Web Science cs532: Assignment #2

Due on Thursday, February 11, 2016

Dr.Michael.L.Nelson 4:20pm

Zetan Li

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 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.

SOLUTION

To extract urls from twitter, we should get access to twitter API first. After register a twitter application, run the script to get the authentication code.[2]

Listing 1: python script to get authentication code

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
import requests
from requests_oauthlib import OAuth1
from urlparse import parse_qs

REQUEST_TOKEN_URL = "https://api.twitter.com/oauth/request_token"
AUTHORIZE_URL = "https://api.twitter.com/oauth/authorize?oauth_token="
ACCESS_TOKEN_URL = "https://api.twitter.com/oauth/access_token"

CONSUMER_KEY = "YwekEH9UrlXUFKUH2XmImE681"
CONSUMER_SECRET = "NWaB0jN5HaQ3f9VqCrMhP2nP8154KGXoPeDxZk6TIOVptAxErb"

OAUTH_TOKEN = "4860544225-qbjIQvrGlrj493eIHAjNu2OH0rdrHlM94XMHe1x"
OAUTH_TOKEN_SECRET = "vfAc4sJwbWExBjrMZiMqRMuknTzbl2le55DKX5gGYOOXR"


def setup_oauth():
    """Authorize your app via identifier."""
    # Request token
    oauth = OAuth1(CONSUMER_KEY, client_secret=CONSUMER_SECRET)
    r = requests.post(url=REQUEST_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)

    resource_owner_key = credentials.get('oauth_token')[0]
    resource_owner_secret = credentials.get('oauth_token_secret')[0]

    # Authorize
    authorize_url = AUTHORIZE_URL + resource_owner_key
```

```
    print ('Please go here and authorize: ' + authorize_url)

    verifier = raw_input('Please input the verifier: ')
    oauth = OAuth1(CONSUMER_KEY,
                   client_secret=CONSUMER_SECRET,
                   resource_owner_key=resource_owner_key,
                   resource_owner_secret=resource_owner_secret,
                   verifier=verifier)

    # Finally, Obtain the Access Token
    r = requests.post(url=ACCESS_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)
    token = credentials.get('oauth_token')[0]
    secret = credentials.get('oauth_token_secret')[0]

    return token, secret


def get_oauth():
    oauth = OAuth1(CONSUMER_KEY,
                client_secret=CONSUMER_SECRET,
                resource_owner_key=OAUTH_TOKEN,
                resource_owner_secret=OAUTH_TOKEN_SECRET)
    return oauth

if __name__ == "__main__":
    if not OAUTH_TOKEN:
        token, secret = setup_oauth()
        print ("OAUTH_TOKEN: " + token)
        print ("OAUTH_TOKEN_SECRET: " + secret)
        print
    else:
        oauth = get_oauth()
        r = requests.get(url="https://api.twitter.com/1.1/statuses/
            mentions_timeline.json", auth=oauth)
        print (r.json())
```

After getting access code, we can use that to get urls from twitter.
To achieve this ,we want to use stream listener to get the tweet from twitter and extract link in them, this is the faster way than search method.[1]
(the sys.out line is to print work progress on the screen)

Listing 2: python script to extract urls from tweets

```
from twitter import *
import sys

CONSUMER_KEY = "YwekEH9UrlXUFKUH2XmImE681"
CONSUMER_SECRET = "NWaB0jN5HaQ3f9VqCrMhP2nP8154KGXoPeDxZk6TIOVptAxErb"

OAUTH_TOKEN = "4860544225-qbjIQvrGlrj493eIHAjNu2OH0rdrHlM94XMHe1x"
OAUTH_TOKEN_SECRET = "vfAc4sJwbWExBjrMZiMqRMuknTzbl2le55DKX5gGYOOXR"

auth = OAuth(OAUTH_TOKEN,OAUTH_TOKEN_SECRET,CONSUMER_KEY,CONSUMER_SECRET)
stream = TwitterStream(auth = auth, secure = True)

tweet_iter = stream.statuses.filter(track='game,sports,movie,comic,PUA,fitness')
url_counter=0
urlfile=open("url_raw.txt","w")
for tweet in tweet_iter:
     if url_counter>=10000:
          break;
     if "entities" in tweet:
          if "urls" in tweet["entities"]:
               for url in tweet["entities"]["urls"] :
                    urlfile.write(url["expanded_url"]+"\n")
                    url_counter=url_counter+1
                    if url_counter>=10000:
                         break
                    sys.stdout.write("\r"+str(url_counter/100.00)+"%")
                    sys.stdout.flush()
urlfile.close()
```

Note that not all tweets have url in them and many urls we get are duplicates to each other. So the tweets without ulrs are ought to be dropped. And we should extract more than 1000 urls for there are many duplicated links. Here, 10k raw links are extracted for later use.

Next step is to extract 1000 unique valid links from raw data. In python **set** can fulfill that need. For url validating, **requests** module is a nice tool for that, the link that received 200 response or the final redirected link with 200 response is left in the list.

Listing 3: python script to get unique&valid urls

