CS 751: Assignment 1

Write a program that extracts 10000 tweets with links from Twitter.
Note that only Twitter API 1.1 is currently available; version 1 code will no longer work.
Reference:
http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/

Rohit Lambi

Spring 2015

Contents

1 Q1

1. Save the tweet URIs, and the mapping to the link(s) each tweet contains - Note: tweets can have more than 1 links
2. For each t.co link, record the HTTP headers all the way to a terminal HTTP status (i.e. chase down all the redirects)
3. How many unique final URIs? How many duplicate URIs?
4. Build a histogram of how many redirects (every URI will have at least 1) http://en.wikipedia.org/wiki/Histogram
5. Build a histogram of HTTP status codes encountered (you will have at least 20000: 10000 301s, and 10000+ more)

1.1 Solution

This assignment has been implemented in Python. Following is the detailed report of how this assignment has been done:

1. To be able to access Twitter's REST API we need the secret keys. I obtained these by registering an app on Twitter's website from my Twitter account.
2. Instead of directly using the Twitter's REST API, I used 'Tweepy' library which is a wrapper to the Twitter's REST API. This makes coding a bit easier.
3. Using the secret keys, get the authentication token from Twitter and use it in the next webservice calls.
4. Twitter has a limitation on the number of times we can access their API within a certain period of time which is 15 minutes. So, to fetch the tweets, I continuously made the webservice calls to fetch the tweets and once the limit got exhausted, I made my program to wait for 15 minutes before making request for the next set of tweets.
5. I saved the Tweet data in a text file in JSON format. The stored information was Tweet id, Tweet text, URLs, Tweet created date
6. To get the final URLs of t.co URLs from Twitter, I used 'Requests' library of Python.
7. For each t.co URL which was stored in file in the previous step, I made a HTTP HEAD request by setting the 'allow redirects' flag to 'True'.
8. From the response, I stored all the HTTP status codes in one text file and in other text file I stored the final URL and number of redirects required to reach the final URI.
9. Moreover, I calculated and stored the count of unique and duplicate URIs in the third text file.

1.2 Number of URIs Encountered:

uniqueUrls: 5971
duplicateUrls: 4306

1.3 Code Listing

tweetFetcher.py

```
'''
Created on Jan 31, 2015

@author: rlambi
'''
import tweepy
import time
import json
from datetime import datetime

CONSUMER_KEY = "iuKUndPfIF5aWnNl0Ayq9Ztgt"
CONSUMER_SECRET = "QuNsF4gL2LssmbcdtKpyLZGiQctz98T4hXWcAKrBYGh72ZTFC8"

OAUTH_TOKEN = "549294315-P89swbZzgiP2n9bq6fW2T2jm5etru6Wr6TN08Lg3"
OAUTH_TOKEN_SECRET = "NMzDaS5doFtHxXxebE68AunmRHTsFfLxwAkk3LsDN75JH"

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)

api = tweepy.API(auth)

fil = open('tweets.txt', 'w')
log = open('log.txt', 'w')
data = {}
url_list = []
urls = []

seqNum = 0
log.write('Start: ' + str(0) + datetime.now().time().__str__() + '\n')

myCursor = tweepy.Cursor(api.search, q="http:links", rpp=100).items()

while True:
    try:
        tweet = myCursor.next()

        url_list = tweet._json['entities']['urls']

        urls[:] = []
        if url_list.__len__() > 0:
            seqNum += 1

            print seqNum, " ", url_list

            for url in url_list:
                urls.append(url['url'])

            data['id_str'] = tweet._json['id_str']
            data['text'] = tweet._json['text']
            data['created_at'] = tweet._json['created_at']
            data['urls'] = urls
            data['seqNum'] = seqNum

            json_tweet = json.dumps(data)
            fil.write(json_tweet + '\n')
    except tweepy.TweepError:

        tmp = 'Sleep at ' + str(seqNum) + ' ' + datetime.now().time().__str__()  + '\n'
        print tmp
        log.write(tmp)
        time.sleep(900)

