OLD DOMINION UNIVERSITY

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 Thursdays 4:20pm – 7:10pm ECSB 2120

Assignment # 1
Joseph Elder UIN: 00844802

Honor Pledge

I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned

September 25, 2014

Contents

1.1 Question 1

1.1.1 The Problem:

Write a Python program that extracts 1000 unique links from

witter. 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 different

shortened URIs for www.cnn.com. For example, http://cnn.it/1cTNZ3V

http://t.co/BiYdsGotTd

Both ultimately redirect to cnn.com, so they count as only 1 unique URI.

Also note the second URI redirects twice -- don't stop at the first

redirect.

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.

1.1.2 The Solution:

The 1000 unique links can be found in `links.txt`. The Twitter Link Extractor (`TwitterLinkExtractor.py`) is written in python version 2.7.6 . The purpose of the Twitter Link Extractor is to collect 1000 Unique URI's from tweets from users who a specified user is following and print them to a text file to be used later to gather

data. The specified user for this assignment will be my twitter account “@up5free”. This program uses the python modules tweepy, requests, and re.

Information on twitter is accessed with the Tweepy module in python. A user is selected using the Tweepy API. Tweepy allows access to the full list of Twitter friends who the selected user is following on Twitter. The Tweepy API function “`user_timeline()`” gets tweets for each of the users being followed. The extractor program grabs the 128 most recent tweets from all users the selected user is following to be processed.

Each tweet for each user is then processed until 1000 unique URI's are found. The text is searched for the URIs with the regular expression module. When one is found a web request is then perfomed on it using the requests python module. The status code of the website is checked and the URI is followed to it's destination sometimes through multiple redirections. Only sites with a status code of 200 which have unique web addresses are allowed into the link collection.

1.2 Question 2

1.2.1 The Problem:

Download the TimeMaps for each of the target URIs. We'll use the mementoweb.org Aggregator, so for example:

URI-R = http://www.cs.odu.edu/

URI-T = http://mementoweb.org/timemap/link/http://www.cs.odu.edu/

You could use the cs.odu.edu aggregator:

URI-T = http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/http://www.cs.odu.edu/

But be sure to say which aggregator you use -- they are likely to give different answers.

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.

See: http://en.wikipedia.org/wiki/Histogram

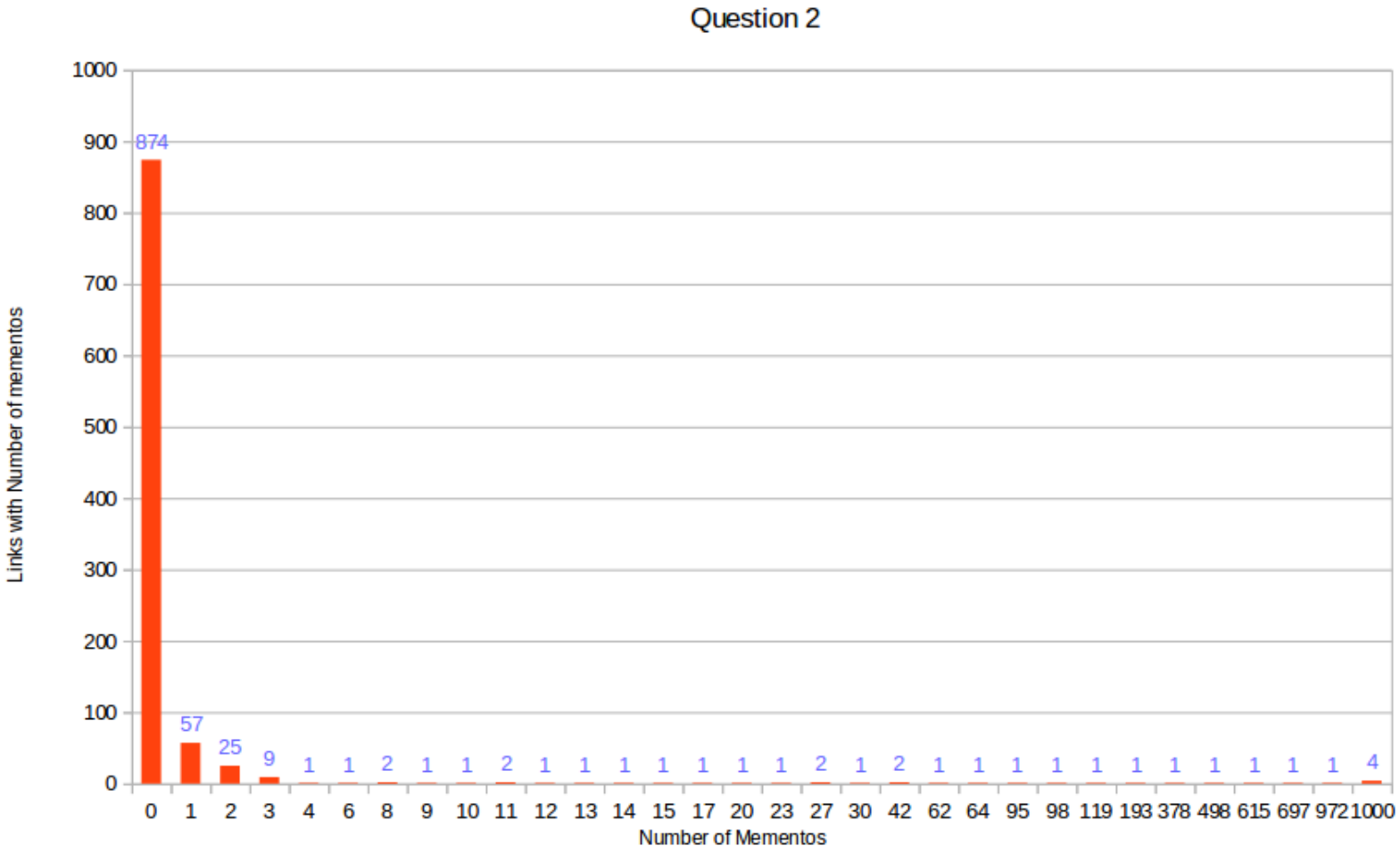
Note that the TimeMaps can span multiple pages. Look for links like:

