Ryan Condotta

Final Report Assignment 2

Question 1:

In order to solve the problem of extracting 1000 unique links from twitter. I had to do several steps with twitter before I even began writing a python program to help solve the problem. I first had to begin by registering a new application with my twitter account so I could have something to work with. Registering a new application on twitter will allow me to use python to access my twitter features including home timeline with an authentication token and secret and consumer key and secret represented as variables: ckey, csecret, atoken, and asecret. Now, I could begin my python programming by authenticating the variables with the twitter_api in order to access my twitter features. Then, I used the function home_timeline with a count of 10000 to get an abundant of tweets which are stored in statuses as shown below:

```
auth = twitter.oauth.OAuth(atoken, asecret, ckey, csecret)
twitter_api = twitter.Twitter(auth=auth)
count = 10000
statuses = twitter_api.statuses.home_timeline(count = 10000)
```

Then once I have the information in a list, I search the tweets for the expanded url of the tweet so that it would cut down on getting the same link. I return this information to the list of url_links as shown below:

```
# get url data into lists
url_links = [ user_mention['expanded_url']
            for status in statuses
               for user_mention in status['entities']['urls'] ]
```

Next, from the extracted url links, I store the information in a new list as shown below to make sure that the information is unique in the new list. So, I search the list for the link and if it's there then I don't add the link to the list and if it's not then I added it in the code below:

```
list = []
listcount = 0
for links in url_links:
        if links in list:
                pass
        else:
                list.append(links)
```

Then, if the new list is less than 1000, I would use beautiful soup to extract more links from the tweets urls in order to reach 1000. This process is defined with the pseudocode below for simplicity:

```
While length of list is less than 1000
        Get beautiful soup response
                Find href links
                        Add links to list if less than 1000 and not in unique list
```

This is a simplified version of the pseudo code but accomplishes the task. It will get the 1000 unique links into the list if it needs any more links. Lastly, the list of links will be outputted to a file called output.txt and displayed as:

http://bit.ly/1KpXwoX
http://youtu.be/9KVxBSgxgkI
https://twitter.com/30for30/status/695431822293860352
https://vine.co/v/iJTll5DpHaW
https://twitter.com/soundersfc/status/695430059994132480
http://nyti.ms/1PDZHpm
http://bit.ly/1PWmqz0
http://m.uscles.com/SH6Lx
https://twitter.com/dallasstars/status/695429548968513541
http://bit.ly/1SJ0DcY

This accomplished the task presented to me. However, I did have to brush up some more on my python and certain features such as outputting to a file in order to accomplish this task.

Question 2:

The next problem given to me was to get the unique list of links from problem 1 in output.txt and get the mementos of the links if they have one, then present the information in a histogram of URIs vs mementos. The first process was to make a python program that would take the links from output.txt and then use odu memento aggregator to get the number of mementos for each of the links. The process begins by taking a link, one at a time, from the output.txt and get the html response from the odu memento aggregator. I would write this response to output1.txt so that I could individually search the output1.txt file for the word memento. Each time the word memento is in the file, I count it and place the link in one list and the total in another list. I also added a third list of uri and count that was outputted to a file named output2.txt.

```
with open(filename1, "r") as file:
    for line in file:
        url = "http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/" + line
        value = value +1
        try:
            response = urllib.request.urlopen(url)
            soup = BeautifulSoup(response, 'html.parser')
            filename = r"C:\Users\Ryan\Documents\WebScience\Assignment2\output1.txt"
            outfile = open(filename, 'w')
            outfile.write(str(soup))
            outfile.close()
            print("Here:" , value)
            count = 0
            with open(filename, "r") as file:
                for line1 in file:
                    if line1.find('memento'):
                        count = count + 1
            print("Count is:", count)
```

```
                newlist.append(line + " " +str(count))
                excellist1.append(line)
                excellist2.append(count)
        except:
                print(value)
                newlist.append(line+ ' 0')
                excellist1.append(line)
                excellist2.append(0)
                pass
```

