Assignment 2

CS 595: Introduction to Web Science

Spring 2016

Manoj Chandra Kompalli

Finished on February 11,2016

Contents

1 Question 1 **2**
1.1 Answer 3
1.2 Code Listing 3

2 Question 2 **6**
2.1 Answer 6
2.2 Code Listing 7
2.2.1 Code1 8
2.3 Results 8

3 Question 3 **9**
3.1 Answer 10
3.1.1 Results 14

1 Question 1

```
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 many different
shortened URIs for www.cnn.com (t.co, bit.ly, goo.gl, etc.).

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.
```

1.1 Answer

I have started off by searching for APIs for twitter. I have found Twitter search and Tweepy.I chose Tweepy because it came with python and looked easy to implement. I have then created app in twitter to generate keys and tokens for authentication.

I had extracted tweets with keyword news. I converted the response to JSON. I have extracted tweet id and link for the tweet.

I have used expanded url property of the tweet object to get the full url. I have generated all the urls upto 1000 which match the keyword "news" into output.JSON file.

The reason I chose a json file over a text file is , because of the readibility of the file and also because pulling json data is pretty easy .I have used tweet id and link as keys.The links I generated were expanded links . Output.json is a huge file containing 1000 lines. Hence, I pulled out first 19 lines from the Output.json file below.

1.2 Code Listing

links.py

```
import tweepy
import json
import time
import sys
import re
import urllib2
# Authentication Keys to Connect to Twitter API
consumer_key="vjemit5xYQdhgrEPa1FeFf5ZO";
consumer_secret="0
    PoNwIkHk29kUweChIIhGzVD3ZfXwRVqqwxYY3zPadY1BZeNq8";
access_token="3485785534-4FlFNlJtg1uNAlMummglVi9feR7fyvkUS0STp0G
    ";
access_token_secret="
    EpzAYEpf6tHdGFKj43HhnBeAhLNgkyXPdZyH72ec8Ew8d";
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

outputFile = open('output.json', 'w')  #opens the file with
    write permissions

all_urls = set()#for unique data
api = tweepy.API(auth) #Accessing tweepy API
searches = tweepy.Cursor(api.search,q="news").items()#Querying
    for news related tweets
```

```

while True:                                                  # Infinite loop through tweets
    try:
        tweet = searches.next()# iterating through all the matched tweets
        item= tweet._json# converting the tweet object to JSON object
        myitem={}# declaring an empty array to store the details

        tweet_id= item['id_str']
        myitem['tweet_id'] = tweet_id# fetching the id of tweet and storing in JSON object
        #myitem['createdDate'] = created_date
        for link in item['entities']['urls']:
                          all_urls.add(link['url'])

                          myitem['link']=link['expanded_url']# expanded url for full url
                          outputFile.write(json.dumps(myitem) + '\n')# writing JSON data to an output JSON file line by line








        if len(all_urls) == 1000:                           #Checks for 1000 urls in the list and breaks out if more
            break
    except tweepy.TweepError:# catching tweepy error which which occurs frequently enough
        time.sleep(60*10)
        continue
    except StopIteration:
        break
```

Listing 1: Python program for getting 1000 uri's from queried tweets

reduced.json

```
{"tweet_id": "697576352514494466", "link": "http://rol.st/1TSvdBH"}
{"tweet_id": "697576352514494466", "link": "http://twitter.com/RollingStone/status/697575085268463616/photo/1"}
{"tweet_id": "697576352447393792", "link": "http://dailym.ai/1o30Ahg"}
{"tweet_id": "697576352397062144", "link": "http://goo.gl/fb/QnSejs"}
{"tweet_id": "697576352396935168", "link": "http://entabe.jp/news/gourmet/10454/dandelion-chocolate-opens-in-japan"}
{"tweet_id": "697576352271233024", "link": "https://www.londontheatre1.com/news/127554/how-the-other-half-loves-at-the-theatre-royal-haymarket/"}
{"tweet_id": "697576352225062912", "link": "http://rol.st/1TSvdBH"}
{"tweet_id": "697576352225062912", "link": "http://twitter.com/RollingStone/status/697575085268463616/photo/1"}
{"tweet_id": "697576352149405696", "link": "http://bowenpress.com/news/bowen_66161.html"}
{"tweet_id": "697576352099266562", "link": "http://bit.ly/1Q68sEy"}
{"tweet_id": "697576352073953281", "link": "http://www.theguardian.com/australia-news/2016/feb/11/reuters-distances-itself-from-greg-hunt-best-minister-award-it-wasnt-our-idea?CMP=share_btn_tw"}
{"tweet_id": "697576351801307136", "link": "https://gleam.io/3ck3D/gamma-glider-giveaway"}
{"tweet_id": "697576351797108736", "link": "http://dd.hokkaido-np.co.jp/news/area/doto/1-0233547.html"}
{"tweet_id": "697576351486779392", "link": "http://bit.ly/iHrAwards"}
{"tweet_id": "697576351344123904", "link": "http://wpo.st/tqZA1"}
{"tweet_id": "697576351314784257", "link": "http://www.starobserver.com.au/news/local-news/new-south-wales-news/talking-turkey-a-groundbreaking-chat-on-lgbti-parenting/145764"}
{"tweet_id": "697576351042138112", "link": "http://yahoo.jp/g9zlYf"}
{"tweet_id": "697576350756909056", "link": "http://bit.ly/1o03tP4"}
{"tweet_id": "697576350509621248", "link": "http://shrd.by/Atawls"}
```

