Srividya Majeti

Assignment 2

CS 532: Introduction to Web Science
Dr. Michael Nelson
Spring 2016

February 11, 2016

Contents

1

Question 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.

For solving the above problem I used Python programming language. Following are the steps I have taken to solve the given problem:

- For using the Twitter API, firstly I registered for a twitter application to generate a consumer key and consumer secret.
- Using the keys that are generated in the above step, I authenticated the application for requesting the tweets.
- To fetch the tweet data I started researching for packages and found multiple of them but I decided to work with 'tweepy'.
- Using 'extractTweets.py', I used the tweet data that I received from the API and fetched tweet text, list of URIs in the tweet, tweet JSON and tweet id. This code is listed in Listing3.1.
- While extracting the data mentioned above, program broke multiple times due to overloading, to resolve this issue I wrote an exception handler to wait with a sleep time of 60*15 and then continue.

- The data is processed in JSON format and written to a output file ‘tweet.json’.
- I loaded the JSON data from the above outputted file ‘tweet.json’, and obtained the final URI with a HTTP response code 200 by checking if the URI has any redirects in its history.
- To get unique URIs I stored all the final URIs obtained in the above step in a set data structure which has an inherent property of storing only unique data and have written into a file ‘uri.json’.

Code Listing

```
'''
CS532: Introduction to Web Science
Author: Srividya Majeti
Assignment 2
'''

import tweepy
import re
import json
import urllib2
import sys
import time
import requests
from sets import Set

CONSUMER_KEY = 'wTSsHE3PTA3ZZPiaKHEiQnLtf'
CONSUMER_SECRET = '
    UblYYCmNYIEffAY4T4QHGHXwAWMFqiueXdxf35xZFhoK3AECP1'
ACCESS_KEY = '157985123-
    WFvzlfDa8KStBZzevMfQBTM7fi8zKHYl2LQpTfGr'
ACCESS_SECRET = '
    lSax0XLwIimJ4VVbuU5OY9BpBic4vsSFi0riAq3DPvTxU'

auth = tweepy.auth.OAuthHandler(CONSUMER_KEY,
    CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)
api = tweepy.API(auth)

tweetJsonFile = open("tweet.json","a")
tweetCounter = 0
search_results = tweepy.Cursor(api.search, q="tesla", lang="
    en").items(5000)

while True:
        try:
                tweet = search_results.next()
                for tweet in search_results:
```

```
            #print tweet._json['entities']['urls']
            tweetJson = {}
            tweetJson['id'] = tweet.id
            tweetJson['text'] = tweet.text
            urlList = []
            uriCount = 0
            tweetCounter += 1
            for entityObj in tweet._json['entities']['urls']:
                uriCount += 1
                urlList.append(entityObj['url'])
            if uriCount > 0:

                tweetJson['uri'] = urlList
                tweetJson['json'] = tweet._json
                tweetJsonFile.write(json.dumps(tweetJson) + "\n")
            if tweetCounter > 1000:
                break
    except tweepy.TweepError:
        print "waiting \n"
        time.sleep(60*15)
        continue
    except StopIteration:
        break

tweetJsonFile.close()

f = open("tweet.json","r")
file= open("uri.json","a")
count = 0
UriSet = Set([])
for line in f:
    data = json.loads(line)
    if len(data['uri']) > 0:
        count += 1
    link= data['uri'][0]
    try:
        r = requests.get(link)
        if r.history:
            for h in r.history:
                UriSet.add(r.url)
                # print '[%s] %s' % (h.status_code, h.url)
            # print '[%s] %s' % (r.status_code, r.url)
```

```
                else:
                        # print '[%s] %s' % (r.status_code,
                            r.url)
                        UriSet.add(r.url)
        except Exception, e:
                print e
                continue

for item in UriSet:
        file.write("%s\n" %item)

print count

file.close()
```

Listing 1.1. "Python code for extracting tweets and checking for re-directs if it is a 200ok and unique then save it."