```
import requests
import sys
def getRealUrl(url):
    try:
        r=requests.get(url)
        if r.status_code==200:
            return r.url
    except Exception, e:
        return None

rawFile=open("url_raw.txt")
lines=rawFile.readlines()
list_url=set()
print('loading raw data...')
for line in lines:
    url=getRealUrl(line)
    if url is not None:
        list_url.add(url)
        l=len(list_url)
        sys.stdout.write("\r"+str(l/10.0)+"%")
        sys.stdout.flush()
        if l >=1000:
            break
rawFile.close()
print('load finished, start writing file...')
newFile=open('url_list.txt','w')
for r_url in list_url:
    newFile.write(r_url+"\n")
newFile.close()
```

Problem 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.

SOLUTION

Instead of use python, shell script is used for this problem. With shell, we can access curl's output directly. Then filter out memento links and check time map links by pipeline the out put into grep.

Listing 4: Shell script for counting memento and recursively check time map

```
#!/bin/bash
rm -f momento.list
touch momento.list
prefix="http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/"
counter=0
num=0
rm -rf tmpdir
mkdir tmpdir
function getUrlCount()
{
     local fid=`date +%s`
     local curNum
     local ts
     local nurl
     curl --silent "$1" > tmpdir/tmp$fid
     curNum=`cat tmpdir/tmp$fid | egrep 'rel="[^"]*memento[^"]*"' | wc -l`
     num=$[$curNum+$num]
     if [[ $curNum -gt 0 ]]; then
          for ts in `cat tmpdir/tmp$fid | grep -o '<[^>]*>;rel="timemap"'`; do
               nurl=`echo -n $ts |  awk -F"[<>]" '{print $2}'`
               if [[ "$nurl" != '' ]]; then
                    getUrlCount "$nurl"
               fi
          done
     fi
}

for line in `cat url_list.txt`; do
     curl --silent "${prefix}${line}"> tmpdir/tmp
     num=`cat tmpdir/tmp | egrep 'rel="[^"]*memento[^"]*"' | wc -l`
     if [[ $num -gt 0 ]]; then
          for timemap in `cat tmpdir/tmp | grep -o '<[^>]*>;rel="timemap"'`; do
               newurl=`echo -n $timemap |  awk -F"[<>]" '{print $2}'`
               if [[ "$newurl" != '' ]]; then
                    getUrlCount "$newurl"
               fi
          done
```

