CS 851: Assignment #1

Due on Thursday, February 12, 2015

Dr Nelson 4:20pm

Victor Nwala

Contents

Problem 1

Write a program that extracts 10000 tweets with links from Twitter.

Listing 1: Code to extract tweets from Twitter and save URIs also save the tweet mappings and collects the creation date of each tweet

```
import tweepy
import time
from datetime import datetime
import datetime

access_token = "384946837-aPnqh9DAtOKljCShMepwPJVg27dROGGysYuy9xog"
access_token_secret = "ow458SMzbIcAVZ3RL2nypCGYuqmkaoHTNlbZCVBiHG6FC"
consumer_key = "c3SExFZ3K6Do6Yw2Kwi84Strl"
consumer_secret = "CXobLgtdn8feYInLs659BxsjnBTCgfpmD5eEyENi1Bu6ttbfau"



#today = utc_datetime.strftime("%Y-%m-%d %H:%M:%S")


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
count = 0
req = 0
while count < 10000:

    query = 'http%3A%2F%2Fwww%2E-filter:link'
    for tweet in tweepy.Cursor(api.search, q=query).items(30):
            #utc_datetime = datetime.datetime.utcnow()
            date_str = str(tweet.created_at)
            #dt_obj = datetime.datetime.strptime(date_str, "%Y-%m-%d %H:%M:%S")
            #Age  = utc_datetime - dt_obj
            for s in tweet.entities['urls']:
                print s['url']
                saveFile = open('urls3.txt','a')
                url = s['url']
                saveFile.write(url)
                saveFile.write('\n')
                saveFile.close()
                saveFile = open('tweetAge3.txt','a')
                saveFile.write(date_str)
                saveFile.write('\n')
                saveFile.close()
                count = count + 1
            length =  len(tweet.entities['urls'])
            if length > 0:
                saveFile = open('length3.txt','a')
                saveFile.write(str(length))
                saveFile.write('\n')
                saveFile.close()
            req = req + 1
```

Problem 1 continued on next page. . .

```
            if req == 50:
                time.sleep(15)
                req = 0
```

The mappings are as follows: Out of 9530 tweets and 10027 URIs, 9097 tweets have 1 link, 380 tweets have 2 links, 50 tweets have 3 links, 3 tweets have 5 links.

For each t.co link, use curl I L to record the HTTP headers all the way to a terminal HTTP status (i.e. chase down all the redirects) How many unique final URIs? How many duplicate URIs? Build a histogram of how many redirects (every URI will have at least 1)

Listing 2: Code to extract status codes of URIs and count the number of redirects of each URI