2

Question 2

Write a Python program that: 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

- With the help of ODU Memento Aggregator, I downloaded TimeMaps for all the URIs that are extracted in question 1 using the following curl command:
  `curl-i--silenthttp://mementoproxy.cs.odu.edu/aggr/timemap/link/1/<uri>`
- I stored the output produced by the cURL command into a file 'PagesList'.
- I processed the data from the above file to create a JSON with the original URI, memento URI , datetime and memento count. The memento count for each URI is derived by counting the URIs with rel="memento". This is outlined in Listing 3.1.
- Using 'uri.py' I wrote the memento count for all the 1000 URIs and URIs with $>$ 0 mementos in 2 different files 'allMementos' and 'mementosExcluding0' respectively.This code is listed in Listing 3.2.
- File 'mementosExcluding0' is used for question 3 and file 'allMementos' is used for plotting a histogram with memento count on x-axis and frequency on y-axis.
- I observed that most of the URIs are not archived as they are recently created tweets. Out of 1000 instances 849 tweets have 0 mementos.
- This distribution is summarized in Figure 3.1.

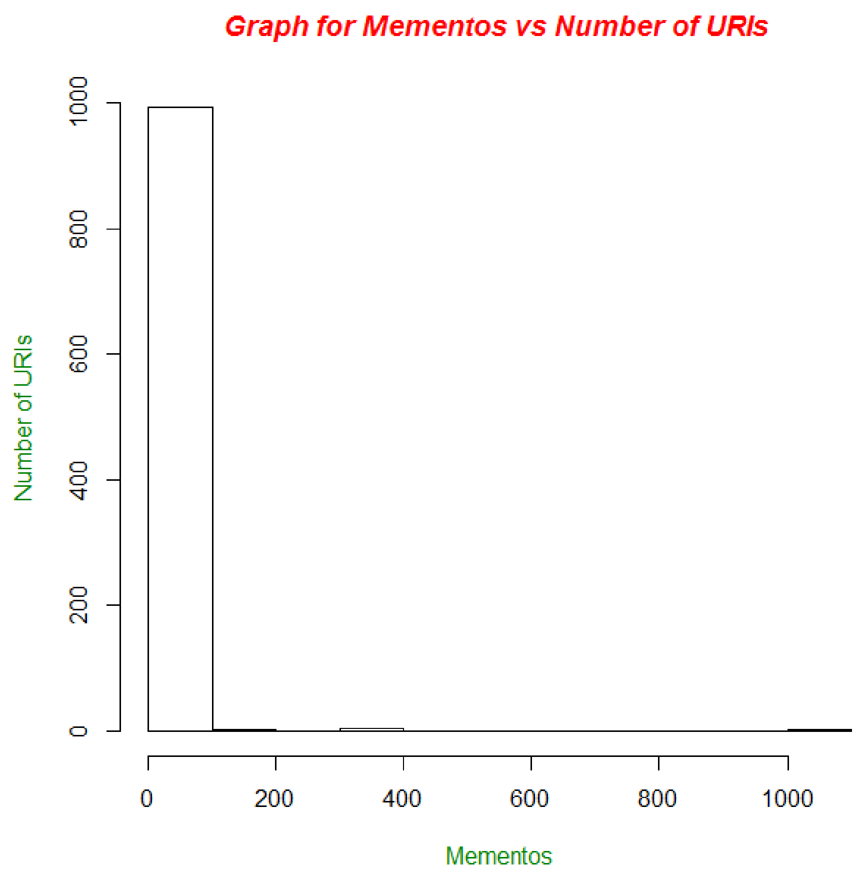


Fig. 2.1. Graph for Mementos vs Number of URIs

Code Listing

```
'''
CS532: Introduction to Web Science
Author: Srividya Majeti
Assignment 2
'''
import commands
import re
import json
import sys

def geturi():
    file=open("uri.json","r")
    uricounter = 0
    for line in file:
        if uricounter <1000:
            f = open('PagesList','w')
            response = getPages(line)
            f.write(response)
            f.close()
            finalCount =getMementosData()
            uricounter += 1
    file.close()

def getPages(uri):
    timemapUri = "http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/"
    command =" curl -i --silent " + timemapUri + str(uri).strip()

    pageList = commands.getoutput(command)
    return pageList

def getMementosData():
    getMementosDataFile = open("PagesList","r")
    outputfile= open('mementoData.json', 'a')
    mementoList= []
    mementoJson ={}
    Json ={}
    count = 0
    for line in getMementosDataFile:
        if 'rel="original"' in line:
            count = 0
            originalLink = (re.findall(r'(https?://[^\s]+>)', line))[0][:-1]
            Json['originaluri'] = originalLink
        if 'rel="memento"' in line:
            count += 1
            link = ""
            if re.findall(r'(https?://[^\s]+>)', line):
                link = (re.findall(r'(https?://[^\s]+>)', line))[0][:-1]
            elif re.findall(r'(www.[^\s]+>)', line):
                link = (re.findall(r'(www.[^\s]+>)', line))[0][:-1]
            else:
                # next(getMementosDataFile)
```

```
                                        print line
                                mementoId = count
                                mementoJson['mementouri'] = link
                                mementoJson['id'] =mementoId
                                if(line.find('datetime="') > -1):
                                        datetime = (line.split('
                                            datetime="'))[1].split('
                                            "')[0]
                                        mementoJson['datetime'] =
                                            datetime
                                mementoList.append(mementoJson)
                                Json['memento'] = mementoList
        finalCount = str(count)
        Json['count'] = finalCount
        outputfile.write(json.dumps(Json) + "\n")

geturi()
```