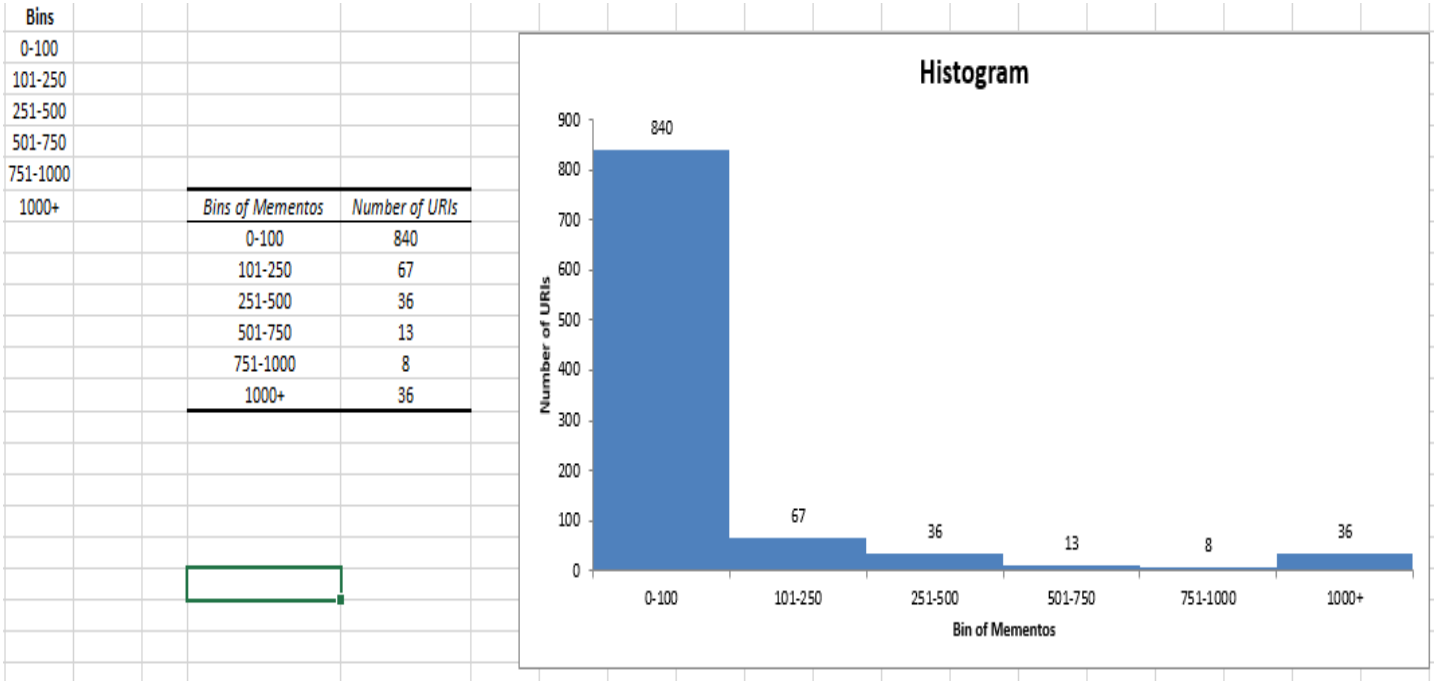
The output2.txt file looked like the structure below until all of the links and there corresponding number of mementos were listed.

```
http://bit.ly/1KpXwoX
 0
http://youtu.be/9KVxBSgxgkI
 0
```

For simplicity of the histogram, I used three different lists so I could output the information to an excel file because it has a built in histogram that I could exploit and was familiar with using. The python code shown below would create a new file with a worksheet that displays two columns with titles of URI and Num of Mementos with all of the listing data below each of the columns. This would help me to exploit the histogram ability by using the output ability from python.

```
#excel Worksheet
workbook = xlsxwriter.Workbook(filename2)
worksheet = workbook.add_worksheet()

bold = workbook.add_format({'bold' : 1})

#data headers
worksheet.write('A1', 'URI', bold)
worksheet.write('B1', 'Num of Mementos', bold)

row = 1
col = 0

#throw data into
for URI in excellist1:
        worksheet.write_string(row, col, URI)
        row += 1

row1 = 1
col1 = 1
for Number in excellist2:
        worksheet.write_number(row1, col1, Number)
        row1 += 1
workbook.close()
```

From the excel file, I had to create the histogram. Since I had the data in the excel file from the python file, I just had to create the bins for the graph which were 0-100, 101-250, 251-500, 501-750, 751-1000, and 1000+. Lastly, I used the histogram ability from the data analysis section in excel and defined the data and the different bins which created the histogram as shown in the graph below from excel.



This accomplished the task of creating the histogram and getting the mementos for each of the links from problem 1. I had to look up the ability to output to an excel file as well as brush up the histogram ability to accomplish this task but when I did, it made it very simple to accomplish the tasks.

