CS532S19: Assignment #2

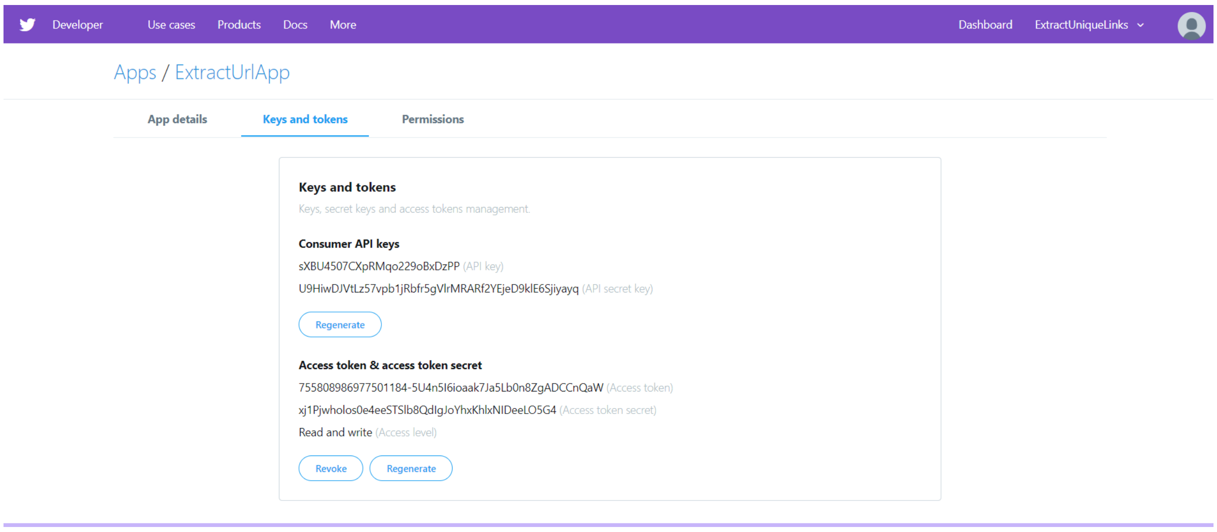
Due on Thursday, February 14, 2019

Nwala, Alexander C

Giridharan Ganeshkumar

Question 1

Write a Python program that extracts 1000 unique (collect more e.g., 1300 just in case) links from Twitter. Omit links from the Twitter domain (twitter.com).



Keys and tokens

Keys, secret keys and access tokens management.

Consumer API keys

sXBU4507CXpRMqo229oBxDzPP (API key)

U9HiwDJVtLz57vpb1jRbfr5gVlrMRARf2YEjeD9klE6Sjiyayq (API secret key)

Access token & access token secret

755808986977501184-5U4n5I6ioaak7Ja5Lb0n8ZgADCCnQaW (Access token)

xj1Pjwholos0e4eeSTSlb8QdlgJoYhxKhlxNIDeeLO5G4 (Access token secret)

Read and write (Access level)

1. As shown in the above screen shot, the first step was to sign into developer.twitter.com and create an app. In this case the app created was ExtractUniqueLinks. Also created an Access token and access token secret
2. The next step was to install tweepy. pip install tweepy
3. The first step in the script is to authorize through the tweepy.OAuthHandler and pass the consumer api key and consumer api secret key. Also set the access token with object created in the previous step by passing the access token and access token secret generated in the first step.
4. As show in the below script we have derived class listener from the base class StreamListener
5. The next step is to start the stream by sending the authorization object and the listener derived class. Also filter the stream with some filters sports,music,awards,NASA,cricket,arts,nature,computers,gardening,genetics and medicine.
6. Inside the listener we override the on data method to capture the tweets being streamed, for assignments purpose had a counter to restrict the number tweets being received. The received tweet json is passed to json.load method so that the we can access the entities-urls.
7. loop through the list of urls, exclude urls containing twitter.com and write other urls to the text file.

