INTRODUCTION TO WEBSCIENCES: Assignment-1

Babitha Bokka

28 September 2014

Contents

1 Question 1 2

1.1 Description . . . 2

1.2 Approach Towards the Solution . . . 2

1.2.1 Desciption of searchTwitter.py . . . 2

1.2.2 Desciption of filterTwitter.py . . . 2

1.3 Source Code . . . 3

1.3.1 searchTwitter.py . . . 3

1.3.2 filterTwitter.py . . . 4

1.4 OutputFiles . . . 5

1.4.1 finalUniqueUri.txt . . . 5

2 Question 2 6

2.1 Description . . . 6

2.2 Approach Towards the Solution . . . 6

2.2.1 momentoTwitter.py . . . 6

2.3 Source Code . . . 7

2.3.1 momentoTwitter.py . . . 7

2.4 OutputFiles . . . 9

2.4.1 momentoUri.txt . . . 9

2.5 Histogram . . . 10

2.5.1 code to generate the Histogram histogramCode.txt . . . 10

2.5.2 Description of Histograms . . . 10

2.5.3 Histogram 1: momento / URI . . . 10

2.5.4 Histogram 2: momento / URI . . . 11

3 Question 3 12

3.1 Description . . . 12

3.2 Approach Towards the Solution . . . 12

3.2.1 description of daysCountTwitter.py . . . 12

3.2.2 description of momentoDays.py . . . 12

3.2.3 daysCountTwitter.py . . . 13

3.2.4 momentoDays.py . . . 14

3.3 OutputFiles . . . 15

3.3.1 carbonDateTwitter.txt . . . 15

3.3.2 carbonDateDays.txt . . . 15

3.3.3 momentoDays.txt . . . 15

3.4 Scatterplot . . . 16

3.4.1 Code to generate the Scatterplot scatterplotCode.txt . . . 16

3.4.2 Description of Histogram . . . 16

3.4.3 Scatterplot 1: . . . 16

3.4.4 Scatterplot 2: . . . 17

1 Question 1

1.1 Description

To extract 1000 unique URIs from twitter based on a searching Keyword and URIs should be non-redirecting.

1.2 Approach Towards the Solution

I started solving this problem with the search keyword ‘noodles’. I requested the four keys required to interact with the twitter API and searched for the keyword by using the TwitterSearchOrder() function. I extracted the expanded URIs from the JSON. I saved all the URIs into a file, and then filtered them by using the Python set datatype, which eliminates all duplicates. I grabbed only the URIs which returned the HTTP 200 Response (ok) to eliminate any redirection.

Note: In order to run the program, erase the existing finalUniqueUri.txt. All results are appended to the file which gives combines results from multiple runs.

1.2.1 Desciption of searchTwitter.py

1. Use TwitterSearchOrder()to search the Twitter API.
2. Choose a keyword to search.
3. Look for keyword in all tweets.
4. If there is a match then extract the expanded URL.
5. Save the extracted URL to extractedUri.txt.
6. extractedUri.txt has all non unique URLs.

1.2.2 Desciption of filterTwitter.py

1. Open the file and read each URL.
2. Request the URL.
3. Get the HTTP Response.
4. If the status code is 200(OK), save the URL to finalUniqueUri.txt.
5. finalUniqueUri.txt contains 1000 unique URLs.

1.3 Source Code

1.3.1 searchTwitter.py

```python
#!/usr/bin/env python
import re
import sys
import time
from TwitterSearch import *

#Main Function
def main():
    try:
        # create a TwitterSearchOrder object
        tso = TwitterSearchOrder()
        # search key word
        tso.setKeywords(['noodles'])
        # we want to see German tweets only
        tso.setLanguage('en')
        # look for 100 tweets per page
        tso.setCount(100)
        # and don't give us all those entity information(is the html)
        tso.setIncludeEntities(False)
        # keys to interact with the twitter API
        # my keys
        ts = TwitterSearch(
            consumer_key = 'fpTauqKqCRj4Gp8m9jb9WCilk',
            consumer_secret = 'OrDd7NssqrvLgOXnzuDkGcS8UbTNoY1jFYJF0HS6daxELfyI2k',
            access_token = '2822384568-jleRlhWap2Y7SMDW9y9tXkji95GHYDJPHK2IZ0b',
            access_token_secret = 'eVWGqNuLEk7xG1t47vLSkwBhJ6cQyNbeiZGShdRZXKF2A'
         )
        for tweet in ts.searchTweetsIterable(tso):
            # for a tweet points to user->entities->url->urls->(urls,expandes_url,)
            try :
                for sea in tweet['user']['entities']['url']['urls']:
                    # sea points to (urls,expandes_url...)
                    data = sea['expanded_url']
                    # if there is some data then write it to file
                    if data:
                        #print data
                        saveFile= open('extractedUri.txt','a')
                        saveFile.write(data)
                        saveFile.write('\n')
                        saveFile.close()
            # spent :( a night to resolve this error
            # not all tweets has expanded url so there is a key value excpetion we have to catch it .
            except KeyError :
                print 'error'
    # catch all the search exceptions if you dnt find a tweet
    except TwitterSearchException as e:
        print(e)
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

1.3.2 filterTwitter.py