Question 3:

To solve the problem of using the carbon dating tool to get the creation date of each of the links, I first tried to download the python script that would make it able to use the tool. However, the functionality of the python code was not compatible with my computer and kept giving me errors based on the link. So, instead I used the web service in a unique way. I created the python script A2Q3.py to help solve this matter. It has four main functions which were: storeMementosandURIs, CurlForReponse, GetCreationDate, and outputExcel. storeMementosandUris works exactly how it sounds, it opens output2.txt which has the uris and the number of mementos and stores those pieces of information into lists named urls and mementos. This was also the easiest point to calculate the number of urls with 0 momentos which was as simple as:

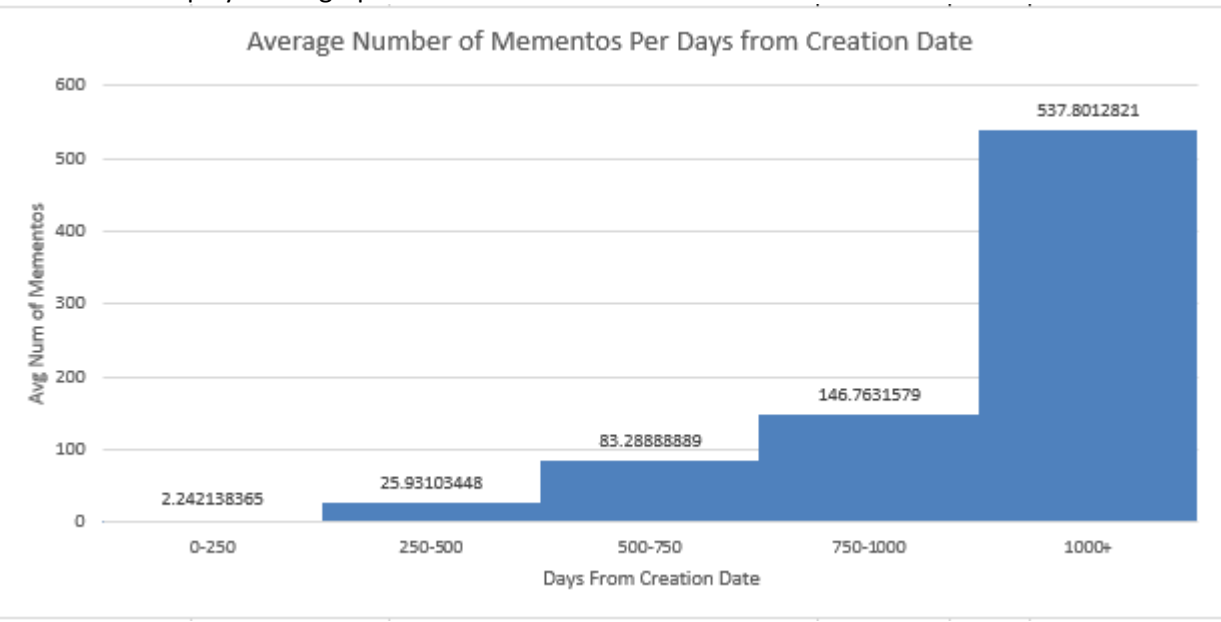
```
count = 0
        for moment in mementos:
                if moment == 0:
                        count = count + 1
```

Then, I called the CurlForResponse function whenever the urls and mementos were stored in there respected lists. This function would use a curl GET and plug in the url into the search parameter and then I would use beautiful soup to get the html of the page as shown:

```
filename = r"http://cd.cs.odu.edu/cd?url=" + line
        print(index+1)
        try:
                response = urllib.request.urlopen(filename)
                soup = BeautifulSoup(response, 'html.parser')
```

I would then write the soup html response to a file named CarbonDate.txt and then call the CreationDate function to retrieve the creation date for the url. When the CreationDate function was called, it would open the CarbonDate.txt file and go to the 4th line of Estimated Creation Date and retrieve the date of the creation date in year-month-day format and store it in a list named date. If it did not have a creation date it would return a 1 that would be added onto a counter for the number of uris without a creation date. Once all the values were stored, it was time to easily output them to Excel using the same process from Question 2 on page 3 of this report. This was the easiest way for me to go about solving the problem and storing each list of values into excel in order to create a graph of days vs mementos.

Once in Excel, I had some work to do. I first calculated the number of days from the current date to get the difference from the creation date. Then, I counted the number of values in each of the bins just to get a feel for what I was working with. Lastly, I calculated the average of mementos for each of the bins and displayed the graph as shown below:



The graph shows that as the number of days increases then the number of mementos generally increases as well which makes logical sense. The use of excel to manipulated and calculate data was handy when it came to this problem and helped me solve it efficiently and effectively.