Listing 1: Python Script

```python
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import json
import re

class listener(StreamListener):

```

Question 1 continued on next page. . .

```
    def __init__(self, api=None):
        super(listener, self).__init__()
        self.num_tweets = 0

    def on_data(self,data):
        #print (data)

        if self.num_tweets < 2001:
            tweet = json.loads(data)
            for url in tweet["entities"]["urls"]:
                if('twitter.com' not in url["expanded_url"]):
                    self.num_tweets += 1
                    fileToStoreListOfUrls.write(url["expanded_url"] + '\n')
                    print(self.num_tweets)
            return True
        else:
            print('Exception')

            return False

    def on_error(self, status):
        print(status)

consumerApiKey = 'sXBU4507CXpRMqo229oBxDzPP'
consumerApiKeySecret = 'U9HiwDJVtLz57vpb1jRbfr5gVlrMRARf2YEjeD9klE6Sjiyayq'
accessToken = '755808986977501184-5U4n5l6ioaak7Ja5Lb0n8ZgADCCnQaW'
accessTokenSecret = 'xj1PjwhoIos0e4eeSTSlb8QdIgJoYhxKhlxNIDeeLO5G4'

fileToStoreListOfUrls = open("C:\ListOfUrls.txt","a+")
auth = tweepy.OAuthHandler(consumerApiKey, consumerApiKeySecret)
auth.set_access_token(accessToken, accessTokenSecret)
twitterStream = Stream(auth,listener())
twitterStream.filter(track=['sports','music','awards','NASA','cricket','
    arts','nature','computers','gardening','genetics','medicine'])
fileToStoreListOfUrls.close()
print ('Completed Streaming.')
```

1. As show in the below script we now expand the url to resolve all the redirects and shortened URIs
2. The expanded URIs are written into a file for further processing.

Listing 2: Python Script

```
import urllib.request

listOfUrlsFile = open("C:\ListOfUrls.txt", "r")
fileToStoreListOfUrlsExpanded = open("C:\ListOfUrlsExpanded.txt","a+")
for currentUrl in listOfUrlsFile:
    try:
        with urllib.request.urlopen(currentUrl) as url:
            currentExpandedUrl = url.geturl()
            fileToStoreListOfUrlsExpanded.write(currentExpandedUrl + '\n')
    except:
        pass
```

Question 1 continued on next page...

```
listOfUrlsFile.close()
fileToStoreListOfUrlsExpanded.close()
```

1. The final step as show in the below script is to remove the duplicates. Got the required help from the professor to get the canonicalizeURI which strips down the url removes the scheme and query strings and other extra parameters which make it now easy for comparison.

2. To get the unique urls we define a set of uniqueUrls and in the for loop add it to the set if it does not exist already in the list. Write it to the file of unique urls ListOfUrls.txt.

