./CreationDates.csv', 'a')

    for i in range(0, len(lines)):
        url = lines[i].strip()
        print i, url

        try:
            creationDate = getCreationDateForURI(url)

            if( len(creationDate) == 0 ):
                continue

            creationDateObj = datetime.strptime(creationDate,
            '%Y-%m-%dT%H:%M:%S')
            delta = datetime.now() - creationDateObj
```

```
                print 'age: ', delta

                outfile.write(str(i) + ', ' + str(delta.days) + '\n')
            except:
                errorMessage()

            print ''

    outfile.close()

def matchMementoCountWithCreationDates():

    memCountLines = []
    ageLines = []

    try:
        infile = open('../MementoCounts.csv', 'r')
        memCountLines = infile.readlines()
        del memCountLines[0]
        infile.close()

        infile = open('./CreationDates.csv', 'r')
        ageLines = infile.readlines()
        del ageLines[0]
        infile.close()
    except:
        errorMessage()
        return

    #key is URL ID, value is MementoCounts
    memCountDict = {}

    #key is URL ID, value is Age
    ageDict = {}

    memCountAgeDict = {}

    for line in memCountLines:
        IDMemCount = line.split(', ')
        memCountDict[ IDMemCount[0].strip() ] = IDMemCount[1].strip()

    for line in ageLines:
        IDAge = line.split(', ')
        ageDict[ IDAge[0].strip() ] = IDAge[1].strip()


    for ID, memCount in memCountDict.items():
        if( ID in ageDict ):
            memCountAgeDict[ID] = { 'memCount': memCount, 'age': ageDict[ID] }


    print 'ID,MementoCount,Age'
    for ID, memCountAge in memCountAgeDict.items():
```

```
            toPrint = ID + ', ' + memCountAge['memCount'] + ', ' + memCountAge['age']
95          print toPrint

matchMementoCountWithCreationDates()
```





1. In order to generate the estimated age of the 1000 URIs, the docker application is used to call CarbonDate. This was done by *getCreationDateForURI* function - Listing 5, line 6.

2. The *getCreationDates* function in Listing 5 line 24 is responsible for writing the creation date for the 1000 URIs. This function writes the Age into *creationDates.csv* file.

3. To generate a plot for urls with mementos and creation dates, I created a dictionary which matches urls from both files the *MementoCount.csv* and *CreationDate.csv* files - Listing 5, *matchMementoCountWithCreationDates function*.
The result is *agePlotData.csv* files.

Listing 7: Extract agePlotData.csv

```
                                    Links                 MementoCount       Age
1. http://route.overnewser.com/                                48            830
nascarworldnews/?url=http://
www.foxsports.com/nascar/story/n
ascar-monster-energy-to-replace-drivers-names
-on-front-windshield-012717%3fcmpid
=feed:-sports-CQ-RSS-Feed&utm_source=twitter-
tools&utm_medium=NascarWorldNews&utm_campaign=article

2. https://www.twitch.tv/giiggly                                 6              9

3. http://www.jsonline.com/story/sp                            41            748
orts/nba/bucks/2017/01/
29/trumps-ban-sparks-rebuke-bucks-o
fficial/97091812/
?hootPostID=5d701bb5e334d49896aa8678d
ef1d1f1&from=global&sessionKey=&autologin=

4. https://www.washingtonpost.com/sports/r                      3              9
edskins/the-nfl-needs-a-great-super-
bowl-to-rescue-a-snooze-of-a-season/2017/0
1/29/c86677ec-e657-11e6-b82f-687d6e6
a3e7c_story.html?tid=sm_tw_ps

5.http://ktla.com/2017/01/29/7-countries-ta                    49              9
rgeted-in-trumps-executive-order-
initially-identified-as-countries-of-concer
n-under-obama-administration/
```

Listing 8: R code matchMementoCreationDates Graph

```
ageData <-  read.csv('./agePlotData.csv', head=TRUE, sep=",")
plot(ageData$Age, ageData$MementoCount, log='xy', main='Age vs.
 Number of mementos', xlab='Age (days)', ylab='Number of mementos')
```

Total URIs: 1000
Number of Mementos: 261
Number of Date Estimate: 230

References

1. http://docs.tweepy.org/en/v3.5.0/index.html

2. https://github.com/bear/python-twitter

3. https://dev.twitter.com/rest/public

4. http://memgator.cs.odu.edu/timemap/link/http://www.cs.odu.edu/

5. http://memgator.cs.odu.edu/timemap/json/http://www.cs.odu.edu/

6. https://en.wikipedia.org/wiki/Histogram

7. http://ws-dl.blogspot.com/2016/09/2016-09-20-carbon-dating-web-version-30.html