```
    fi
    echo -e $line"\t"$num >>momento.list
    counter=$[$counter+1]
    percent=$[$counter/10]
    echo -e -n "\r"$percent"%"
done
rm -rf tmpdir
```

Once we get the data, it is written in "url memento" format. The symbols in url will trigger error in data importing in R. So a script is written to keep only memento numbers in the data list.

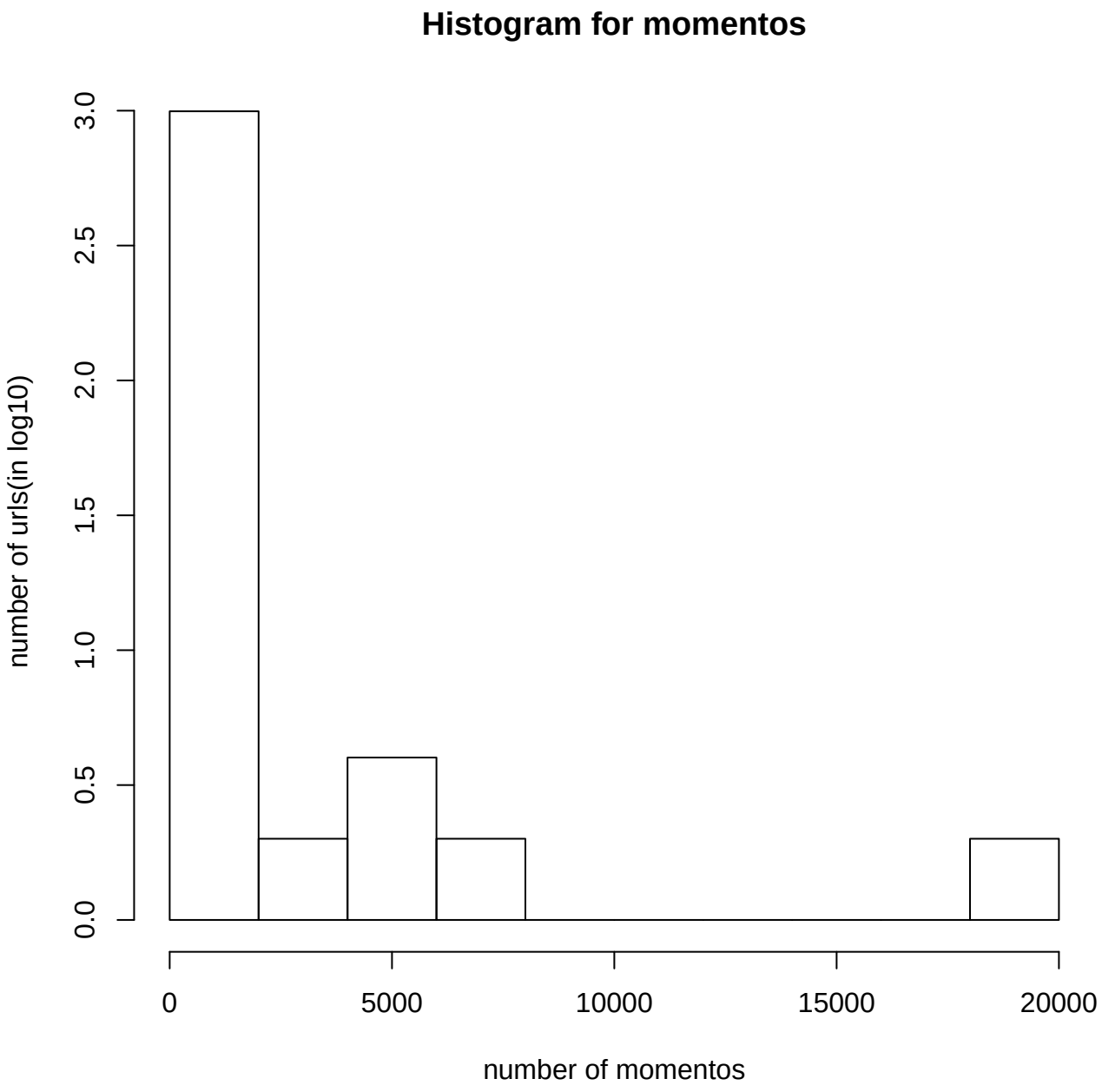
Listing 5: Convert the data to safe format for R

```
#!/bin/bash
cut -f2 momento.list > momento_R.list
```

Below is the R code to plot histogram of the data. Since the url with zero memento is far more than others, y axis value is scaled by log10.
We add 1 to each y value to avoid log10(0) calculation, for this will result in -infinity and make a hole in the graph.

Listing 6: R script for histogram