3. All the output files and the intermediate files are saved to the OutputFiles folder in git hub.

Listing 3: Python Script

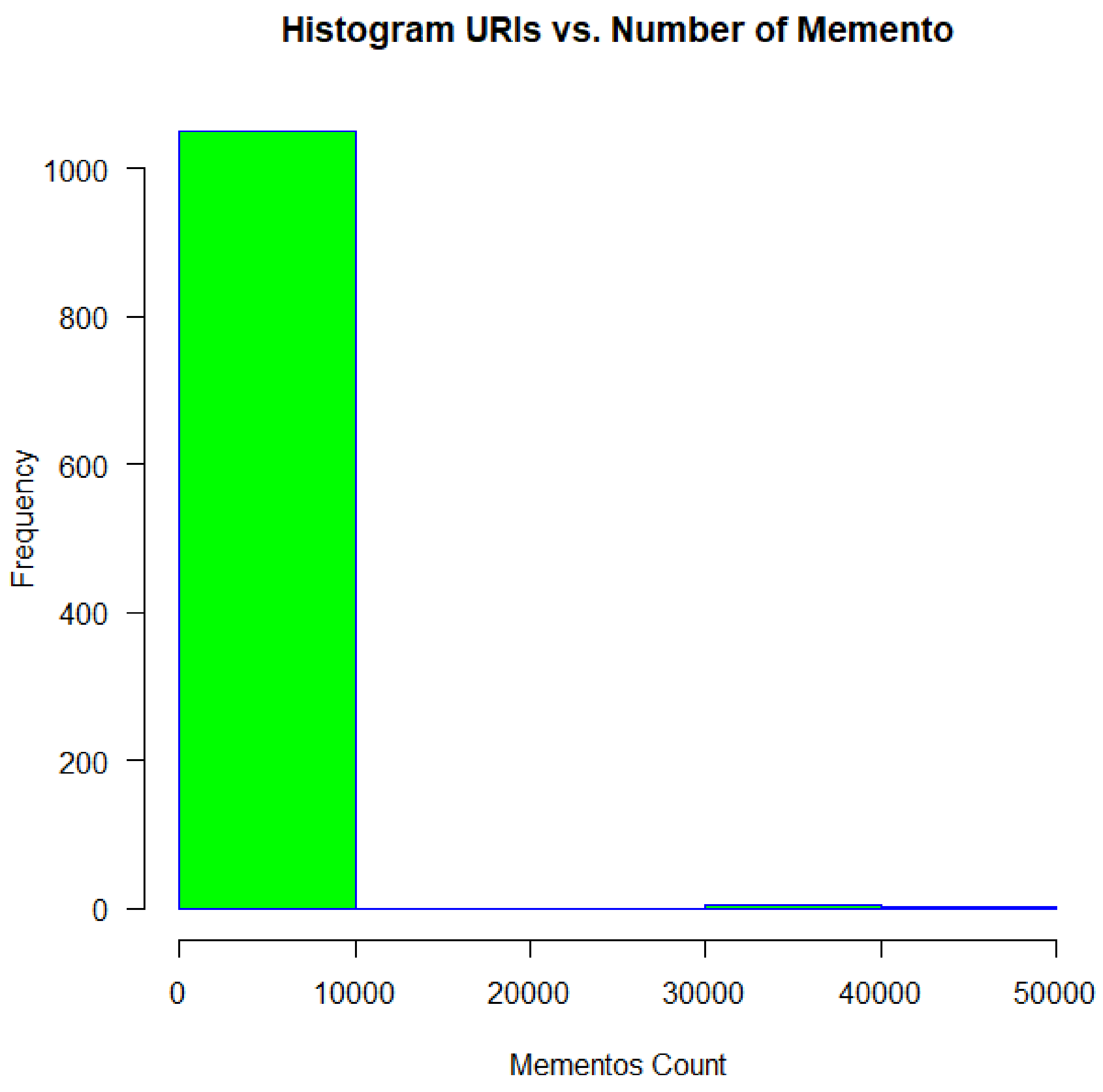
```
from urllib.parse import urlparse
import hashlib

def canonicalizeURI(uri):
    uri = uri.strip()
    if( len(uri) == 0 ):
        return ''
    exceptionDomains = ['www.youtube.com']
    try:
        scheme, netloc, path, params, query, fragment = urlparse( uri )
        netloc = netloc.strip()
        path = path.strip()
        optionalQuery = ''

        if( len(path) != 0 ):
            if( path[-1] != '/' ):
                path = path + '/'

        if( netloc in exceptionDomains ):
            optionalQuery = query.strip()

        return netloc + path + optionalQuery
    except:
        print('Error uri:', uri)

    return ''

uniqueUrls = set()
output_file = open("C:\ListOfUniqueUrls.txt", "w")
for currentUrl in open("C:\ListOfUrlsExpandedInput.txt", "r"):
    currentCanonicalizedURI = canonicalizeURI(currentUrl)
    if currentCanonicalizedURI not in uniqueUrls:
        output_file.write(currentUrl)
        uniqueUrls.add(currentCanonicalizedURI)
output_file.close()
```

Question 2

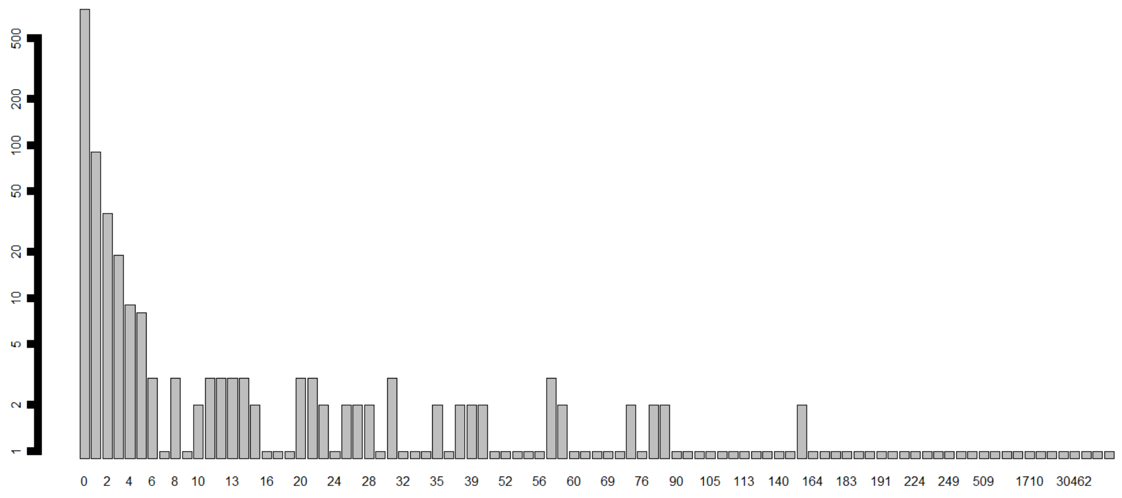
Download the TimeMaps for each of the target URIs. We'll use the ODU Memento Aggregator. Create a histogram of URIs vs. number of Mementos (as computed from the TimeMaps). For example, 100