```python
#!/usr/bin/env python
import sys
import time
import requests
import urllib2

# Main Function
def main():
    # set is a datatype which has all unique values
    processed_urls = set()
    # open the file which has all the extracted url
    f = open('finalUniqueUri.txt', 'r')

    lines = [ line.strip() for line in f.readlines() ]
    extracted_data = set (lines)

    # getting each lines from the list
    for url in extracted_data :
        try:
            response = requests.get(url=url, timeout=1)
            #print repr(response.headers)
            # get all the url with the 200 ok response so that they are unique
            if response.status_code == 200 :
                processed_urls.add(url)
                #print response.status_code,url
            else :#code for 300 400 to 500
                #print response.status_code
                pass
        except requests.exceptions.ConnectionError :
            pass
        except requests.exceptions.TooManyRedirects :
            pass
        except requests.exceptions.ReadTimeout :
            pass
    # get the all the links from set and store as a list
    final_processed_url = list (processed_urls)
    # OUT of all the links i need only 1000 links slicing the list
    for extracted_url in final_processed_url[0:1000]:
        # open the file to append to add the data
        saveFile= open('A2_final_output.txt','a')
        saveFile.write(extracted_url)
        saveFile.write('\n')
        # close the file
        saveFile.close()

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

1.4 OutputFiles

A sample of unique URI's :

1.4.1 finalUniqueUri.txt

```
 Sample Unique URI's :

http://soulcityusa.wordpress.com
http://jeanetteshealthyliving.com
https://www.youtube.com/watch?v=eqgShg7nk88&feature=youtube_gdata_player
http://youtube.com/kidrauhl
http://foodnex.tumblr.com
http://gammarayblog.tumblr.com/
http://www.dara-does-it.com
http://m.youtube.com/watch?v=6xmhlK456Hg
http://www.theprettybee.com
http://www.instagram.com/chrisllyvillanueva
http://www.intoxicatingprose.com/
http://www.vouchercodes.oneplaceshopping4less.com/
https://www.facebook.com/HootUSA
http://www.sweetherseyliving.com
http://twitpic.com/dzdqhe
http://instagram.com/astrobread
http://im.fireproof.uk
http://www.facebook.com/iwishiwas27
http://instagram.com/biancakee
http://geoffmoyle.com.au/
http://woorixx.tumblr.com
http://theelectronicshowcase.com
http://www.refugee-action.org.uk
```

2 Question 2

2.1 Description

To extract the timemaps of the 1000 unique URLs extracted obtained in Question 1 and count the number of mementos for each URL. Each memento represents a date and time where an individual URL was modified.

2.2 Approach Towards the Solution

To find the number of mementos, I used regular expressions (regexp) to locate the strings rel=”memento” and rel=”timemap”. Occurances of the string rel=”memento” were recorded to obtain a count of mementos for each URL. If there was a line containing rel=”timemap”, another page of mementos was available. I looped through each momento page until all mementos were counted. I then stored the results (mometo count, URL) in a text file.

2.2.1 momentoTwitter.py

1. Open finalUniqueUri.txt .
2. Read each URL.
3. Append it to http://mementoweb.org/timemap/link/ .
4. Request the complete URL.
5. Get the response, and count the number of mementos.
6. Check if there is a timemap line.
7. If a timemap line is encountered, loop and collect all momentos.
8. Store the results to momentoUri.txt .