Listing 2.1. "Python code for getting mementos data and writing the output into a json file"

Code Listing

```
'''
CS532: Introduction to Web Science
Author: Srividya Majeti
Assignment 2
'''

import json

f= open('data.json','r')
f1 = open('urisWithMementosExcluding0','w')
f2 = open('allMementos','w')
f3=open ('mementosExcluding0','w')
for line in f:
        data= json.loads(line)
        f2.write(data['count'] + "\n")
        if data['count'] != "0":
                f1.write(line)
                f3.write(data['count'] + "\n")
```

Listing 2.2. "Python code for writing all mementos count and URIs with count > 0 in 2 different files "

3

Question 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.

- I downloaded the 'Carbon Date' tool from `http://ws-dl.blogspot.com/2014/11/2014-11-14-carbon-dating-web-version-20.html`.
- With the help of this tool, estimated creation date of URIs is obtained.
- I carbon dated URIs which had mementos > 0. This is outlined in Listing 3.1.
- Furthermore I parsed the estimated creation date if any and calculated the age of each URI in days using 'getCreatedTime.py'.. This is outlined in Listing 3.2.
- Figure 3.1 illustrates a graph with Mementos on x-axis and Age(in days) on y-axis. Looking at the graph I observe that for most of the URIs, as the number of mementos increases age also increases.
- Figure 3.2 illustrates the data of Age(in Days) and Number of mementos, with Number of mementos in ascending order.
- If I run the code for all the 1000 URIs, 525 URIs did not have estimated Creation date. But if I consider only the URIs that have > 0 mementos then all of them have an Estimated creation date.
- Out of the 1000 URIs, 151 URIs fulfilled the given criteria of having > 0 Mementos and an estimated creation date.