<http://mementoweb.org/timemap/link/1000/http://www.cnn.com/>;rel="timemap"; type="application/link-format"; from ="Sun, 08 Jul 2001 21:30:54 GMT"

This indicates another page of the TimeMap is available. There can be many pages to a TimeMap.

1.2.2 The Solution:

The Twitter Link Extractor is also used to download Timemaps for the 1000 unique target URIs. These Timemaps tell whether or not a specific website has been archived. The Twitter extractor makes web requests from python to use the MementoWeb.org aggregator located at http://mementoweb.org/ A memento exists for every instance of that site being archived. For all 1000 links the number of mementos is counted and written to the file `links.txt` along with the original URI, its status code, and the destination URI.



The results recorded in `links.txt` are opened by another python file, `StatsReporting.py`. `StatsReporting.py` serves two purposes fulfilled by two separate functions. The first purpose and function exist to generate data to a text file `two.txt`. The data recorded in `two.txt` is used to generate the histogram above in the figure above. The histogram shows on the X-axis the Number of Mementos found for URIs. The Y-axis of the histogram shows the frequency of sites with that Number of Mementos listed on the X-axis. The large majority of URIs found on Twitter for this particular data set lacked any mementos at all, with 874 of the links coming having 0 mementos. The second greatest number of mementos found for URIs was 1 memento, with the third greatest being 2 mementos.

1.3 Question 3

1.3.1 The Problem:

Estimate the age of each of the 1000 URIs using the "Carbon Date" tool:

http://ws-dl.blogspot.com/2013/04/2013-04-19-carbon-dating-web.html

Note: you'll have better luck downloading and installing the tool

rather than using the web service (which will run slowly and likely be unreliable).

