Assignment 2

Dr. Nwala

Nathaniel Everett

February 11, 2018

Contents:

Problem 1. Write a Python program that extracts 1000 unique links from Twitter. Omit links from the Twitter domain (twitter.com). You might want to take a look at:

https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/
http://adilmoujahid.com/posts/2014/07/twitter-analytics/

see also:

http://docs.tweepy.org/en/v3.5.0/index.html
https://github.com/bear/python-twitter
https://dev.twitter.com/rest/public

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.). For example:

```
$ curl -IL --silent https://t.co/DpO767Md1v | egrep -i "(HTTP/1.1|^location:)"
HTTP/1.1 301 Moved Permanently
location: https://goo.gl/40yQo2
HTTP/1.1 301 Moved Permanently
Location: https://soundcloud.com/roanoketimes/ep-95-talking-hokies-recruiting-one-week-before-
signing-day
HTTP/1.1 200 OK
```

You might want to use the streaming or search feature to find URIs. If you find something inappropriate for any reason you see fit, just discard it and get some more links. We just want 1000 links that were shared via Twitter.

Hold on to this collection and upload it to github -- we'll use it later throughout the semester.

Solution: Part 1 of this assignment is done in twt.py, modified from the instructor's code. It streams tweets of a certain filter (in this example, football) and returns the links found. The links will be put into a file named linklists.txt. The code was run multiple times until there were 1000 unique links. All duplicate links were manually searched and removed. linklists.txt is in the Output folder, while twt.py is in the Programs folder.

Problem 2. 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://memgator.cs.odu.edu/timemap/link/http://www.cs.odu.edu/

or:

URI-T = http://memgator.cs.odu.edu/timemap/json/http://www.cs.odu.edu/

(depending on which format you'd prefer to parse)

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. The x-axis
will have the number of mementos, and the y-axis will have the
frequency of occurence.

* = https://en.wikipedia.org/wiki/Histogram

What's a TimeMap?
See: http://www.mementoweb.org/guide/quick-intro/
And the week 4 lecture.

Solution: Following Problem 1, another python program, timemaps.py, opens linklists.txt and appends "http://memgator.cs.odu.edu/timemap/json/" to each URI listed in linklists.txt. The output is stored into a different text file, timemaps.txt. A third python program, mementos.py, parses the text in each link from timemaps.txt and puts it into a json file, mementos.json. In my file, I ran into a number of links that ended up being 404s, these were assumed to have no mementos. Using a json viewer, the links that did not result in 404s had their mementos manually counted, and using an R program, MementosHistogram.r, they were all shown in a histogram. Given the high amount of links with 0 mementos in them, the count for the 0 mark is very high and goes well above the bounds that the other marks have. I scaled the histogram so that the other values shown were more displayable. The histogram is saved to a pdf file, MementosHistogram.pdf, in the Output folder, while the mementos.py, mementos.json, timemaps.py, and timemaps.txt files are all in the Programs folder.

Problem 3. Estimate the age of each of the 1000 URIs using the "Carbon
Date" tool:

http://ws-dl.blogspot.com/2017/09/2017-09-19-carbon-dating-web-version-40.html

Note: you should use "docker" and install it locally. You can do
it like this:

http://cd.cs.odu.edu/cd?url=http://www.cs.odu.edu/

But it will inevitably crash when everyone tries to use it at the
last minute.

For URIs that have > 0 Mementos and an estimated creation date,
create a graph with age (in days) on the x-axis and number of
mementos on the y-axis.

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:        1000
no mementos:        137
no date estimate:   212
```

Solution: For the age of the site, since I was ultimately unable to get Docker running on my machine, I decided to get the age manually. This involved using the cd.cs.odu.edu link. Considering that I didn't have many links to work with from mementos.json, this did not take as long as expected, but a large json file may need some additional programming. The program scatterplot-date.r was used to plot the estimated ages of the sites in accordance with the number of mementos. The plot has the age calculated in days as the x-axis and the number of mementos from the y-axis, not counting those that have 0 mementos. There is also console output for three categories, the total number of URIs (assuming 1000 by my calculations), the ones with 0 mementos, and the ones without a date estimate. The Output folder therefore contains the histogram, scatterplot-date.pdf, and scatterplot-date.r is in the Programs folder.

Files included:
MementosHistogram.r (R program for histogram)
MementosHistogram.pdf (histogram)
scatterplot-date.r (R program for age of sites)
scatterplot-date.pdf (scatterplot)
mementos.json (output of mementos.py, used for both R programs)
listlinks.txt (output of twt.py, input of timemaps.py)
twt.py (modified python program, extracts links from tweets, omitting all twitter.com domain links, may need to run multiple times to get what is necessary)
timemaps.py (python program to be run after twt.py, turns links into memgator links for ODU's memento aggregator, output to timemaps.txt)
timemaps.txt (output of timemaps.py, input of mementos.py)
mementos.py (python program to be run after timemaps.py, extracts json information from each link to be stored in mementos.json)