```

```
63            tmp = 'Wakeup: ' + datetime.now().time().__str__() + '\n'
64            print tmp
65            log.write(tmp)
66            continue
67
68        if seqNum == 10000:
69            break
70
71
72    print "Total tweets: ", seqNum
73    log.write('Stop: ' + datetime.now().time().__str__() + '\n')
74    log.write("Total tweets: " + str(seqNum))
75
76    log.close()
77    fil.close()
```

Listing 1: Python program to fetch 10,000 Tweets with links

fetchFinalURI.py

```
'''
Created on Feb 5, 2015

@author: rlambi
'''
import requests
import requests.exceptions
import json
import csv
import os

filePath = '/home/rlambi/rohit/Courses/Digital-Libraries/Assignments/'
outputDir = filePath + 'output_2/'

if not os.path.exists(outputDir):
    os.makedirs(outputDir)

finalURIs = []
duplicateUrlsCount = 0
uniqueUrlsCount = 0
skippedUrlsCount = 0
serialNo = 1
fileTweets = open(filePath + 'tweets.txt', 'r')
fileHttpStatuses = open(outputDir + 'http-statuses.txt', 'w')
fileUrls = open(outputDir + 'tweets-processed-1.txt', 'w')
skippedUrls = open(outputDir + 'skippedUrls.txt', 'w')

xx = 1
for line in fileTweets:

    tweet = json.loads(line)

    tweetedOn = tweet['created_at']
    urlList = tweet['urls']

    for url in urlList:

        resp = requests.Response()
        try:
            resp = requests.head(url, allow_redirects=True, timeout=10)

            redirectCount = len(resp.history)
            terminalURI = resp.url
            print serialNo, '\t', tweet['seqNum'], '\t' + url + '\t' + terminalURI + '\t' +
                tweetedOn + '\t' + str(redirectCount)
            fileUrls.write(str(serialNo) + '\t' + str(tweet['seqNum']) + '\t' + url + '\t' +
                 terminalURI + '\t' + tweetedOn + '\t' + str(redirectCount) + '\n')
            resp.history.append(resp)

            # Check in HttpStatusMap if this statusCode is already present and add/update
                its count accordingly
            for respHist in resp.history:
                fileHttpStatuses.write(str(respHist.status_code) + '\n')

            if finalURIs.count(terminalURI) == 0:
                finalURIs.append(terminalURI)
            else:
                duplicateUrlsCount += 1

            serialNo += 1

        except requests.exceptions.Timeout:
            print 'Timeout ' + str(tweet['seqNum']) + '\t' + url
            skippedUrls.write(str(tweet['seqNum']) + '\t' + url)
            skippedUrlsCount += 1
        except:
            print 'Skipped ' + str(tweet['seqNum']) + '\t' + url
            skippedUrls.write('Skipped ' + str(tweet['seqNum']) + '\t' + url)
            skippedUrlsCount += 1

```

```
#       if xx == 50:
#           break
#       xx += 1



uniqueUrlsCount = len(finalURIs)
print 'uniqueUrls: ', uniqueUrlsCount
print 'duplicateUrls: ', duplicateUrlsCount
print 'skippedUrls: ', skippedUrlsCount


fileUrlCount = open(filePath + 'url-count.txt', 'w')
fileUrlCount.write('uniqueUrls:', '\t', str(uniqueUrlsCount))
fileUrlCount.write('duplicateUrls:', '\t', str(duplicateUrlsCount))
fileUrlCount.close()

skippedUrls.close()
fileUrls.close()
fileTweets.close()
fileHttpStatuses.close()
```

Listing 2: Python program to fetch the final URI for each of the t.co URIs from the Listing 1