2.3 Source Code

2.3.1 momentoTwitter.py

```python
#!/usr/bin/env python
import re
import sys
import time
import requests
import urllib2

#Main Function
def main():
    #open the file to read
    f = open('finalUniqueUri.txt', 'r')
    #regular expression to find the momento
    momento        = re.compile(r'rel.*?=.*?"memento".*?')
    #regular expresion to find the timemap
    time_map_match = re.compile(r'<[^>]+>;rel\w*?=\w*?"timemap".*?')
    # read all the lines from the file
    for line in f.readlines() :
        try :
            # add the url to the momento org to get the mometo(how may time the webpage has been modified)
            momento_url    = "http://mementoweb.org/timemap/link/" + line
            # get the response by opening the url
            response       = urllib2.urlopen(url=momento_url,timeout=10)
            # getting the  complete time map response
            time_map       = response.read()
            # count the number of momento
            count_momento  = len(momento.findall(time_map))
            # get the timemap string
            count_time_map_exist = time_map_match.findall(time_map)
            # while there is a timemap in the response (get all the count of momento as sometime the momento may be separte link)
            while len(count_time_map_exist) == 1 :
                # stripping out the url from the string_url extracted
                url            = count_time_map_exist[0]
                url_string     = url.strip('<')
                stripped_url   = url_string.split('>')
                momento_url_1  = stripped_url[0]
                # for the url extracted which has more momento loop it  utill we get all
                response_1     = urllib2.urlopen(url=momento_url_1,timeout=10)
                time_map_1     = response_1.read()
                count_momento  = len(momento.findall(time_map_1)) + count_momento
                count_time_map_exist = time_map_match.findall(time_map_1)

            saveFile= open('A2_momento.txt','a')
            saveFile.write("{:<20} {} " .format(count_momento,line))
            saveFile.close()

        except urllib2.HTTPError:
            #some url will not have timemap then make the timemap none
            time_map = None
            count_momento = 0
            saveFile= open('momentoUri.txt','a')
            #the way you write two or more elements to a file and format it to 20 spaces
            saveFile.write("{:<20} {} " .format(count_momento,line))
            saveFile.close()
        #catch the file errors like file caanot be opended
        except IOError :
            pass
        except urllib2.URLError :
            pass
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

2.4 OutputFiles

2.4.1 momentoUri.txt

```
Momento_Count  :      URI  :

0                       https://www.facebook.com/MyChickenRun
12                      http://www.youtube.com/watch?v=Eubi9YI2dKE
0                       http://geladoesntgiveadamn.tumblr.com/
0                       http://youtu.be/1BKO2V9EaZ0?a
0                       http://www.instagram.com/the_sanging_rebel
0                       http://instagram.com/teustimao/
17                      http://blogmylunch.com
0                       http://Facebook.com/Robbiewhaylez
0                       http://ifoodi.blogspot.com/
0                       http://www.talkinggoodfood.co.uk
0                       http://facebook.com/dimano.sterling
270                     http://www.yellowkorner.com
0                       http://Emerald.com
0                       http://attackontiphan.tumblr.com
0                       http://instagram.com/xxxmn88
0                       http://linggez-network.blogspot.com
0                       http://Instagram.com/hellocalm_
845                     http://www.wagamama.com
0                       http://pennyroyaltea.co.vu
0                       http://www.oufancyphones.com
0                       http://www.facebook.com/rappstartailgate
25                      http://thedaintypig.com
0                       http://instagram.com/victorparrini
```

2.5 Histogram

2.5.1 code to generate the Histogram histogramCode.txt

