Assignment 2

Tarek Fouda

2016-02-10

1 Introduction

This report explains how I managed to solve Assignment 2 in Web Science class which is due 02/11/2016. It is mainly consisted of three Questions. Will show my approaches and implementation in each of the three Questions.

2 Problem 1- Extracting 1000 unique URLs from Twitter newsfeed

2.1 My approach

Baiscally I had to create a twitter account which I already have, also I had to register for a new twitter application to get the Consumer key, Consumer secret key, Key token and secret key token.

These variables allow you to use a Twitter python API, by just inserting these variables in your code. You would be able to, for example, Post a status from python code. But in this part of the assignment we are required to 1000 unique links from the newsfeed or even from someone's profile on Twitter.

I downloaded python-Twitter API from Github `https://github.com/ideoforms/python-twitter-examples`. It has a lot of python files which do specific tasks, for example twitter-list-retweets.py which lists all the retweets received on a specific tweet. What we only care about is extracting links. Creating a twitter application was the first thing I should do, following the steps on `https://github.com/ideoforms/python-twitter-examples` made me capable of creating a twitter app and now I can go to the config.py file and Type in my accesscode and secret keys Tokens as follows:

```
1 consumer_key = "KEtYeXDYJwgdtX0IHuwUf1Hsw"
2 consumer_secret = "
      xkErPgMDfParcniEkbhRLgf8T6EXGOfhNP1bAbKSbog5rqCKxi"
3 access_key = "308651543-bWKmgqe2AP3xTx85jyHPBUrovjdMtNej2SOqOjZd"
4 access_secret = "PgKsQJxjvqaocAZmNG2D5t2Q7ZkAcDoPTHvLfOEe2ghj9"
```

Listing 1: Config.py

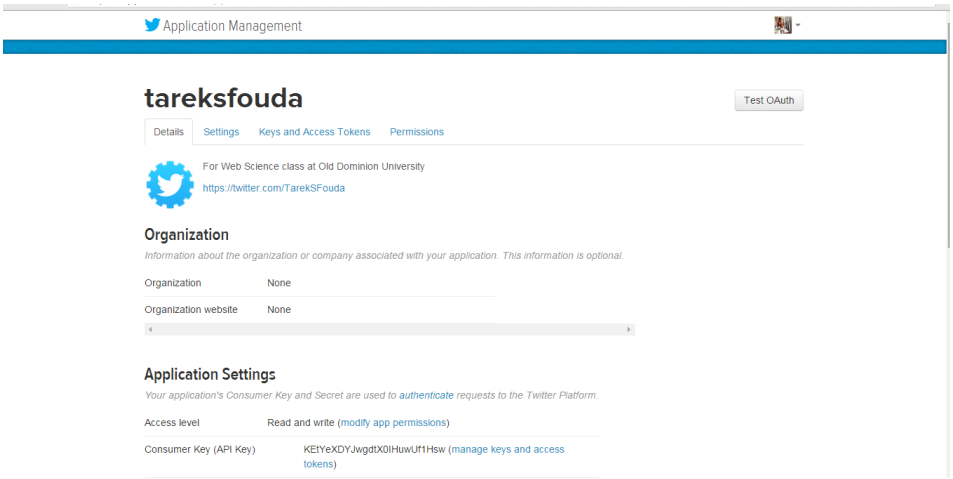

Figure 1: Twitter app

The actual code that should be implemented lies in twitter-stream-extract-links.py.

```
from twitter import *
# Tarek Fouda
# This program prints 1000 URLs in my twitter news feed
# using twitter API package downloaded from GITHUB.
config = {}
execfile("config.py", config)
auth = OAuth(config["access_key"], config["access_secret"], config[
    "consumer_key"], config["consumer_secret"])
stream = TwitterStream(auth = auth, secure = True)
tweet_iter = stream.statuses.filter(track = "social")
listCount = 1
textCount = 1
l = []
for tweet in tweet_iter:
        if (textCount<=1000):
                for url in tweet["entities"]["urls"]:
                        try:
                                l.insert(listCount,url["
                                    expanded_url"])
                                if (l.count(url["expanded_url"])<=
                                    1):
                                        print "Found URL: %s" % url
                                            ["expanded_url"]
                                        f = open('urls.txt','a')
                                        f.write(url["expanded_url"]
                                            +'\n')
                                        textCount = textCount +1
                                        f.close()
                                else:
                                        pass
                        except:
```

Figure 2: Printing the unique URLS

```
27                                  print "except"
28                  listCount = listCount+1
29          else:
30                  break
```

Listing 2: Python program for extracting links from my Twitter newsfeed

as shown above the code loops to find all Tweets found in my newsfeed, we basically put all the expanded URLS that are found in Twitter feed in a list. How can we assure that the link is Unique and not repetetive? This check happens in line 18. Tha if condition statement does not put the URL in the list unless it is unique. I kept all the llist in a text file by writing every unique URL in a text file called urls.txt as shown in line 20.