Histograms

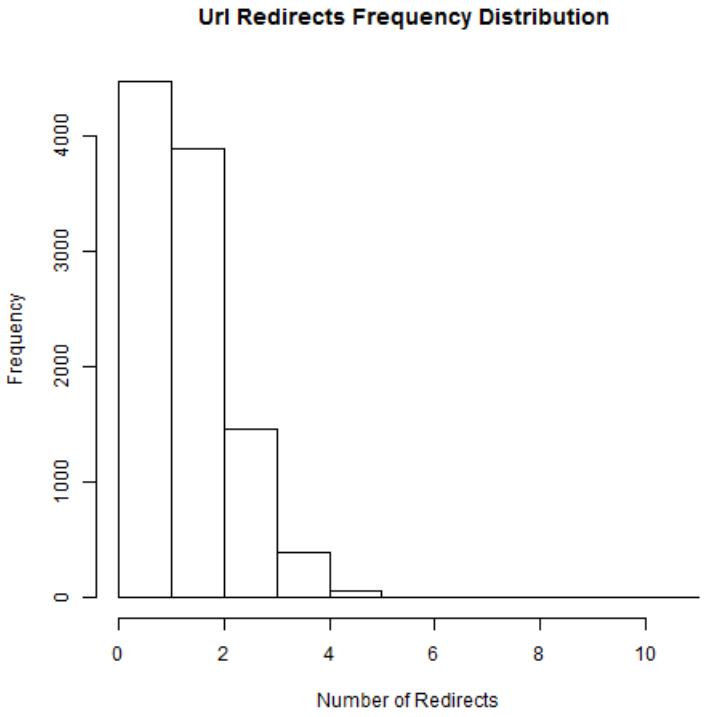

Figure 1: Number of Redirects

url-redirect-histogram.R

```
1 redirects=read.csv("tweets-processed-1.txt",stringsAsFactors=F,header=FALSE,sep="\t")
2 redirects
3 data=redirects[,1]
4 png("url-redirect-histogram.png")
5 hist(data,main="Url Redirects Frequency Distribution ",freq=T,xlab="Number of Redirects",
      ylab="Frequency")
6 dev.off()
```

Listing 3: R program to plot histogram for the number of redirects encountered

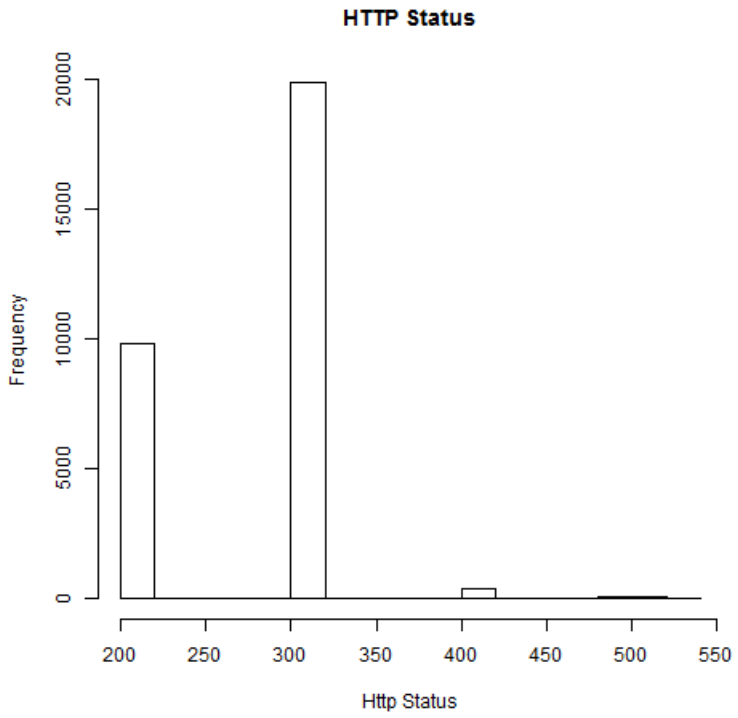

Figure 2: HTTP statuses encountered

http-statuses.R

```
1  inputFile=read.csv("http-statuses.txt",stringsAsFactors=F,header=FALSE,sep="\t")
2  inputFile
3  data=inputFile[,1]
4  png("http-statuses.png")
5  hist(data,main="HTTP Status",freq=T,xlab="Http Status",ylab="Frequency")
6  #barData <- table(data)
7  #png("http-statuses-bar.png")
8  #barplot(barData, main="HTTP Status", xlab="Http Status",ylab="Frequency")
9  dev.off()
```