```
hist(A2_momento$momento,xlab= "momento" , ylab= "URI",main ="Histogram of momento/URI
    ",xlim=c(0,6000),ylim=c(1,1000),las=1,breaks=500)
```

2.5.2 Description of Histograms

Figure 1 represents mementos vs URI. If you observe the initial histogram, it does not give you a clear picture how many URIs have how many momentos.

The scaled histogram, Figure 2, provdes additional insight about URIs and respective mementos.

2.5.3 Histogram 1: momento / URI

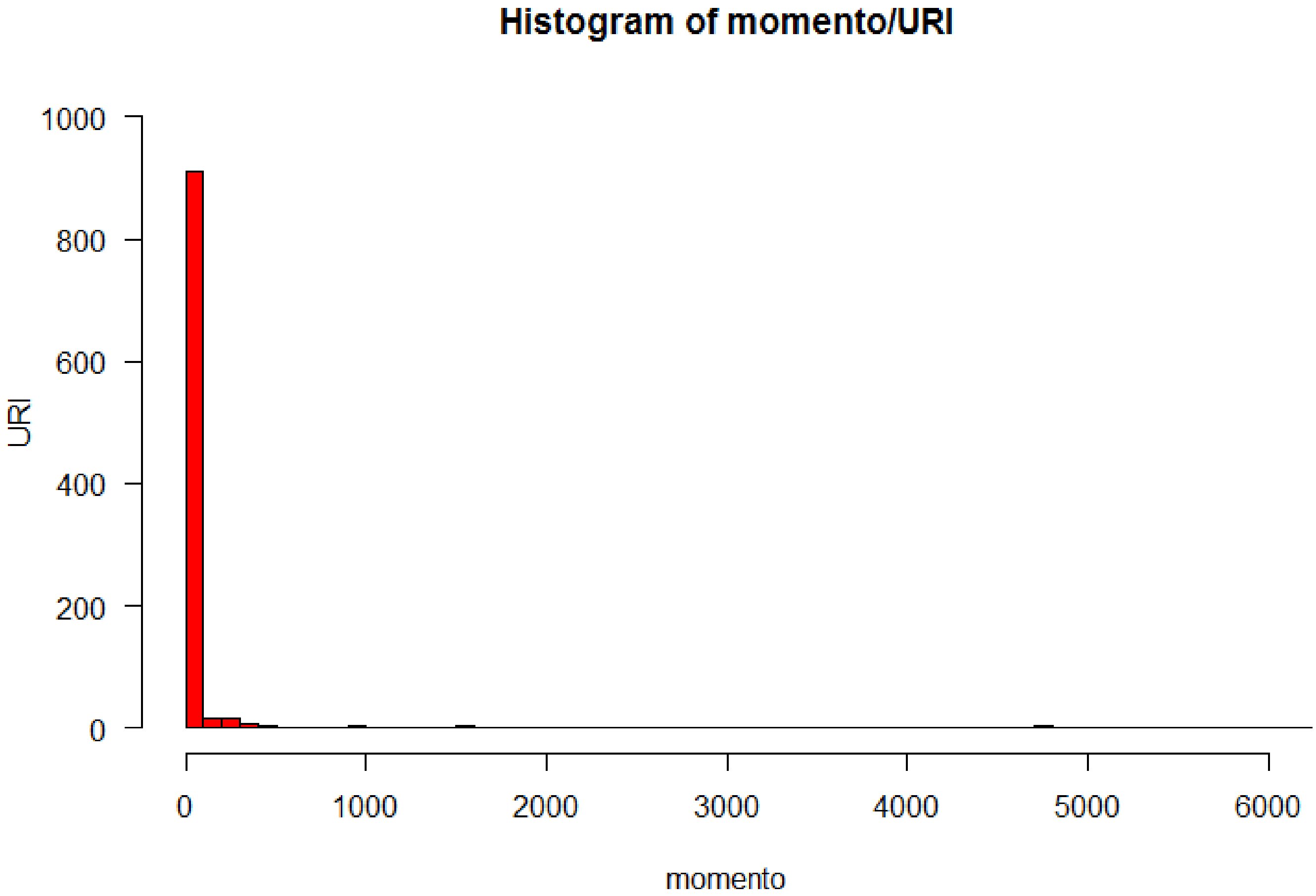

Figure 1: Intial-histogram

2.5.4 Histogram 2: momento / URI

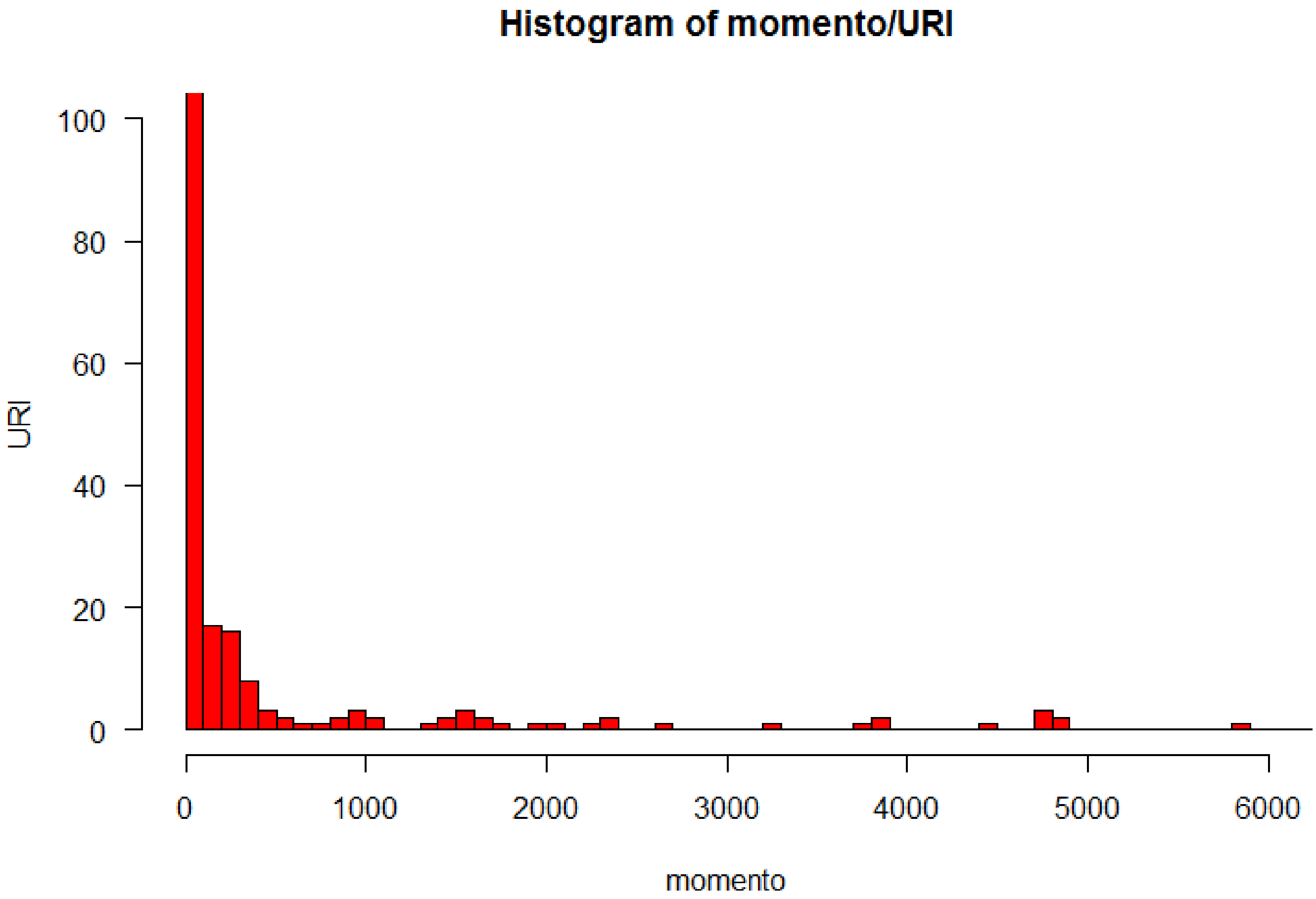

Figure 2: Scaled-histogram

3 Question 3

3.1 Description

Estimate the age of the each unique url by using the carbon date tool.

3.2 Approach Towards the Solution

To estimate the carbon date(estimated creation date) of each URL we download the carbondate tool files and run local.py get the creation dates and store them to a carbondateDays.txt.

CarbonDateTwitter.txt has the carbon date and URL, to estimate the age of the each url till today (date it was created and to till date gives us the number of days the url has been created) program daysCountTwitter.py will read each line and parses the date and estimates the number of days.

Relation between the memento and days can be obtained by running momentoDays.py which uses dictionary to store all the days and URL from carbonDateDays.txt for each URL it reads momentoUri.txt and checks whether there is a URL with greater than 0 momentos if it encounters any of the URL then that memento is appended to the dictionary.Then the results(days–momento) are stored in momentoDays.txt .

3.2.1 description of daysCountTwitter.py

1. Modify the local.py extract the dates for each URL.
2. Store in carbonDateTwitter.txt.
3. Now load the file in to daysCountTwitter.py.
4. Program calculates the number of days it has been since the URL has been created .
5. Store the days and URL into carbonDateDays.txt.

3.2.2 description of momentoDays.py

1. Open the file momentoDays.txt.
2. Store the days and URL into a dictionary with key as URL key:URL value :list[date] value as days.
3. Now open the momentoUri.txt .
4. Read each line and compare the URL with the dictionary key URL if there is a match and the number of mementos for that URL is greater that zero store the URL in momentoDays.txt.
5. momentoDays.txt has days , mementos.

3.2.3 daysCountTwitter.py