Question 2 continued on next page...

URIs with 0 Mementos, 300 URIs with 1 Memento, 400 URIs with 2 Mementos, etc. The x-axis will have the number of mementos, and the y-axis will have the frequency of occurence.



Question 2 continued on next page. . .



1. As mentioned we will be using the ODU Memento Aggregator for this purpose.
2. A function getTimeMapAndMomentos is defined that accepts the uri and the currentUrlLineCounter as an argument.
3. We have the memgatorJsonRequestUrlPrefix which contains the ODU memgator url along with the format timemap and json attributes included. In this case - https://memgator.cs.odu.edu/timemap/json/
4. The uri received in the method is appended to the memgatorJsonRequestUrlPrefix and a GET request is made.
5. The response is stored in a file with the name as the currentUrlLineCounter
6. Also, we return the mementos by reading from the json, that is by getting the length of the mementos list tag
7. The main script as shown below reads the list of unique urls form the previous problem's output and sends the url to the getTimeMapAndMomentos method. The returned mementos is written into a csv file. We write into a csv so that we can have a one to one mapping between the url and the memento.

Listing 4: Python Script

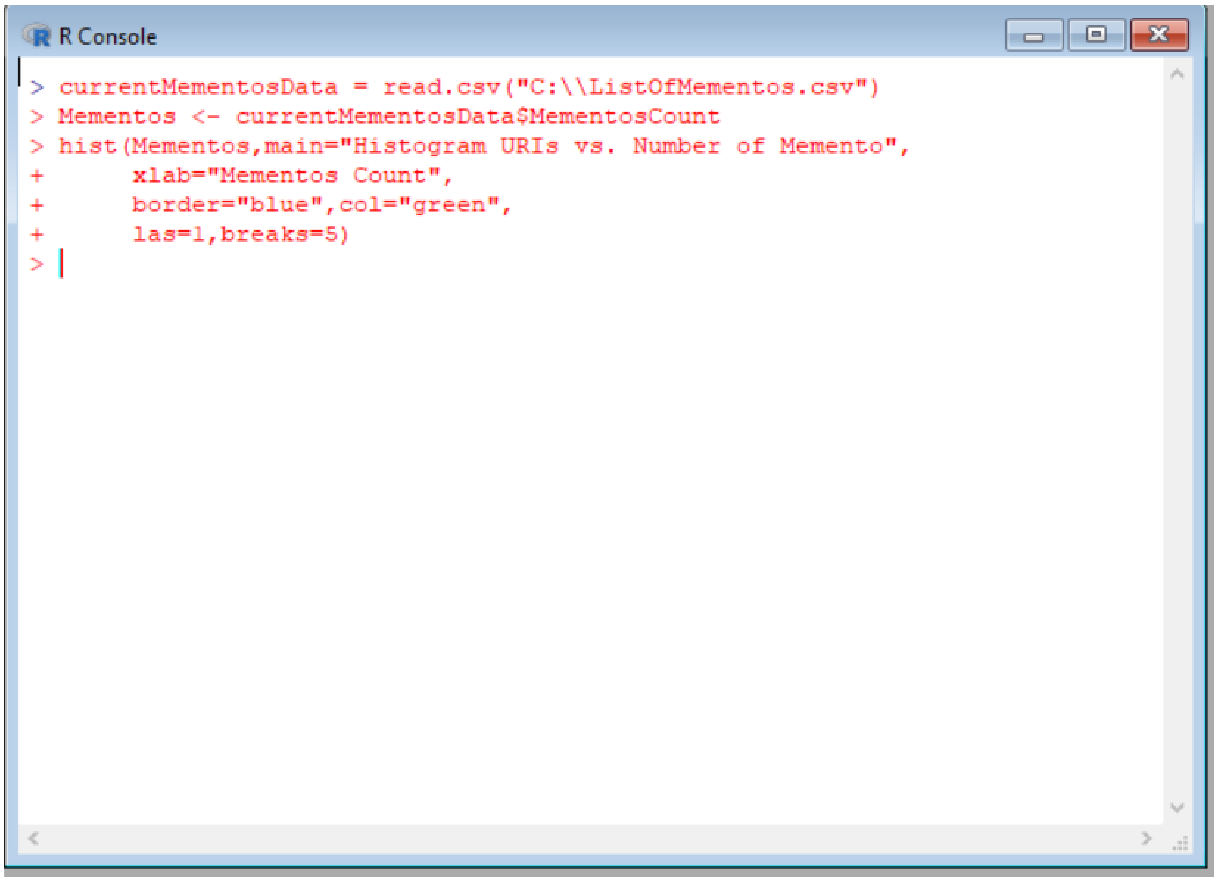
```
import urllib.request
import json
import csv

urlCounter = 0
def getTimeMapAndMomentos(uri,currentUrlLineCounter):
    try:
        currentfileName = "C:\Timegate\\" + str(currentUrlLineCounter) + ".
            txt"
        output_file = open(currentfileName , "w")
        memgatorJsonRequestUrlPrefix = "https://memgator.cs.odu.edu/timemap
            /json/"
        contents = urllib.request.urlopen(memgatorJsonRequestUrlPrefix +
            uri).read()
```