Listing 4: R program to plot histogram for the HTTP statuses encountered

2 Q2

1. Use Carbon Date to estimate the age of each link(s) in a tweet - See: http://ws-dl.blogspot.com/2013/04/2013-04-19-carbon-dating-web.html
2. Create a histogram of (Agetweet Agelink) - Many (most?) deltas will be 0, but there should be many more than 0
3. For these deltas, compute: median, mean, std dev, std err
4. Use wget to download the text for all the links. Hold on to those, we will come back to them later.

2.1 Solution

1. For Carbon-Dating, I downloaded its source code from github, installed the pre-requisites.
2. I registered an app on Bitly.com and updated the access token in Carbon-Date's 'config' file and made the other necessary configuration changes.
3. I started Carbon-Date server locally and wrote a Python script that fetches the Estimated URI created date for each URI stored in the previous step. At the same time, I calculated the difference between this date and the date on which this URI was Tweeted.
4. Output of this script is a text file which contains URI, Tweet created date(this information is carry forwarded from the previous step), Estimated URI created date and delta between the two dates.
5. Lastly, I created a python program to fetch the text for all the links.

2.2 Code Listing

URIAge.py

```
'''
Created on Feb 7, 2015

@author: rlambi
'''
import requests
import json
import csv
import os
import datetime


def skip(row):
    print 'Skipped ' + '\t' + row['serialNo']  + '\t' + row['tweetSeqNo'] + '\t' + row['tcoURL'] + '\t' +  row['finalURL'] + '\t' +  row['tweetedOn'] + '\n'
    errorLogFile.write('Skipped ' + '\t' + row['serialNo']  + '\t' + row['tweetSeqNo'] + '\t' + row['tcoURL'] + '\t' +  row['finalURL'] + '\t' +  row['tweetedOn'] + '\n')

serverURL = 'http://localhost:8080/cd?url='

tweetAgeMap = {}
filePath = '/home/rlambi/rohit/Courses/Digital-Libraries/Assignments/'
inputFilePath = filePath + 'output_2/tweets-processed-1.txt'
outputDir = filePath + 'output_3/'
outputFilePath = outputDir + 'url-dates.csv'
xx = 1
errorLogFile = open(outputDir + 'errorLog.txt', 'w')

if not os.path.exists(outputDir):
    os.makedirs(outputDir)

with open(inputFilePath) as inputFile:
        reader = csv.DictReader(inputFile, delimiter='\t', fieldnames=['serialNo', 'tweetSeqNo', 'tcoURL', 'finalURL', 'tweetedOn', 'redirectCount'])

        with open(outputFilePath, 'wb') as outputFile:
            writer = csv.writer(outputFile, delimiter='\t')

            try:
                for row in reader:
                    print 'Processing ', row['serialNo'], row['finalURL']
                    resp = requests.get(serverURL + row['finalURL'])
                    print resp.status_code

                    if(resp.status_code == 200):
                        jsonResponse = json.loads(resp.text)

                        creationDate = datetime.datetime.strptime(jsonResponse['Estimated Creation Date'], '%Y-%m-%dT%H:%M:%S') # 2006-02-22T00:00:00   "%Y-%m-%d"
                        tweetedDate = datetime.datetime.strptime(row['tweetedOn'], "%a %b %d %H:%M:%S +0000 %Y") # Tue Feb 03 08:16:18 +0000 2015

                        age = abs((creationDate.date() - tweetedDate.date()).days)

                        writer.writerow([row['serialNo'], row['tweetSeqNo'], row['tcoURL'], row['finalURL'], row['tweetedOn'], jsonResponse['Estimated Creation Date'], age])

                        if xx == 5:
                            break
                        xx += 1
                    else:
                        skip(row)


            except requests.exceptions.ConnectionError:
                print 'Cannot connect to server. Make sure server is running'
```

```
            except:
                print 'Exception'
                skip(row)


errorLogFile.close()
```