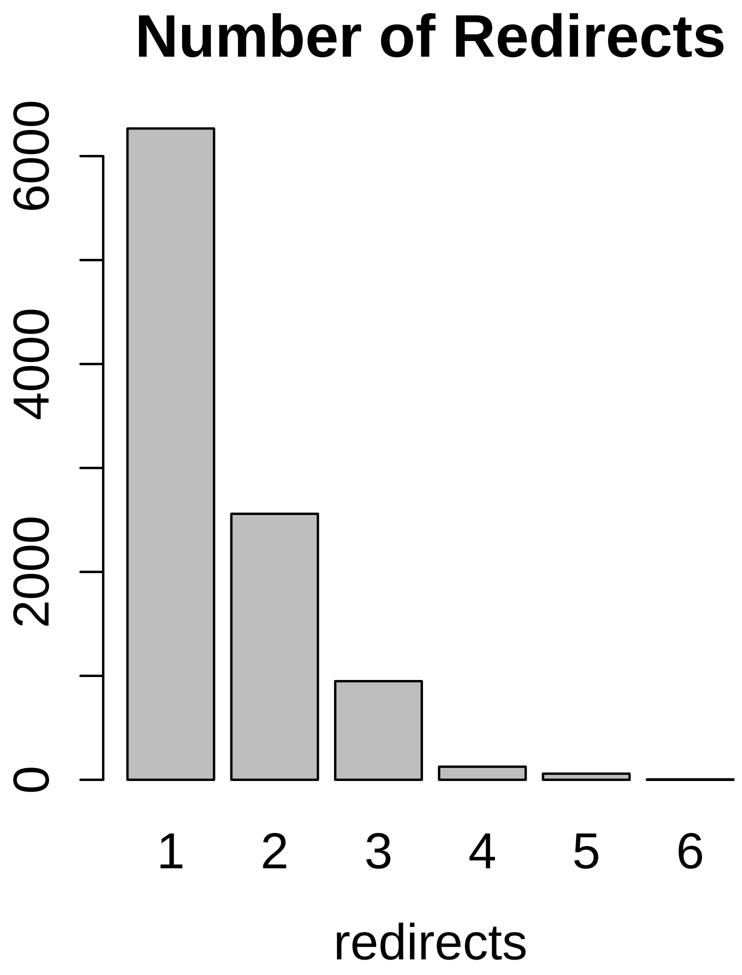
```
# -*- coding: utf-8 -*-
import requests
import urllib2
import urllib
from urlparse import urlparse
import subprocess
import os, sys
import httplib
import re



#saveFile = open("code.txt",'a')

fh = open("urls3.txt",'r')

for line in fh:
     count = 0
     try:
          url=line
          word = 'HTTP/1.'
          proc = subprocess.Popen(["curl $1 -s -L -I "  + url ], stdout=subprocess
              .PIPE, shell=True)
          (out, err) = proc.communicate()
          index = 0
          while index < len(out):
                  index = out.find(word, index)
                  if index == -1:
                      break
                  end = index + 13
                  res = out[index:end]
                  res1 = res.split()
                  #print res1[1]
                  if ( res1[1] == '301' or  res1[1] == '302' or  res1[1] == '303'
                      or  res1[1] == '307'):
                    count = count + 1
                  else:
                    #count = 0
                  #if count != 0:
                    print count
                    if count != 0:
                         saveFile1 = open("NewCount.txt",'a')
                         saveFile1.write(str(count) + '\n')
                         saveFile1.close()
                  #saveFile.write(str(res1[1]) + '\n')
                 #saveFile.write()
                  index +=7
     except BaseException, e:
          print 'failed ',str(e)
```

Listing 3: Code to collect final URIs of each link

```
import requests
import urllib2
from urlparse import urlparse
fh = open("urls3.txt",'r')
saveFile = open("NewUrl.txt",'a')
for line in fh:
     url=line


     try:
          def get_redirected_url(url):

              opener = urllib2.build_opener(urllib2.HTTPRedirectHandler)
              request = opener.open(url)
              return request.url
          k = get_redirected_url (url)


          print k
          saveFile.write(k)
          saveFile.write('\n')


     except BaseException, e:
          saveFile.write('0')
          saveFile.write('\n')
saveFile.close()
fh.close()
```