2.2 The list

Listing the 1000 links I got will make this report unofficial so I refered to the text file I created that has all the URLS which is named urls.txt

3 Problem number 2 - Finding TimeMaps and Histogram

In this problem, it was required to download the TimeMaps for each link I got in the first problem. I basically downloaded a python code from Github `github.com/jocll`. The file is called timemap.py and it simply takes a url as an argument and print out the TimeMaps and mementos for this URL. But in my situation, I need to find the TimeMaps for all URLS I have, so I had to amend the code to be able to read the URLs from the urls.txt and loop to perform getting Time Maps for all the URLs I have. the code was as follows:

```
1  from datetime import datetime
2  import dateutil.parser
3  import re
4  import StringIO
5  import urllib
6  import urllib2
7  import re
8  from sys import argv
9
10 #=============================================================================
11 # KEYWORDS
12 #=============================================================================
13
14 #=============================================================================
15 # REGULAR EXPRESSIONS
16 #=============================================================================
17 i = 0
18 tokenizer = re.compile('(<[^>]+>|[a-zA-Z]+="[^"]*"|[;,])\\s*')
19
20 #=============================================================================
21 class TimeMap(object):
```

```
22 #============================================================
23
24     def __init__(self, uri=None, data=None):
25         self.original      = None
26         self.timebundle    = None
27         self.timegate      = None
28         self.timemap       = None
29         self.first_memento = None
30         self.last_memento  = None
31         self.mementos      = {}
32         self.__tokens = TimeMapTokenizer(uri, data)
33         link = self.get_next_link()
34         while link != None:
35             if link[0] == 'memento':
36                 self.mementos[link[1]] = link[2]
37             elif link[0] == 'original':
38                 self.original = link[2] if link != None else None
39             elif link[0] == 'timebundle':
40                 self.timebundle = link[2] if link != None else None
41             elif link[0] == 'timegate':
42                 self.timegate = link[2] if link != None else None
43             elif link[0] == 'timemap':
44                 self.timemap = link[2] if link != None else None
45             elif link[0] == 'first memento':
46                 self.mementos[link[1]] = link[2]
47                 self.first_memento = link[1] if link != None else
                       None
48             elif link[0] == 'last memento':
49                 self.mementos[link[1]] = link[2]
50                 self.last_memento = link[1] if link != None else
                       None
51             link = self.get_next_link()
52
53     def get_next_link(self):
54         uri = None
55         datetime = None
56         rel = None
57         resource_type = None
58         for token in self.__tokens:
59             if token[0] == '<':
60                 uri = token[1:-1]
61             elif token[:9] == 'datetime=':
62                 datetime = token[10:-1]
63             elif token[:4] == 'rel=':
64                 rel = token[5:-1]
65             elif token[:5] == 'type=':
66                 resource_type = token[6:-1]
67             elif token[:6] == 'until=':
68                 datetime = token[7:-1]
69             elif token == ';':
70                 None
71             elif token == ',':
72                 return ( rel, dateutil.parser.parse(datetime)
73                                 if datetime != None else None,
74                                 uri, resource_type )
75             else:
76                 raise Exception('Unexpected timemap token', token)
```

```
77          if uri == None:
78              return None
79          else:
80              return ( rel, dateutil.parser.parse(datetime)
81                       if datetime != None else None,
82                       uri, resource_type )
83
84      def __getitem__(self, key):
85          return self.mementos[key]
86
87  #================================================================
88  class TimeMapTokenizer(object):
89  #================================================================
90
91      def __init__(self, uri=None, data=None):
92          if uri is not None:
93              self._tmfile = urllib2.urlopen(uri)
94          elif data is not None:
95              self._tmfile = StringIO.StringIO(data)
96          self._tokens = []
97
98      def __iter__(self):
99          return self
100
101     def next(self):
102         if len(self._tokens) == 0:
103             line = self._tmfile.readline()
104             if len(line) == 0:
105                 raise StopIteration
106             self._tokens = tokenizer.findall(line)
107         return self._tokens.pop(0)
108
109 #    ================================================================
110 # MAIN FOR TESTING
111 #    ================================================================
112
113 if __name__ == "__main__":
114     import logging
115     logging.basicConfig()
116     with open('urls.txt') as fil:
117         countOfzeromementos = 0
118         for line in fil:
119             urls=line
120             try:
121               timemap_uri = "http://mementoproxy.cs.odu.edu/aggr/
                      timemap/link/1/" + urls
122               tm = TimeMap(uri=timemap_uri);
123               print "Original:      ", tm.original
124               print "Time Bundle:   ", tm.timebundle
125               print "Time Gate:     ", tm.timegate
126               print "Time Map:      ", tm.timemap
127               print "First Memento:", tm.first_memento
128               print "Last Memento:  ", tm.last_memento
```

```
129                 print "Mementos:"
130                 for memento in sorted(tm.mementos.keys()):
131                     print memento, "=", tm.mementos[memento] + "{0}".
                        format(i)
132                     i = i+1
133             except:
134               countOfzeromementos = countOfzeromementos +1
135               print urls+ "has zero mementos, and the number of
                      urls having zero mementos are : " +"{0}".format(
                      countOfzeromementos)
```

Listing 3: Time map python code

In line 116, we Opened the text file to read all the URL's in there, and we looped on each URL to count the Time Maps, and if a 404 message is printed, then the corresponding link has zero mementos. that was kept track of in the exception part, from line 133.