Listing 5: Python program to find the estimated creation date of URI using Carbon Dating and calculating the delta between this date and URI tweeted date

fetchWebpages.py

```
'''
Created on Feb 8, 2015

@author: rlambi
'''
import subprocess
import os
import thread
import threading
import csv
import datetime

def fetchWebPage(url):
    print 'Fetching ', url
    subprocess.Popen(["wget","-E","-H","-k","-K","-p", url])

sites = 'sites'
if not os.path.exists(sites):
    os.makedirs(sites)

os.chdir(sites)

fieldNames = ['sno','seqNum','tcoUrl','url']

print datetime.datetime.now()
with open('../sample-urls.txt') as csvfile:
    reader = csv.DictReader(csvfile, fieldnames=fieldNames, delimiter='\t')
    for row in reader:
        fetchWebPage(row['url'])
        thread.start_new_thread(fetchWebPage, (row['url'], ))

print datetime.datetime.now()

print 'Waiting for all threads to complete'
while threading.activeCount() > 1:
    print str(threading.activeCount())
    pass

print 'Completed fetching all webpages'
print datetime.datetime.now()
```

Listing 6: Python program to fetch the text for all the links

Histogram

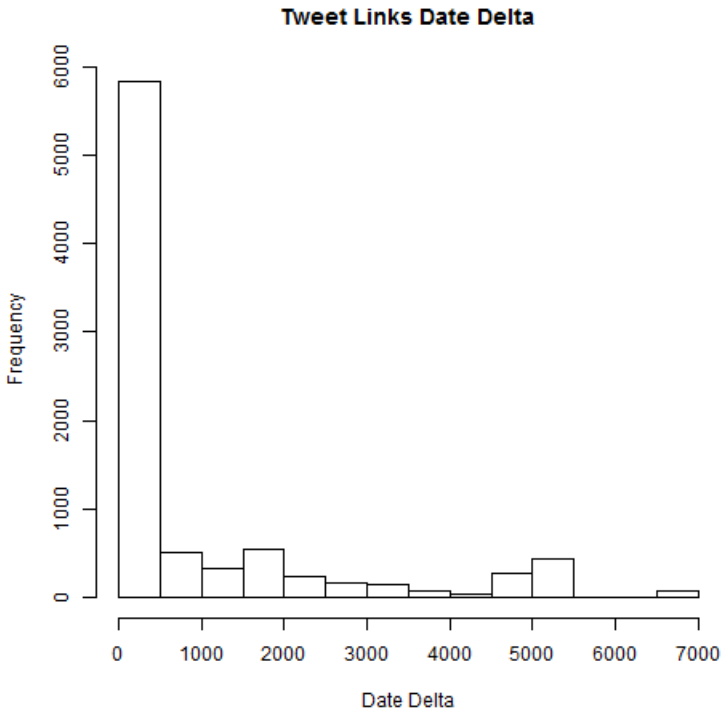

Figure 3: Delta between the Tweet and URI dates and their number of occurrences

tweet-link-age-histogram.R

```
1 inputFile=read.csv("url-dates.csv",stringsAsFactors=F,header=FALSE,sep="\t")
2 inputFile
3 data=inputFile[,1]
4 png("tweet-link-age-histogram.png")
5 hist(data,main="Tweet Links Date Delta",freq=T,xlab="Date Delta",ylab="Frequency")
6 dev.off()
```

Listing 7: R program to plot the histogram for Tweet and URI dates

2.3 Summary of Tweet and URI's date delta

Median: 93
Mean: 952.355
Standard Deviation: 1587.182
Standard Error: 17.03

3 References

[1] Tweepy library. https://github.com/tweepy/tweepy.

[2] Tweepy Documentation. http://docs.tweepy.org/en/v3.2.0/.

[3] Using twitter api keys. "http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/."

[4] Github for carbondate. https://github.com/HanySalahEldeen/CarbonDate.

[5] Producing simple graphs in r. http://www.harding.edu/fmccown/r/.