Listing 2: JSON data of extracted urls from tweets

2 Question 2

```
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://mementoproxy.cs.odu.edu/aggr/timemap/link/1/http://www.cs.odu.edu/

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.

* = https://en.wikipedia.org/wiki/Histogram
```

2.1 Answer

Each time map could have many mementos. First thing,I did was to navigate to the Time map url. Then ,it downloaded a file which gave me the mementos of a single url cs.odu.edu. By using regular expression to locate rel mementos told me if the url had a memento or not. The memcount.py mines for mementos and returns 2 output files which are very useful for the histogram to be plotted next and also for the carbon dating tool. I realized that a file with just an array of memento counts would be sufficient to plot a histogram.The file memcount.py reads the urls from output.json file which was generated in the previous program and writes all mementos of different urls to memcount.json file.It also seperates a list of memento counts and a list of url counts of which have more than 0 mementos and writes the output to two seperate files.This is useful for the carbon dating program. I have found that out of 1000 urls only 33 urls had mementos.

The next part is taking the counts generated and plotting a histogram. I have scaled the y axis to 10 because most of the urls have zero mementos.Some urls have mementos in the range of 0-7000.

They are very less in number and due to this, it is very difficult to represent them in the graph. In the next histogram I have limited the mementos to 600. We can now clearly see the variation in the frequency of urls and mementos.In the last plot, I have introduced breaks to clearly show how many urls have a good number of mementos.

2.2 Code Listing

```
#!/usr/local/bin/python3
import re
import sys
import urllib2
import json

mymementos = re.compile(r'rel.*?=.*?"memento".*?')#use regular
    expressions to find mementos
file3=open('abovezerocounts.json','w')
file4=open('abovezerourls.json','w')

def getTimeMap(url):
    mem_url = "http://mementoproxy.cs.odu.edu/aggr/timemap/link
        /1/" + url #plug in the url to a timemap
    try:
        response = urllib2.urlopen(mem_url)
        timemap = response.read()
    except urllib2.HTTPError:
        timemap = None
    return timemap

def countMementos(mem_url):
        time_map = getTimeMap(mem_url)
        if not time_map:# if no time maps
                count=0
        else:


                count=len(mymementos.findall(str(time_map)))#
                    finds the count of all mementos per url
                if count>0:
                        file3.write("%s\n"% count)#writes the
                            count of urls onto a json file
                        file4.write("%s\n"% time_map)#writes all
                             the urls on to a json file
                #print count
        return count

if __name__=="__main__":
        file1=open('output.json','r')# input a json file that
            contains 1000 urls
        file2=open('memcount.json','w')
        #memcountlist=[]
        for line in file1.readlines():
                one_line = json.loads(line)# loads a json object
                link = one_line['link']
```

```
                    counter=countMementos(link)# counter has count
                        of the urls
                    file2.write("%s"% counter)#outputs count of
                        mementos of each url to a json file
                    file2.write("\r\n")
            #for item in memcountlist:


file1.close()
file2.close()
```

Listing 3: Python program for processing Time Maps for a given file full of links

2.2.1 Code1

```
d = read.table('memcount.json',col.name=c("mementos"))
hist(d$mementos,xlim=c(0,7000),ylim=c(0,10),breaks=500,col=5,
    main="URIs vs Mementos",ylab="No. of URI's",xlab="Mementos")
```

Listing 4: R program for generating the last histogram for Question 2

2.3 Results

Here,we can see that by limiting URI's to 10 and Mementos to 600 ,all the mementos which fall under 600 visible clearly.