```
#!/usr/bin/env python
from datetime import datetime


#Main Function
def main():
    try :
        # current date
        now = datetime.now()
        # open the carbondate file whoch has all the dates when th eurl is created
        f = open('carbonDateTwitter.txt', 'r')
        # read all the lines(date,url)
        for line in f.readlines() :
            # split the line and strip all the spaces
            dateUrl=line.strip().split()
            # get the lenght after split
            len_date_url = len(dateUrl)
            # to get rid of lines (\r\n) since i did extract links from windows we had empty sets of data
            if len_date_url == 0:
                pass
            # if you just have date and url in each line
            elif len_date_url == 2 :
                date = dateUrl[0]
                url = dateUrl[1]

                try :
                    # using strip time function to convert the string date format to actual date type
                    date_object = datetime.strptime(date,"%Y-%m-%dT%H:%M:%S")
                    # get the number of days by subtracting the till date and past date
                    days = (now - date_object).total_seconds() / ( 3600.0 * 24 )
                    # convert that to int type
                    number_days = int(days)
                    # write it to file
                    saveFile= open('carbonDateDays.txt','a')
                    saveFile.write("{:<20} {} " .format(number_days,url))
                    saveFile.write('\n')
                    saveFile.close()
                # catch any exception generated from
                except :
                    date_object = datetime.strptime(date,"%Y-%m-%dT%H:%M:%S")
    except IOError :
            pass

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

3.2.4 momentoDays.py

```python
#!/usr/bin/env python
import sys
#Main Function
def main():
    try :
        # declaring a dictionary
        url_dict = {}
        # decalring s list
        mom_days = []
        # open the file
        f = open('carbonDateDays.txt', 'r')
        # read all the files
        for line in f.readlines() :
            # strip and split
            daysURL = line.strip().split()
            days     = daysURL[0]
            url      = daysURL[1]
            # assigning the key and value to dictionary
            url_dict[url] = [ int(days) ]
        f.close()
        f = open('momentoUri.txt', 'r')
        for line in f.readlines() :
            momento      = line.strip().split()

            if len(momento) == 2:
                try:
                    momento_count    = momento[0]
                    momento_url      = momento[1]
                    # appending the momento coutn to url== mometo_url , now it has days and moneto count
                    url_dict[ momento_url ].append( int( momento_count ) )
                except KeyError :
                    pass
        # close the file
        f.close()
    except IOError :
            pass
    try :
        for i,meme in url_dict.iteritems():
            if meme[1] >0 :
                # print 'days',meme[0]
                # print 'momento',meme[0]
                saveFile= open('momentoDays.txt','a')
                saveFile.write("{:<20} {} " .format(meme[0],meme[1]))
                saveFile.write('\n')
                saveFile.close()
    except ValueError :
        pass
    except IndexError :
        pass
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        sys.exit(1)
```

3.3 OutputFiles

3.3.1 carbonDateTwitter.txt

```
Carbon_Date :          URI:

2012-03-01T00:00:00   https://www.facebook.com/MyChickenRun
2011-01-04T00:00:00   http://www.youtube.com/watch?v=Eubi9YI2dKE
2011-07-01T00:00:00   http://geladoesntgiveadamn.tumblr.com/
2014-05-21T00:00:00   http://youtu.be/1BKO2V9EaZ0?a
                      http://www.instagram.com/the_sanging_rebel
2012-06-26T00:00:00    http://blogmylunch.com
                      http://Facebook.com/Robbiewhaylez
2008-02-18T00:00:00    http://ifoodi.blogspot.com/
2012-12-11T00:00:00    http://www.talkinggoodfood.co.uk
2013-03-19T00:00:00    http://facebook.com/dimano.sterling
2010-07-15T00:00:00    http://www.yellowkorner.com
2001-02-01T00:00:00    http://Emerald.com
```

3.3.2 carbonDateDays.txt

```
Days  :       URI's  :

939                   https://www.facebook.com/MyChickenRun
1361                  http://www.youtube.com/watch?v=Eubi9YI2dKE
1183                  http://geladoesntgiveadamn.tumblr.com/
128                   http://youtu.be/1BKO2V9EaZ0?a
822                   http://blogmylunch.com
2412                  http://ifoodi.blogspot.com/
654                   http://www.talkinggoodfood.co.uk
556                   http://facebook.com/dimano.sterling
1534                  http://www.yellowkorner.com
4985                  http://Emerald.com
655                   http://attackontiphan.tumblr.com
985                   http://instagram.com/xxxmn88
```

3.3.3 momentoDays.txt

```
Days:               Momento:

2887                  18
 822                  17
1993                  31
1361                  12
4985                 165
293                   205
4985                  3769
4620                  5
2107                  185
1534                  270
4227                  7179
2885                  427
```

3.4 Scatterplot

3.4.1 Code to generate the Scatterplot scatterplotCode.txt

```
plot(momento_days$Days , momento_days$Momento , xlab="Days",ylab="Momento",main ="
    ScatterPlot for Days/Momento",las=1,xlim=c(1,5000),ylim=c(0, 10000),col=2)
```

3.4.2 Description of Histogram

Figure 1 brings up the relation between the days and the memento.Figure 1 and Figure 2 are plotted on the same data but Figure 2 gives additional insight.