Keeping in mind I kept track of the numbers of the links that have zero mementos, in line 134 of the code.

This a sampleof the command prompt upon running the amended code:

We were required also to draw a Histogram to show the number of URLs and the mementos, for example 100 URL's having 0 mementos and 200 URL's having 1 memento and so on.

Figure 3: A samplefor the URLs with their mementos

4 Problem number 3 Estimate the age of URLs

Finally the last part of the assignment was to estimate the age (Creation date) of each of the URLs, I downloaded Carbon date API and in the local.py file it was easy to pass a URL as an argument and print out its carbon date, but What is required is calculating the age of all URLs, so I implemented a function in cdate.py file to call the local.py downloaded within the Carbondate API, and pass every URL as an argument in a loop. so the final code will calculate the carbon date for all URLS. Then I copied and pasted the output in a file which to be read by a function in Python called q3.py which basically reads all the text file and disregard everything other than the URL and the corresponding Creation date.

```
import re
from sys import argv
txt = open("EstimatedTime.txt")
#print txt.read()
x=re.findall(r'\"(.+?)\"',txt.read())
lengthOfList = len(x)
counter=0
while(counter < lengthOfList):
        check2 = x[counter-2]
        check = x[counter]
        if(check[0].isdigit()):
                print check
                if(check2[0] == "h"):
                        print check2
        counter = counter +1
```

Listing 4: q3.py

Basically I put all the output in a text file and read it in a list, then as soon as I find a creation date, I print it out along with the corresponding URL, Keeping in mind I do not print out the URLs that do not have a creation date. Here is a figure which shows a sampleof the output upon calling q3.py :

The following figure represents a graph between The Creation Dates and mementos for only the links that have both mementos and creation dates. X axis represents the age in Years and Y- axis represent the corresponding mementos. The Graph is based on the Output file shown above and Figure 3 which states the mementos for all URLs.

http://inosmi.ru/social/20160208/235315525.html\\r
2016-02-11T22:49:21
http://injo.com/2016/02/535008-two-presidential-candidates-left-the-campaign-trail-to-vote-on-this-senate-bill/?utm_source=Twitter
2016-02-11T22:44:28
http://www.focus.de/regional/chemnitz/wer-kennt-diese-maenner-erst-klauen-sie-ein-auto-dann-werden-sie-geblitzt_id_5268569.html?utm_source=facebook
2015-05-19T10:37:23
http://www.skyscanner.ru/news/deshevye-strany-dlia-puteshestvii-v-2016-godu?utm_medium=social+paid
2016-02-11T22:47:54
https://blab.im/yuvraj-kewate-social-media-trends-and-tips-for-2016-http-bit-ly-23v0b0u?utm_source=twitter
2013-10-19T21:21:08
http://blog.hubspot.com/marketing/ultimate-guide-social-media-image-dimensions-infographic\\r
2015-07-19T11:46:55
http://cgn.mx/es/\\r
2016-02-11T08:57:01
http://www.newsminer.com/news/kris_capps/get-your-backyard-rinks-ready-for-hockey-week-contest/article_4d61ca40-d09d-11e5-9f61-63d744ea11ed.html?utm_medium=social
2014-01-10T13:06:42
http://smq.tc/SVkt1l\\r
2016-02-09T01:43:21
http://www.unotv.com/noticias/estados/distrito-federal/detalle/legisladores-violan-reglamento-transito-815110/?utm_source=shared
2016-02-11T22:49:39

Figure 4: A samplefor the output from q3.py, URLs and their age

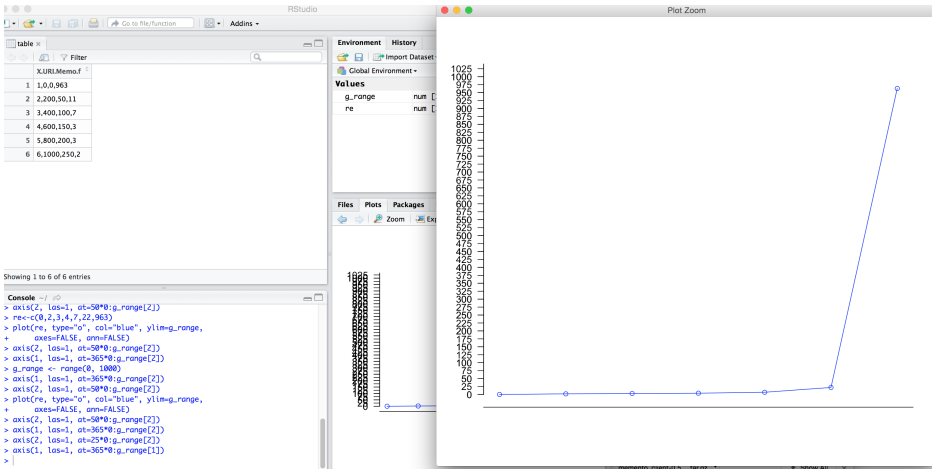

Figure 5: A samplefor the output from q3.py, URLs and their age