The following graph has breaks introduced to make it clear that how many umber of urls those many mementos.Especially in the region of 0-20

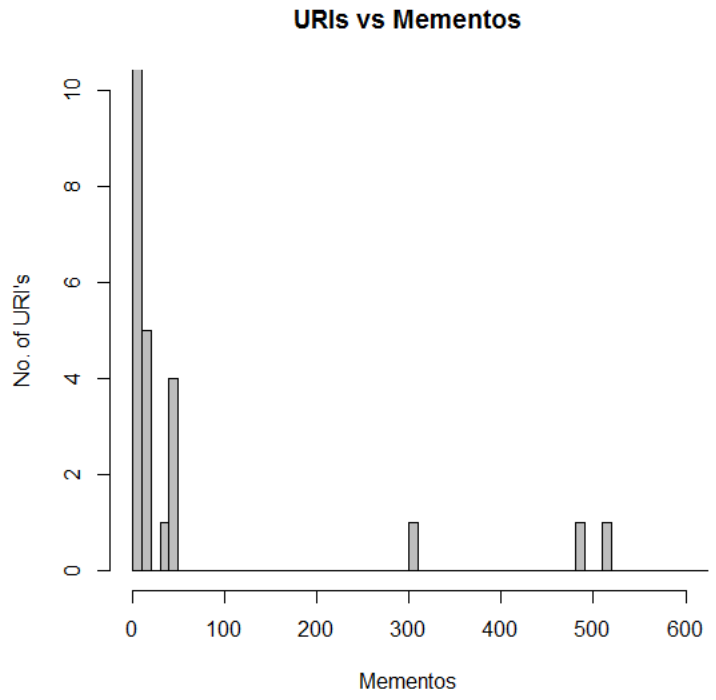

Figure 1: Histogram of URIs vs. number of Mementos for URIs with less than 600 Mementos

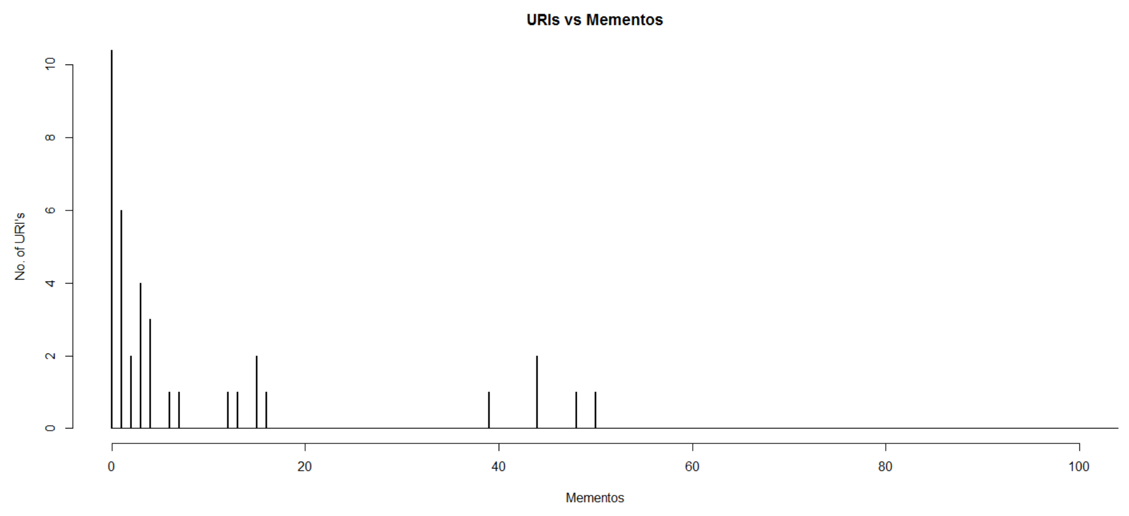

Figure 2: Histogram of URIs vs. number of Mementos for URIs with less than 100 Mementos and breaks inserted between each memento

3 Question 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 to download the tool and install; don't try to use the
web service.