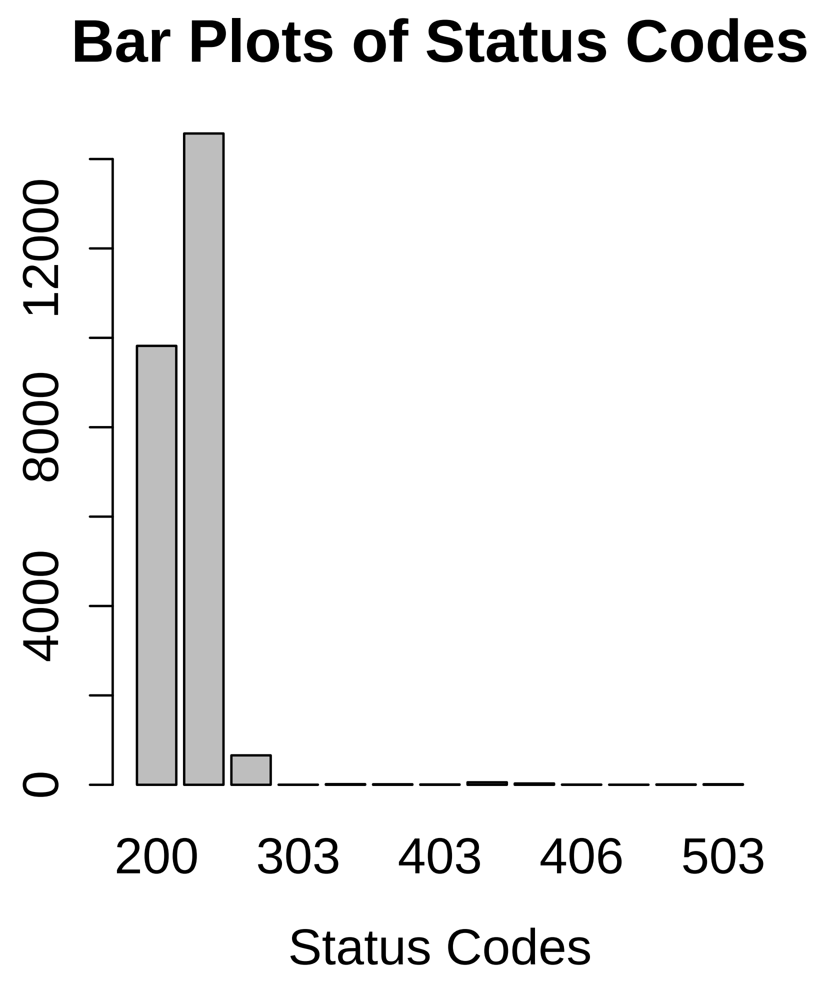
After running both python programs, I had 9850 final URIs of which 769 were unique and 9081 where duplicates.

Figure 1: Number of redirects from 10027 URIs



Problem 1 continued on next page. . .

Figure 2: HTTP Status Codes



Problem 2

Use Carbon Date to estimate the age of each link(s) in a tweet. Create a histogram of (Agetweet minus Agelink). Many (most?) deltas will be 0, but there should be many greater than 0. For these deltas, compute: median, mean, std dev, std err. Use wget to download the text for all the links. Hold on to those, well come back to them later.