Question 2 continued on next page...

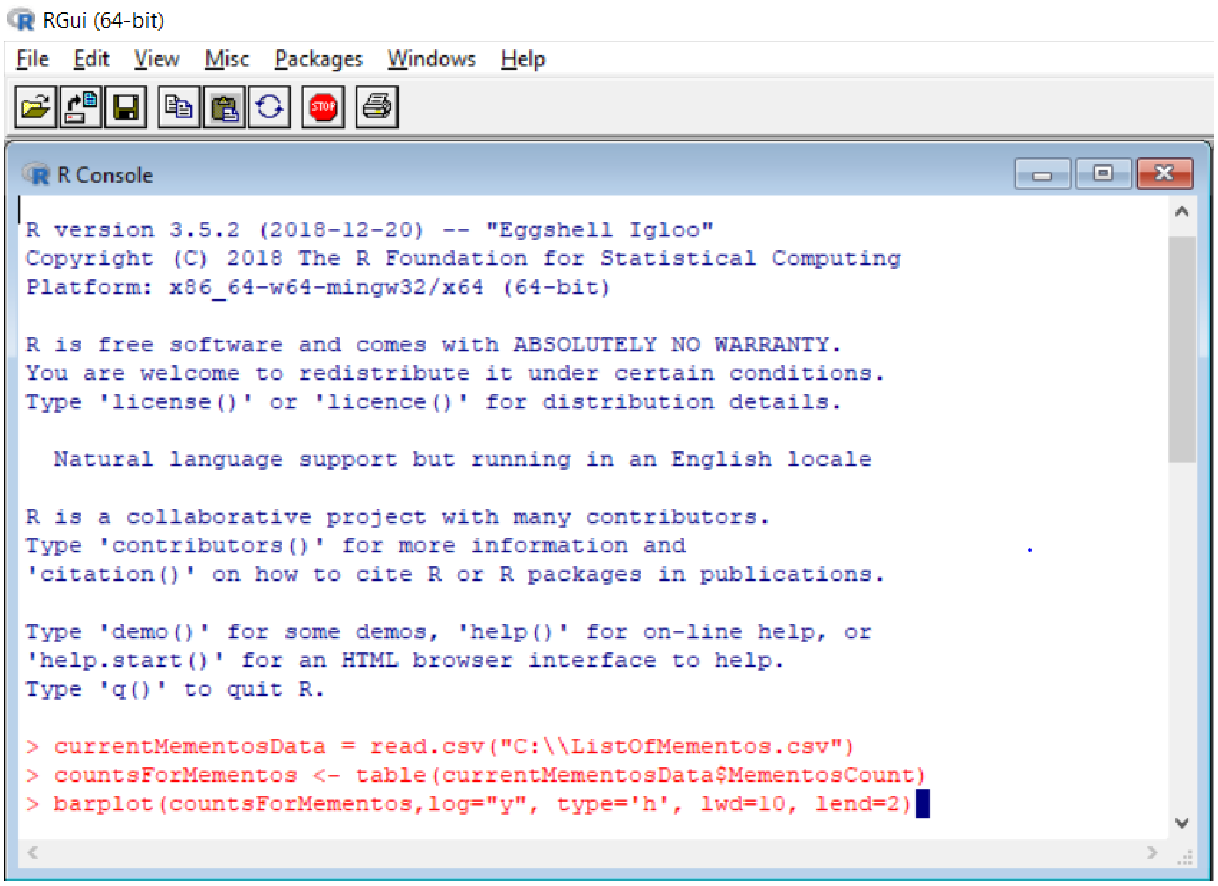
```
            #print(contents)
            jsonContents = json.loads(contents)
            output_file.write(str(contents))
            output_file.close()
            mementosCount = len(jsonContents["mementos"]["list"])
            return mementosCount
    except:
            return 0

with open("C:\ListOfMementos.csv", "w", newline='') as
    csvFileListOfMementosTemp:
    writer = csv.writer(csvFileListOfMementosTemp)
    currentData = ['URL','MementosCount']
    writer.writerow(currentData)
    for currentUrl in open("C:\ListOfUniqueUrls.txt", "r"):
        urlCounter = urlCounter + 1
        currentMementosCount = getTimeMapAndMomentos(currentUrl,urlCounter)
        currentData = [currentUrl,currentMementosCount]
        writer.writerow(currentData)
        print(urlCounter)
csvFileListOfMementosTemp.close()
print("Completed.")
```