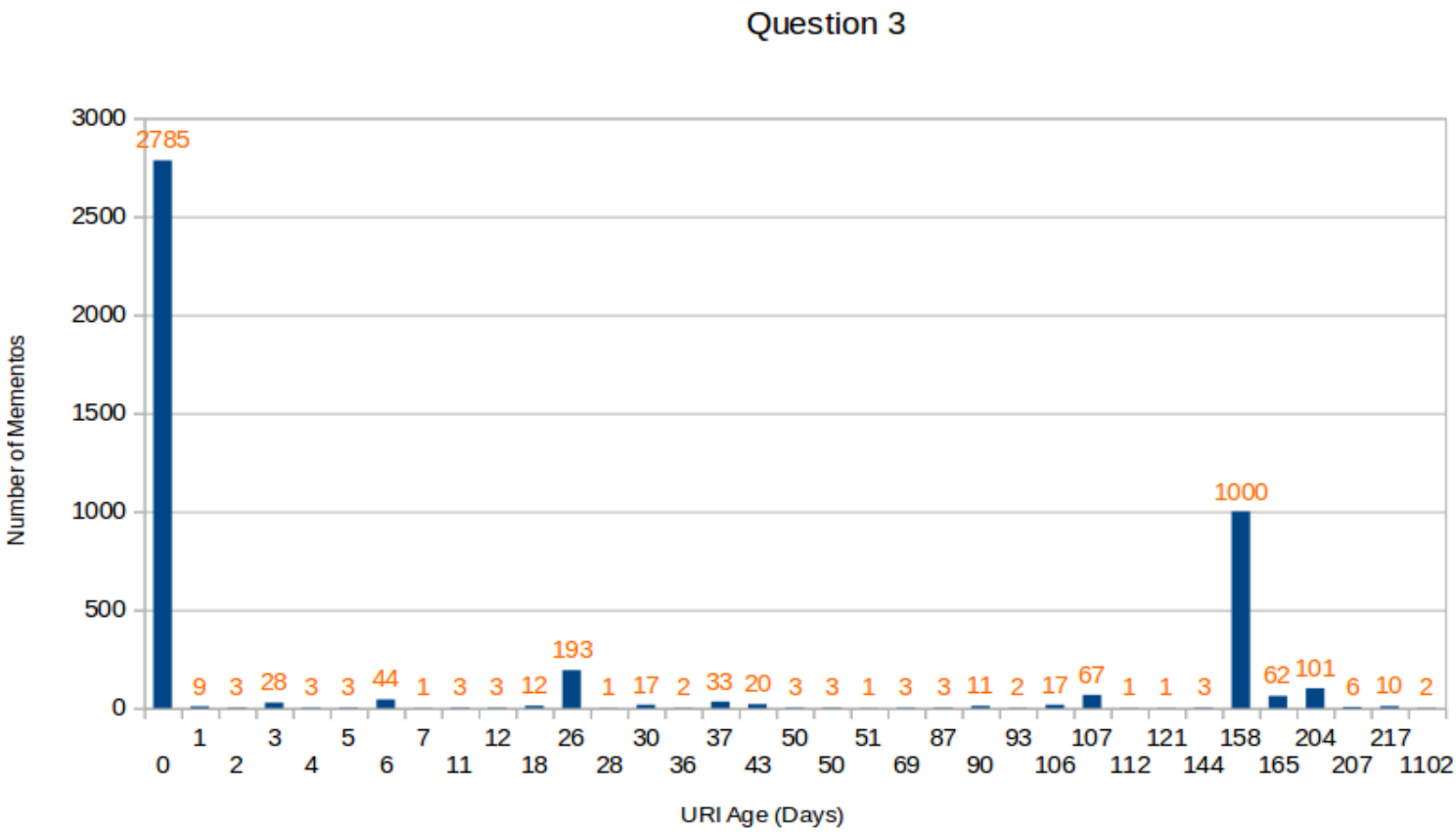
For URIs that have > 0 Mementos and an estimated creation date,

create a graph with age (in days) on one axis and number of mementos

on the other.

1.3.2 The Solution:

The Twitter Link Date Test (`TwitterLinkDateTest.py`) is to estimate the creation date of each of the 1000 links using Carbon Date. The program writes its results to the text file `DateEstimation.txt`. After downloading the source code of the Carbon Date tool from github and configuring Carbon Date as well as its components, `TwitterLinkDateTest.py` must be placed in the master Carbon Date folder by default named CarbonDate-master. The CarbonDate-master folder also contains the python file `local.py`.



Carbon Date is run by supplying the “cd” function which is located in `local.py`. `TwitterLinkDateTest.py` imports `local.py` to make use of the cd function. The cd function is called for each URI the output is handled getting the estimated creation date and printing It and the URI to `DateEstimation.txt`. The second function of `StatsReporting.py` opens the `DateEstimation.txt` file in addition to the `links.txt` file in order to determine if a URI has more than 0 mementos and to determine if that URI with mementos has an estimated creation date. If both conditions are met the URI, its creation date, and the number of mementos found for that particular URI are written to the text file `three.txt`. The information found in file three.txt is used to generated the graph above showing the number of mementos for a URI based on the URI's age in days.

References

1. https://www.python.org/doc/
2. http://docs.tweepy.org/en/latest/
3. http://www.google.com/
4. http://ws-dl.blogspot.com/2013/04/2013-04-19-carbon-dating-web.html
5. http://docs.python-requests.org/en/latest/
6. https://docs.python.org/2/library/re.html