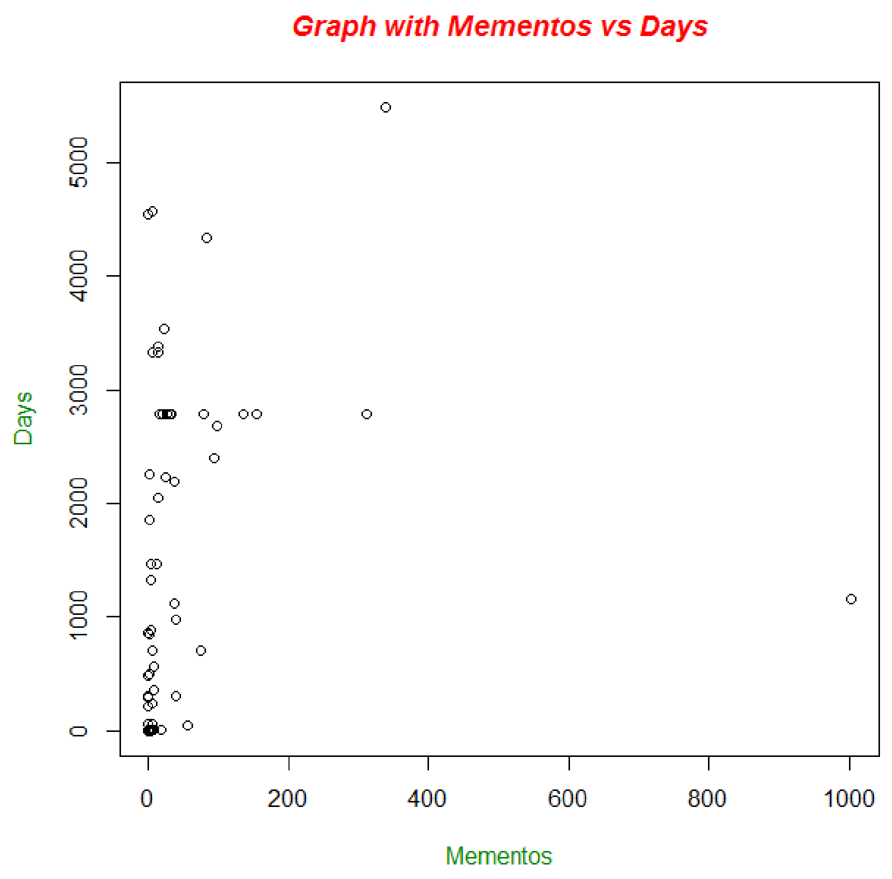
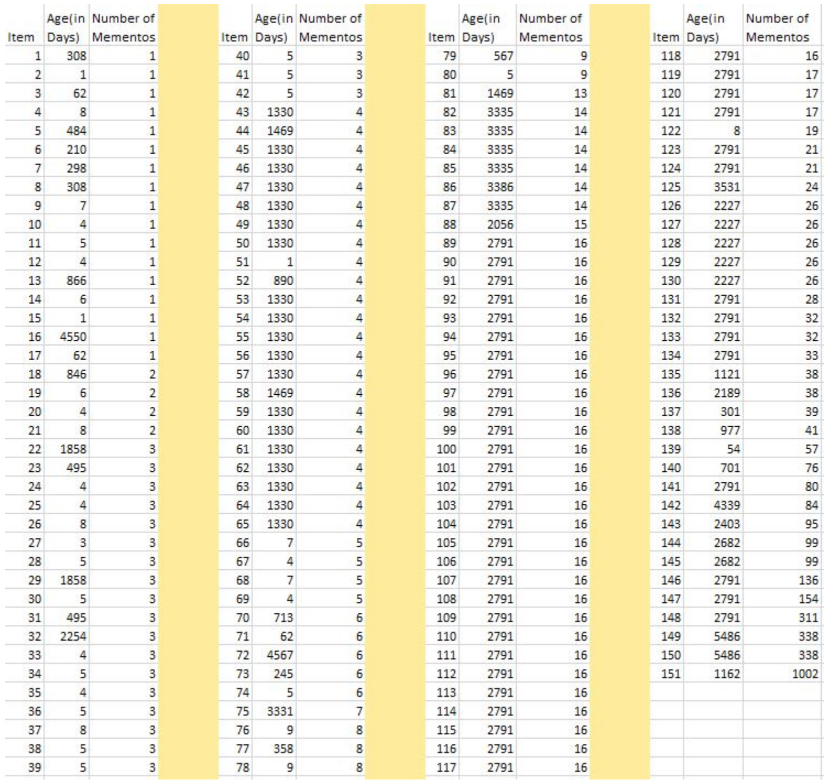