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.
```

3.1 Answer

"Carbon Date" webservice gives the created date,modified date etc of a url .One url at a time.The tool however, can be used to run over a list of urls.

Here we need all the urls which generate atleast one memento.We can get them using memcount.py of problem 2. Our first goal is to find the created date of the urls. File local.py outputs the created dates of all the 33 urls which were having atleast one memento.

We can use these dates and subtract them from the current date which gives the total number of days or Age of each uri.days.py does this job.

Now, we have number of days in one file(numberdays.txt) and number of mementos of each file in another url.We can plot a scatter plot with days on y axis and urls on x axis.The

```
#!/usr/local/bin/python3
import re
import sys
import urllib2
import json

mymementos = re.compile(r'rel.*?=.*?"memento".*?')#use regular
    expressions to find mementos
file3=open('abovezerocounts.json','w')
file4=open('abovezerourls.json','w')

def getTimeMap(url):
    mem_url = "http://mementoproxy.cs.odu.edu/aggr/timemap/link
        /1/" + url #plug in the url to a timemap
    try:
        response = urllib2.urlopen(mem_url)
        timemap = response.read()
    except urllib2.HTTPError:
        timemap = None
    return timemap

def countMementos(mem_url):
        time_map = getTimeMap(mem_url)
        if not time_map:# if no time maps
                count=0
        else:


                count=len(mymementos.findall(str(time_map)))#
                    finds the count of all mementos per url
                if count>0:
                        file3.write("%s\n"% count)#writes the
                            count of urls onto a json file
                        file4.write("%s\n"% time_map)#writes all
                            the urls on to a json file
                #print count
        return count

if __name__=="__main__":
        file1=open('output.json','r')# input a json file that
            contains 1000 urls
        file2=open('memcount.json','w')
        #memcountlist=[]
        for line in file1.readlines():
                one_line = json.loads(line)# loads a json object
                link = one_line['link']
                counter=countMementos(link)# counter has count
                    of the urls
                file2.write("%s"% counter)#outputs count of
                    mementos of each url to a json file
                file2.write("\r\n")
        #for item in memcountlist:


file1.close()
file2.close()
```

Listing 5: Python program for generating count of urls and mementos which have more than zero mementos

```
from checkForModules import checkForModules
import json
from ordereddict import OrderedDict
#import simplejson
import urlparse
import re

from getBitly import getBitlyCreationDate
from getArchives import getArchivesCreationDate
from getGoogle import getGoogleCreationDate
from getBacklinks import *
from getLowest import getLowest

from getLastModified import getLastModifiedDate
#Topsy service is no longer available
#from getTopsyScrapper import getTopsyCreationDate
from htmlMessages import *
from pprint import pprint

from threading import Thread
import Queue
import datetime

import os,sys, traceback




def cd(url, backlinksFlag = False):

    #print 'Getting Creation dates for: ' + url


    #scheme missing?
    parsedUrl = urlparse.urlparse(url)
    if( len(parsedUrl.scheme)<1 ):
        url = 'http://'+url


    threads = []
    outputArray =['','','','','','']
    now0 = datetime.datetime.now()


    lastmodifiedThread = Thread(target=getLastModifiedDate, args
        =(url, outputArray, 0))
    bitlyThread = Thread(target=getBitlyCreationDate, args=(url,
         outputArray, 1))
    googleThread = Thread(target=getGoogleCreationDate, args=(
        url, outputArray, 2))
    archivesThread = Thread(target=getArchivesCreationDate, args
        =(url, outputArray, 3))

    if( backlinksFlag ):
        backlinkThread = Thread(target=
            getBacklinksFirstAppearanceDates, args=(url,
            outputArray, 4))

    #topsyThread = Thread(target=getTopsyCreationDate, args=(url
        , outputArray, 5))


    # Add threads to thread list
    threads.append(lastmodifiedThread)
    threads.append(bitlyThread)
    threads.append(googleThread)
    threads.append(archivesThread)

    if( backlinksFlag ):
        threads.append(backlinkThread)

    #threads.append(topsyThread)


    # Start new Threads
    lastmodifiedThread.start()
    bitlyThread.start()
    googleThread.start()
    archivesThread.start()

    if( backlinksFlag ):
        backlinkThread.start()

    #topsyThread.start()