```
pdf("momento_plot.pdf")
mydata=read.table("momento_R.list",sep="\t",header = FALSE)
h = hist(mydata$V1, plot = F
    )
h$counts=log10(h$counts+1)
plot(h,ylim = c(0,log10(900)), main = "Histogram for momentos",
    xlab = "number of momentos", ylab = "number of urls(in log10)")
dev.off()
```


Histogram for momentos
number of urls(in log10)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
5000
10000
15000
20000
number of momentos

Problem 3

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

http://ws-dl.blogspot.com/2014/11/2014-11-14-carbon-dating-web-version-20.html

Note: you'll should download the library and run it locally; 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.

Not all URIs will have Mementos, and not all URIs will have an estimated creation date. State how many fall into either categories.

SOLUTION

Since we have data files in problem 2. We want to push these url with mementos greater than 0 into carbon date tool and filter out those have no estimated birth date to get the final data.

Below is the code to get the carbon date of urls that have memento numbers greater than 0. We import the local.py as module, and re-parsing the string formatted result to a json object. Then we can retrieve the "Estimated carbon date" value out of the object.

Note that this script must be run under carbon date tool directory.

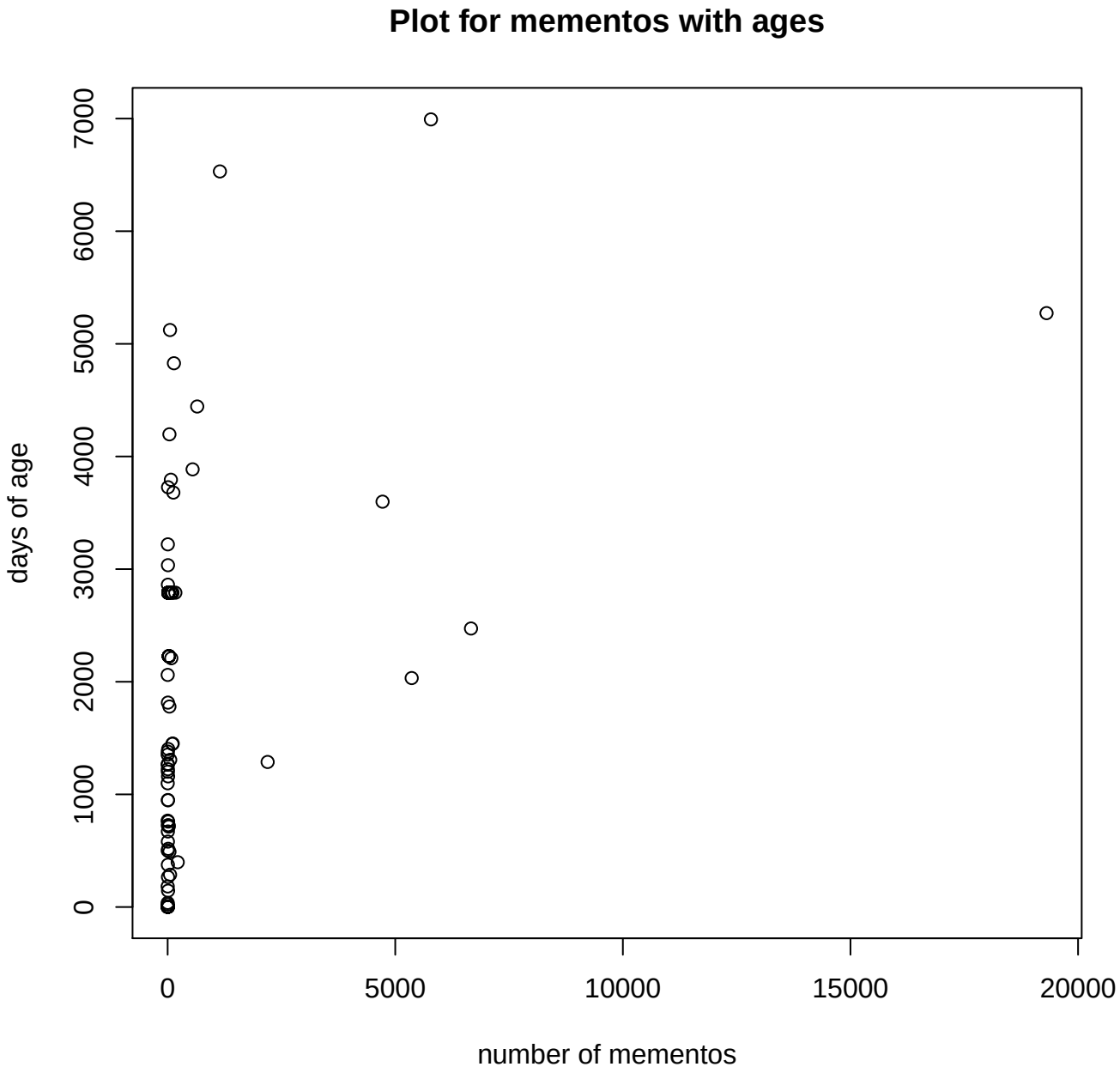
Listing 7: Python code to get carbon date

```
from local import cd
import json
import datetime
time_now=datetime.datetime.now()
data=open("../../momento.list")
outputf=open('../../cd.list','w')
strline=data.readlines()
counter=0
for line in strline:
    col=line.split('\t')
    memento=int(col[1])
    if memento>0:
        r=cd(col[0])
        re=json.loads(r)
        if re['Estimated Creation Date'] != '':
            createTime=datetime.datetime.strptime(re['Estimated Creation Date'
                ],'%Y-%m-%dT%H:%M:%S')
            deltaTime=time_now-createTime
            outputf.write(str(deltaTime.days)+"\t"+str(col[1]).strip()+"\n")
    counter=counter+1
    print(str(counter/10.0)+"%")
data.close()
outputf.close()
```

Then plot these data into scatter graph.

Listing 8: R script to plot scatter graph

```
pdf("age_plot.pdf")
mydata=read.table("cd.list",sep="\t",header = FALSE)
plot(mydata$V2,mydata$V1, main = "Plot for mementos with ages",
     xlab = "number of mementos", ylab = "days of age")
dev.off()
```



References

[1] ideoforms. *twitter-stream-extract-links.py*, 2015 (accessed Febrary 9, 2016).

[2] Thomas Sileo. *Using Twitter REST API v1.1 with Python*, 2013 (accessed Febrary 8, 2016).