3.4.3 Scatterplot 1:

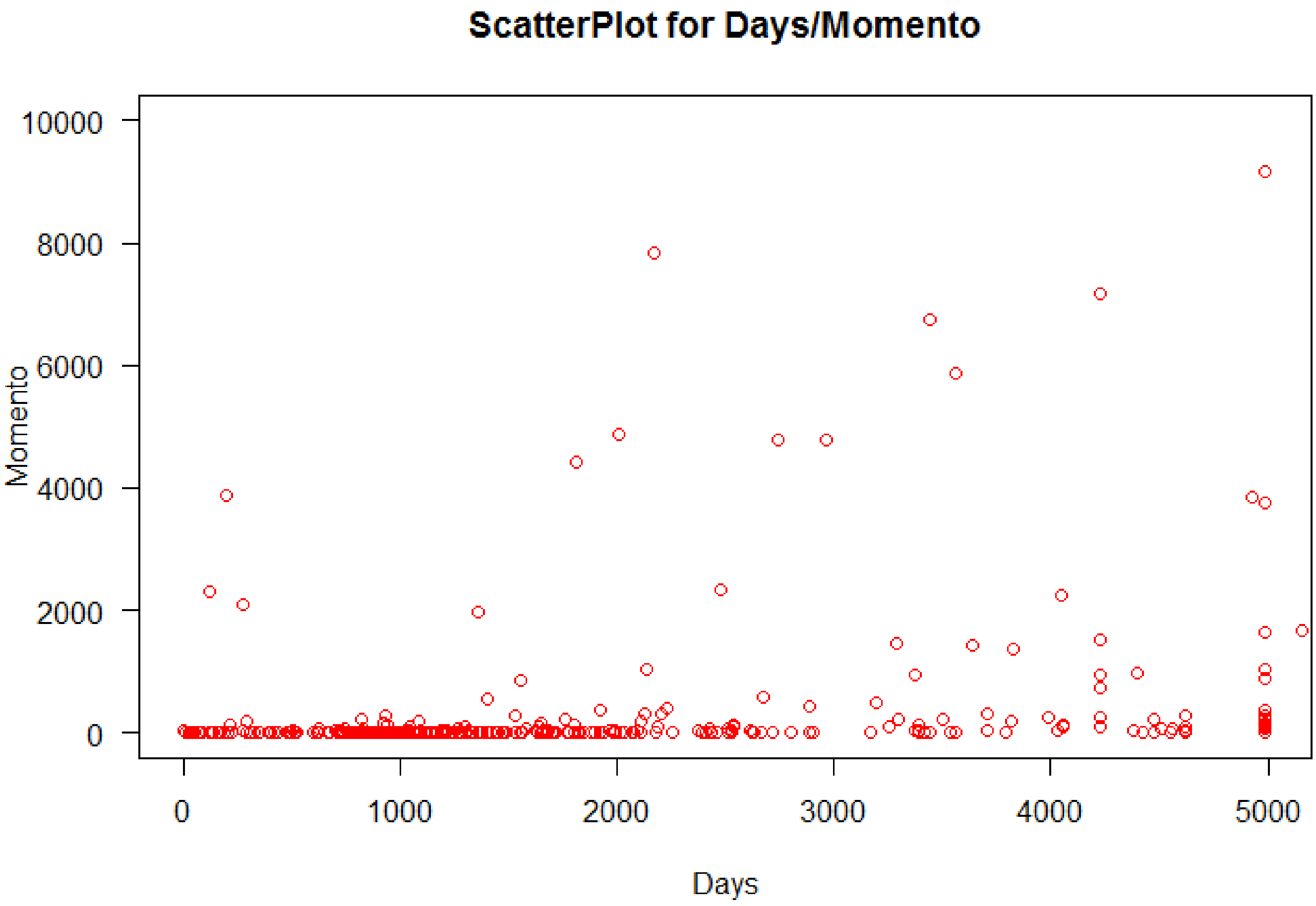

Figure 3: intial scatterplot

3.4.4 Scatterplot 2:

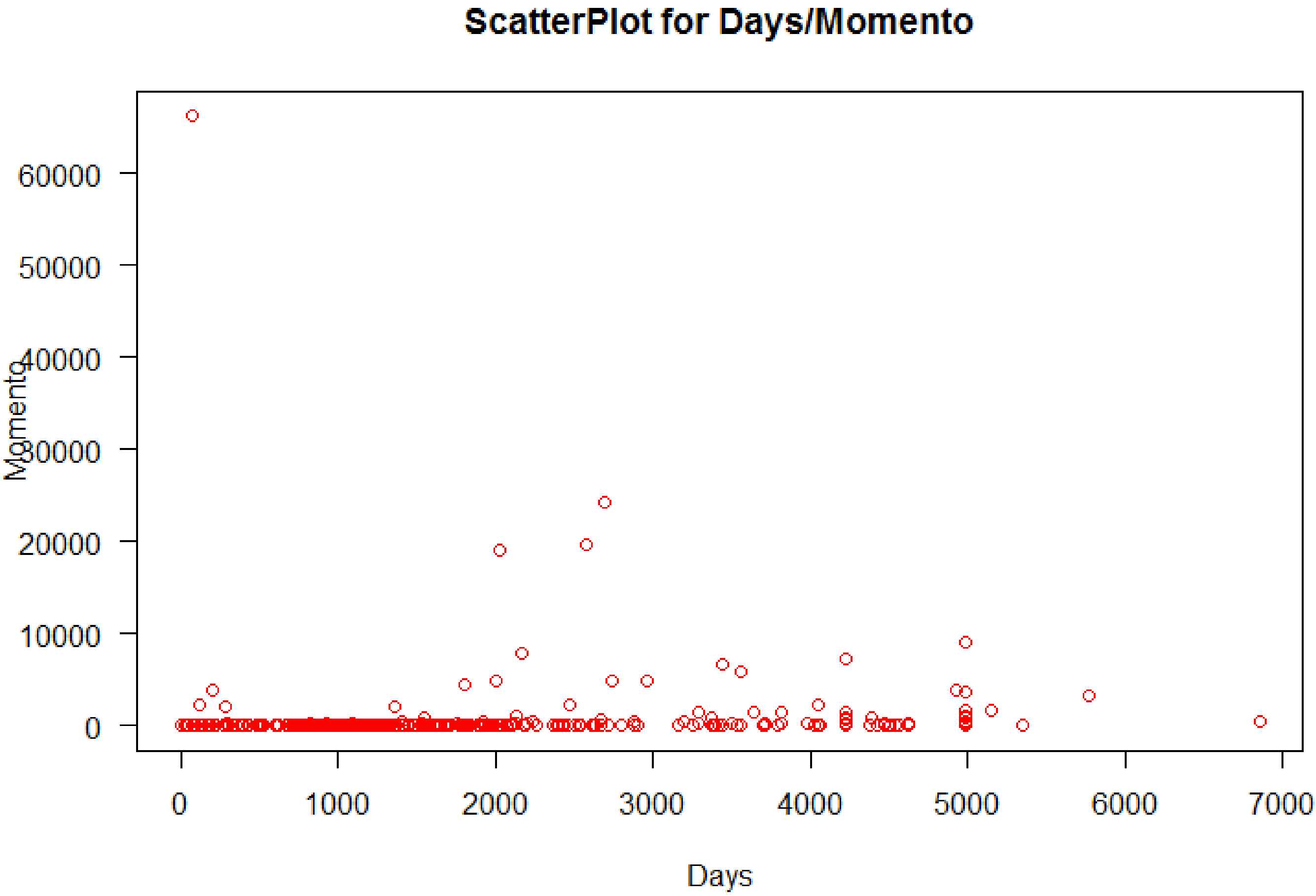

Figure 4: Scaled scatterplot

References

[1] error. http://www.dotnetperls.com/keyerror.

[2] latex search. https://www.sharelatex.com/learn/Code_listing.

[3] python. http://www.toptal.com/python/top-10-mistakes-that-python-programmers-make.

[4] search. https://pypi.python.org/pypi/TwitterSearch/.

[5] tweepy search. https://www.youtube.com/user/sentdex.

[6] Twitter api keys. https://apps.twitter.com/app/6952225/keys.

[7] youtube. https://www.youtube.com/watch?v=phsj6TUNZeI.

[8] youtube-tutorial. http://youtu.be/Hj1pgap4UOY.

[]