Listing 4: Code to carbon-date each URI

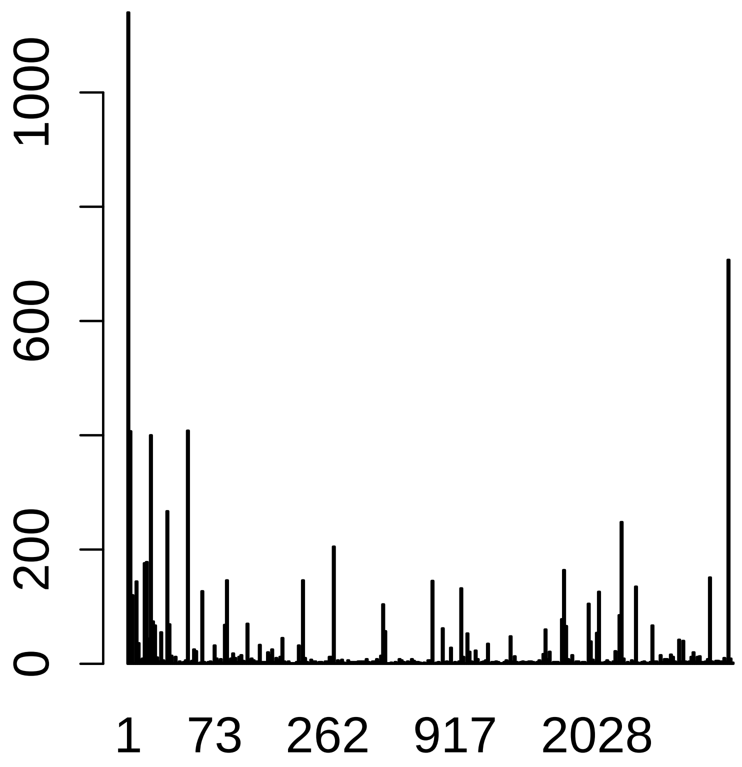
```
from checkForModules import checkForModules
import json
from ordereddict import OrderedDict
#import simplejson
import urlparse
import re

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

from getLastModified import getLastModifiedDate
from getTopsyScrapper import getTopsyCreationDate
from htmlMessages import *
from pprint import pprint

from threading import Thread
import Queue
import datetime

import os,sys, traceback




fh = open("new.txt",'r')

for line in fh:
    url=line
    url=url.replace('\n','')



    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 = ''

        topsy = outputArray[5]

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



        result = []

        #result.append(("URI", url))
        result.append(("Estimated Creation Date", lowest))
        #esult.append(("Last Modified", lastmodified))
        #result.append(("Bitly.com", bitly))
        #result.append(("Topsy.com", topsy))
        #result.append(("Backlinks", backlink))
        #result.append(("Google.com", google))
        #result.append(("Archives", archives))
        values = OrderedDict(result)
        r = json.dumps(values, sort_keys=False, indent=2, separators=(',', ': '))

        now1 = datetime.datetime.now() - now0


        #print "runtime in seconds: "
        #print now1.seconds
        #print r
        print 'runtime in seconds:  ' +  str(now1.seconds) + '\n' + r + '\n'
        k = str(now1.seconds) + '\n' + r

        i =lowest
        print i
        saveFile = open("carbonDate.txt",'a')
        saveFile.write(i)
        saveFile.write('\n')
        saveFile.close()
        return r
    cd(url)




#if len(sys.argv) == 1:
    #print "Usage: ", sys.argv[0] + " url backlinksOnOffFlag ( e.g: " + sys.argv
        [0] + " http://www.cs.odu.edu  [--compute-backlinks] )"
#elif len(sys.argv) == 2:
    #fix for none-thread safe strptime
    #If time.strptime is used before starting the threads, then no exception is
        raised (the issue may thus come from strptime.py not being imported in a
        thread safe manner). -- http://bugs.python.org/issue7980
    #time.strptime("1995-01-01T12:00:00", '%Y-%m-%dT%H:%M:%S')
    #cd(sys.argv[1])
#elif len(sys.argv) == 3:
    #time.strptime("1995-01-01T12:00:00", '%Y-%m-%dT%H:%M:%S')

    #if(sys.argv[2] == '--compute-backlinks'):
       # cd(sys.argv[1], True)
    # else:
        #cd(sys.argv[1])
```

Listing 5: Code to calculate Agetweet minus Agelink each tweet and link respectively

```
import time
import datetime
import calendar

fh = open("carbonDate.txt",'r')

date1 = '2015-02-04T00:00:00'
date1 = date1.split(" ")
date1[-1] = date1[-1][:18]
date1 = " ".join(date1)
epoch1 = int(calendar.timegm(time.strptime(date1, '%Y-%m-%dT%H:%M:%S')))
print  epoch1

for line in fh:
     date=line
     try:
          date = date.split(" ")
          date[-1] = date[-1][:18]
          date = " ".join(date)
          epoch = int(calendar.timegm(time.strptime(date, '%Y-%m-%dT%H:%M:%S')))

          t2 =epoch1 - epoch
          day = (t2/86400)
          day = abs(day)
          print day
          if day > 0:
               saveFile = open("day.txt",'a')
               saveFile.write(str(day))
               saveFile.write('\n')
               saveFile.close()
     except BaseException, e:
          print e


fh.close()
```