1. Once we have the data in csv, we can now use the R package to plot the graph.
2. We draw a histogram using R Package, we take the data and pass it to an vector object. Then pass the vector data to the hist() plot graph method as shown below.
3. Also below the histogram is a bar plot that has the distinct count urls for every mementos count. To achieve this we transpose the data to get the counts of url for a given number of mementos count as mentioned in the second command line.
4. Then plot it using barplot method so that we have the mementos count on the x-axis and the count of urls on the y-axis.

Question 2 continued on next page. . .



```
> currentMementosData = read.csv("C:\\ListOfMementos.csv")
> Mementos <- currentMementosData$MementosCount
> hist(Mementos,main="Histogram URIs vs. Number of Memento",
+       xlab="Mementos Count",
+       border="blue",col="green",
+       las=1,breaks=5)
> 
```



```
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> currentMementosData = read.csv("C:\\ListOfMementos.csv")
> countsForMementos <- table(currentMementosData$MementosCount)
> barplot(countsForMementos,log="y", type='h', lwd=10, lend=2)
```

Question 3

Estimate the age of each of the 1000 URIs using the "Carbon Date" tool. Not all URIs will have Mementos, and not all URIs will have an estimated creation date. Show how many fall into either categories. For example, total URIs,no mementos, no date estimate.

1. Started the problem by instaling docker and using powershell command then installed the carbon date locally with docker image pull oduwsdl/carbondate
2. The next step was to run the carbon date locally so used docker container - docker container run –rm -it -p 8888:8888 oduwsdl/carbondate ./main.py -s
3. Once set up it runs on the localhost as http://localhost:8888/
4. Similar to the second problem solution we have getCarbonDate method which accepts the uri and returns the estimated-creation-date.
5. The getCarbonDate method takes the uri and appends it to the getCarbonDateUrlPrefix which will be http://localhost:8888/cd/.
6. Then a GET request is made wiht the resultant urls. The response is read a json object adn the estimated-creation-date is retrieved.
7. The main script as shown below reads the list of unique urls form the first problems output and sends the url to the getCarbonDate method. The returned estimated-creation-date is wriiten into a csv.