Fig. 3.1. Graph for Mementos vs age



Item	Age(in Days)	Number of Mementos
1	308	1
2	1	1
3	62	1
4	8	1
5	484	1
6	210	1
7	298	1
8	308	1
9	7	1
10	4	1
11	5	1
12	4	1
13	866	1
14	6	1
15	1	1
16	4550	1
17	62	1
18	846	2
19	6	2
20	4	2
21	8	2
22	1858	3
23	495	3
24	4	3
25	4	3
26	8	3
27	3	3
28	5	3
29	1858	3
30	5	3
31	495	3
32	2254	3
33	4	3
34	5	3
35	4	3
36	5	3
37	8	3
38	5	3
39	5	3
40	5	3
41	5	3
42	5	3
43	1330	4
44	1469	4
45	1330	4
46	1330	4
47	1330	4
48	1330	4
49	1330	4
50	1330	4
51	1	4
52	890	4
53	1330	4
54	1330	4
55	1330	4
56	1330	4
57	1330	4
58	1469	4
59	1330	4
60	1330	4
61	1330	4
62	1330	4
63	1330	4
64	1330	4
65	1330	4
66	7	5
67	4	5
68	7	5
69	4	5
70	713	6
71	62	6
72	4567	6
73	245	6
74	5	6
75	3331	7
76	9	8
77	358	8
78	9	8
79	567	9
80	5	9
81	1469	13
82	3335	14
83	3335	14
84	3335	14
85	3335	14
86	3386	14
87	3335	14
88	2056	15
89	2791	16
90	2791	16
91	2791	16
92	2791	16
93	2791	16
94	2791	16
95	2791	16
96	2791	16
97	2791	16
98	2791	16
99	2791	16
100	2791	16
101	2791	16
102	2791	16
103	2791	16
104	2791	16
105	2791	16
106	2791	16
107	2791	16
108	2791	16
109	2791	16
110	2791	16
111	2791	16
112	2791	16
113	2791	16
114	2791	16
115	2791	16
116	2791	16
117	2791	16
118	2791	16
119	2791	17
120	2791	17
121	2791	17
122	8	19
123	2791	21
124	2791	21
125	3531	24
126	2227	26
127	2227	26
128	2227	26
129	2227	26
130	2227	26
131	2791	28
132	2791	32
133	2791	32
134	2791	33
135	1121	38
136	2189	38
137	301	39
138	977	41
139	54	57
140	701	76
141	2791	80
142	4339	84
143	2403	95
144	2682	99
145	2682	99
146	2791	136
147	2791	154
148	2791	311
149	5486	338
150	5486	338
151	1162	1002

Fig. 3.2. Data for Number of Mementos and age

Code Listing

```
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))
```

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



    result = []

    result.append(("URI", url))
    result.append(("Estimated Creation Date", lowest))
    result.append(("Last Modified", lastmodified))
    result.append(("Bitly.com", bitly))
    result.append(("Topsy.com", "Topsy is out of service"))
    result.append(("Backlinks", backlink))
    result.append(("Google.com", google))
    result.append(("Archives", archives))
    values = OrderedDict(result)
    r = json.dumps(values)

    now1 = datetime.datetime.now() - now0


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

    return r


output = open('createdTime','w')
readData= open('urisWithMementosExcluding0.json','r')
uriCounter =0
for line in readData:
        # print line
        # print line.rstrip('\n')
        data= json.loads(line)
        uri = data['originaluri']
        uriCounter += 1
        if uriCounter <1001:
                r= cd(uri)
                output.write(r + "\n")

output.close()

# 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 3.1. "Python code which takes URIs > 0 Mementos as input and writes the JSON output with Estimated Creation date into a file."

Code Listing

```
'''
CS532: Introduction to Web Science
Author: Srividya Majeti
Assignment 2
'''

import json
import datetime
import dateutil.parser

now = datetime.datetime.now()
readCreatedTime = open('createdTime','r')
ageFile=open('UriWithAge.json','w')
noEstimatedDatecounter =0
for line in readCreatedTime:
        data= json.loads(line)
        Age= {}
        AgeList =[]
        if len(data['Estimated Creation Date']) >0:
                Age['uri'] = data['URI']
                EstimatedDate= data['Estimated Creation Date
                    ']
                d1 = dateutil.parser.parse(EstimatedDate)
                nowDate =now.isoformat()
                d2 = dateutil.parser.parse(nowDate)
                days = abs((d2 - d1).days)
                Age['days'] = days
                ageFile.write(json.dumps(Age) + "\n")
                # print Age
        else:
                noEstimatedDatecounter +=1
ageFile.close()

print "URI's with no EstimatedDate",noEstimatedDatecounter

read= open('UriWithAge.json','r')
output= open('days.json','w')
for line in read:
        data=json.loads(line)
        # print data['days']
        output.write(str(data['days'])+"\n")
```

Listing 3.2. "Python code for calculating age of URI in days and writing them into a file"

References

1. Bibliography management with bibtex:. pwdhttp://www.sharelatex.com/learn/.
2. code for extracting tweets:. http://stackoverflow.com/questions/22469713/managing-tweepy-api-search.
3. Get consumer key and secret:. https://twittercommunity.com/t/how-do-i-find-my-consumer-key-and-secret/646.
4. How to use bibtex:. http://www.bibtex.org/Using/.
5. Mementos data:. http://www.mementoweb.org/guide/quick-intro/.
6. Retrieve data from twitter:. http://stackoverflow.com/questions/15628535/how-can-i-retrieve-all-tweets-and-attributes-for-a-given-user-using-python code.