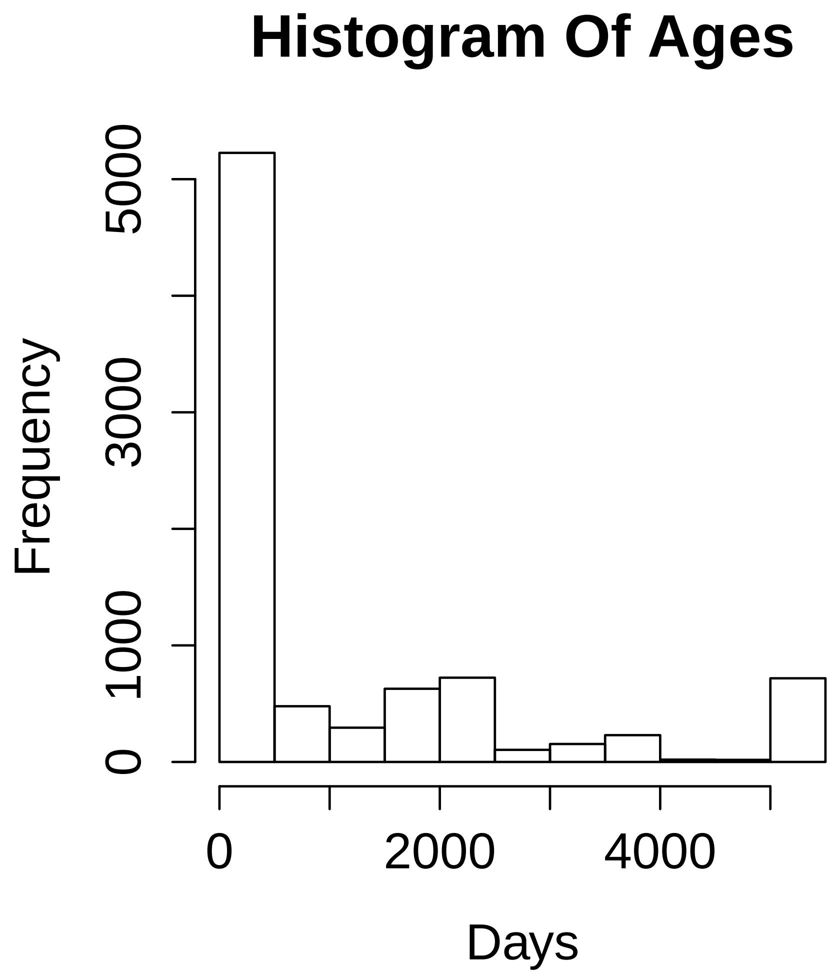
Mean = 1084, Median = 118, Standard Deviation = 1587.08, Standard devition error = 17.12089.
I used this (function(x) sd(x)/sqrt(length(x))) function in Rstudio to calculate the Standard devition error.

Figure 3:Bar Plot of Age Difference



Problem 2 continued on next page. . .

Figure 4: Histogram of Age Difference



Listing 6: Code to download pages with wget.

```
import requests
import urllib2
import urllib
from urlparse import urlparse
import subprocess
import os, sys
import httplib
import re

fh = open("NewUrl.txt",'r')
count = 0
for line in fh:
     try:
          url=line

          proc = subprocess.Popen(["wget -e robots=off -P ./wgetFiles/ -p -k  " +
               url ], stdout=subprocess.PIPE, shell=True)
```

Problem 2 continued on next page. . .

```
        (out, err) = proc.communicate()
    except BaseException, e:
        print 'failed ',str(e)
```

CONCLUSION

I observed that the method I used to collect the links from twitter had too many duplicates. Hence my procedures are correct but my results are not my desired result. I made this discovery deep into the deadline for submission, hence I could not re-do the whole process. I have downloaded more URLs to use at a latter stage i.e. if non-duplicates are required at a latter state in this course. I used only one date as the created date for tweets because I descovered they all had the same created date but at different times in that same day.
Finally, initially to collect my tweets I used streamListener, I noticed I could not make it fast, I spoke to Alexander (a fellow classmate) and he suggested API search, I believe streamListener does a better job collecting unique items than API search, I just had to find out late. I should also state I re-used some of the old code I had written in the web science course.