Listing 5: Python Script

```
import urllib.request
import json
import csv

urlCounter = 0
def getCarbonDate(uri):
    try:
        getCarbonDateUrlPrefix = "http://localhost:8888/cd/"
        contents = urllib.request.urlopen(getCarbonDateUrlPrefix + uri).
            read()
        #print(contents)
        jsonContents = json.loads(contents)
        responseCarbonDate = jsonContents["estimated-creation-date"]
        print(responseCarbonDate)
        return responseCarbonDate
    except:
        return "No Estimate Creation Date"

with open("C:\ListOfCarbonDates.csv", "w", newline='') as
    csvFileListOfMementosTemp:
    writer = csv.writer(csvFileListOfMementosTemp)
    currentData = ['URL','Estimate Creation Date']
    writer.writerow(currentData)
    for currentUrl in open("C:\ListOfUniqueUrls.txt", "r"):
        urlCounter = urlCounter + 1
        currentCarbonDate = getCarbonDate(currentUrl)
        currentData = [currentUrl,currentCarbonDate]
        writer.writerow(currentData)
        print(urlCounter)
```

Question 3 continued on next page. . .

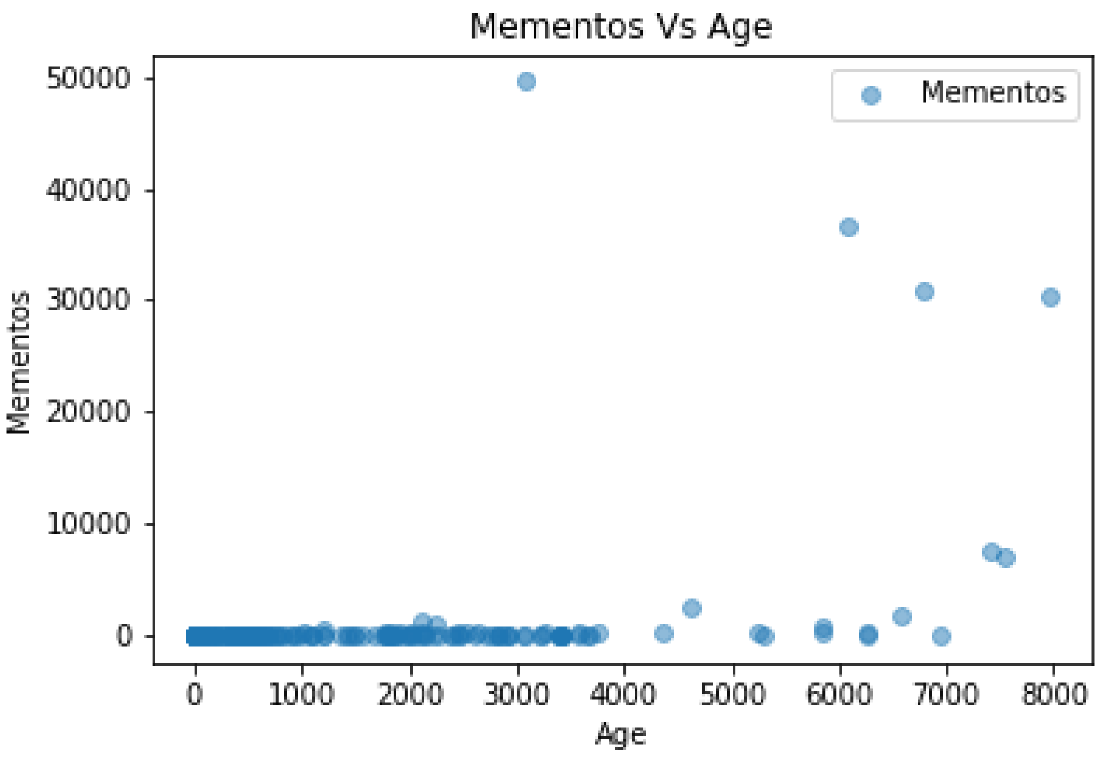
```
csvFileListOfMementosTemp.close()
print("Completed")
```

1. List of Unique URIs : 1054
2. No Mementos : 772
3. No Date Estimate : 208

Listing 6: Python Script

```
import matplotlib.pyplot as plt
import csv

x = []
y = []

with open('C:\ListOfAgeAndMemento.csv','r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    line_count = 0
    for row in plots:
        if line_count == 0:
                line_count += 1
        else:
            x.append(int(row[2]))
            y.append(int(row[1]))
            line_count += 1
plt.axis([1,8000,1,50000])
plt.scatter(x,y, label='Mementos',alpha=0.5)
plt.xlabel('Age')
plt.ylabel('Mementos')
plt.title('Mementos Vs Age')
plt.legend()
plt.show()
```

Question 3 continued on next page. . .


Mementos Vs Age
Mementos
Mementos
Age
0
10000
20000
30000
40000
50000
0
1000
2000
3000
4000
5000
6000
7000
8000