    # Wait for all threads to complete
    for t in threads:
        t.join()

    # For threads
    lastmodified = outputArray[0]
    bitly = outputArray[1]
    google = outputArray[2]
    archives = outputArray[3]

    if( backlinksFlag ):
        backlink = outputArray[4]
    else:
        backlink = ''




    try:

        lowest = getLowest([lastmodified, bitly, google, archives[0][1], backlink]) #for thread
    except:
       print sys.exc_type, sys.exc_value , sys.exc_traceback




    file2=open('dates.csv','a')
    print lowest
    file2.write("%s\n"% lowest)


```

```
file1=open('abovezerourls.json','r')

for line in file1.readlines():

        cd(line)
```

Listing 6: Python program for extracting the created date of all the urls

```
from datetime import datetime

now=datetime.now()
file1=open('created_dates.txt', 'r')
file2= open('numberdays.txt','w')

for line in file1.readlines() :
        print line
        dateUrl=line.strip().split()
        date = dateUrl[0]
        date_object = datetime.strptime(date,"%Y-%m-%dT%H:%M:%S"
            )
        print date_object
        days = (now - date_object).total_seconds() / ( 3600.0 *
            24 )
        number_days = int(days)
        file2.write("%s\n"% number_days)
        file2.write("\r\n")
# except:
        # date_object = datetime.strptime(line,"%Y-%m-%dT%H:%M:%
            S")
```

Listing 7: Python program for calculating the ages by subtracting the created date from current date

3.1.1 Results

Graph shows an increasing curve .That is, those urls with less mementos have less age and those urls with more mementos have more age,which is the ideal situation.

```
d = read.table('numberdays.txt',col.name=c("number_of_days"))
b = read.table('abovezerocounts.json',col.name=c("memento_counts
    "))
  plot(b$memento_counts,d$number_of_days,xlab="memento counts",
      ylab="days")
```

Listing 8: R program for generating the scatterplot for Question 3

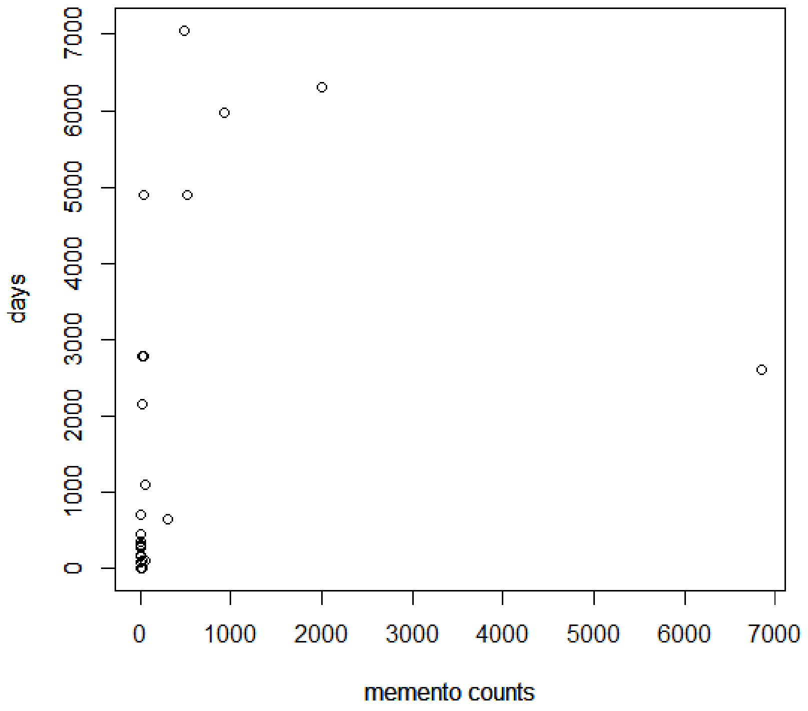

Figure 3: Number of Mementos vs. Days (Age of url)

References

[1] Carbon date service. http://ws-dl.blogspot.com/2014/11/2014-11-14-carbon-dating-web-version-20.html.

[2] Date time function in python. https://docs.python.org/2/library/datetime.htmll.

[3] R histogram. http://www.r-tutor.com/elementary-statistics/quantitative-data/histograml.

[4] R scatterplot. http://www.harding.edu/fmccown/r/.

[5] Tweepy library documentation. http://docs.tweepy.org/en/latest/api.html/.

[6] Writing json data to file:. http://stackoverflow.com/questions/899103/writing-a-list-to-a-file-with-python.