Assignment 2

Fall 2014
CS595 Web Science
Dr. Michael Nelson

Mathew Chaney

September 25, 2014

Contents

List of Figures

Listings

1 Question 1

1.1 Question

Write a Python program that extracts 1000 unique links from Twitter. 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.

1.2 Resources

- Getting Started with Twitter API: http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/
- Twitter Search API: https://dev.twitter.com/rest/public/search
- Twitter API - Get / Search Tweets: https://dev.twitter.com/rest/reference/get/search/tweets

1.3 Answer

Using the python module requests made this task a breeze as well as the initial code provided by Thomas Sileo's blog post.

```
# -*- encoding: utf-8 -*-
import requests
from requests_oauthlib import OAuth1
from urllib import quote

REQUEST_TOKEN_URL = "https://api.twitter.com/oauth/request_token"
AUTHORIZE_URL = "https://api.twitter.com/oauth/authorize?oauth_token="
ACCESS_TOKEN_URL = "https://api.twitter.com/oauth/access_token"

CONSUMER_KEY = "PDBekXkvUto4V0XYZrrizcEub"
CONSUMER_SECRET = "0E1KNfpNWF8Eh4iwbDFYFKDMBSouni3uRZrpsoGhbJcLZZmnBq"

OAUTH_TOKEN = "2560074793-g7ES1sQmwl3YKAfCJnIBa0lh3wHLjmPqj96XFuV"
OAUTH_TOKEN_SECRET = "tGYCQa9LL2i6wmApJzbGzHdVIVA65xiwVffPmbqWJwZPs"

SEARCH_URI = "https://api.twitter.com/1.1/search/tweets.json?q="

SEARCH_ITEMS = map(quote, [ 'space x',
                            'elon musk',
```

```python
                            'richard garriott',
                            'starcraft 2',
                            'ebola virus',
                            'world cup',
                            'singularity',
                            'rick and morty',
                            'iphone 6',
                            'android',
                            'robin williams',
                            'tony stewart',
                            'bitcoin',
                            'game of thrones',
                            'facebook',
                            'youtube',
                            'google',
                            'chris roberts',
                            'hyper light drifter',
                            'golang'])

def get_oauth():
    return OAuth1(CONSUMER_KEY,
                client_secret=CONSUMER_SECRET,
                resource_owner_key=OAUTH_TOKEN,
                resource_owner_secret=OAUTH_TOKEN_SECRET)

def find_uris(uris):
    with open('output', 'a') as outfile:
        for search_item in SEARCH_ITEMS:
            result = requests.get(SEARCH_URI + search_item + '&filter%3Alinks&count=1000', auth=oauth)
            for status in result.json()['statuses']:
                for url in status['entities']['urls']:
                    if len(uris) == 1000:
                        return
                    if 'expanded_url' in url:
                        try:
                            result = requests.get(url['expanded_url'], timeout=4)
                            # only add expanded uris if they aren't in the list already
                            if result.status_code == 200 and result.url not in uris:
                                add_uri(uris, result.url)
                                outfile.write('%s\n' % result.url)
                        except Exception as e:
                            print e
                            continue

def add_uri(uris, uri):
    uris.add(uri)
    print 'added uri #%d: %s' % (len(uris), uri)

if __name__ == "__main__":
    oauth = get_oauth()
    uris = set()
    # read in previous set of uris
    try:
        with open('output', 'r') as infile:
            for line in infile.readlines():
                add_uri(uris, line.strip())
    except IOError:
        pass
    find_uris(uris)
```

Listing 1: urifinder.py

The script was run multiple times to get the desired 1000 unique URIs. It would end prematurely at times, so the data set was initialized with the data of the previous run and then passed on to the `find_uris` function to preserve work performed.

2 Question 2

2.1 Question

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.

2.2 Resources

- R: http://www.cs.odu.edu/~sainswor/uploads/Teaching/cs595f13-R.pdf

2.3 Answer

Using a simple python script to get the timemaps and then parse the results, traveling down the rabbit hole if the target uri has more than 1000 mementos, was done with the following code.

```
# -*- encoding: utf-8 -*-
#! /usr/bin/python

import requests
import re

MW_URI = "http://mementoweb.org/timemap/link/"

if __name__ == '__main__':
    with open('output', 'r') as f:
        output = open('results', 'w')
        mementos = {}
        for uri in f.read().split('\n'):
            if uri is '':
                continue
            count = 0
            target_uri = MW_URI + uri
            while True:
                result = requests.get(target_uri)
                if result.ok:
                    count = count + result.text.count('rel="memento"')
                last_line = result.text.split('\n')[-1]
                if 'rel="timemap"' not in last_line:
                    break
                sites = re.findall(r'<([^<|^>]+)>', last_line)
                target_uri = sites[1]
            mementos[uri] = count
            print 'found %d mementos for uri: %s' % (count, uri)
            output.write('%s %d\n' % (uri, count))
    output.close()
```

Listing 2: mementofinder.py

And the results of running the python code in Listing 2. A log scale was used along the y axis to show more detail among the results.

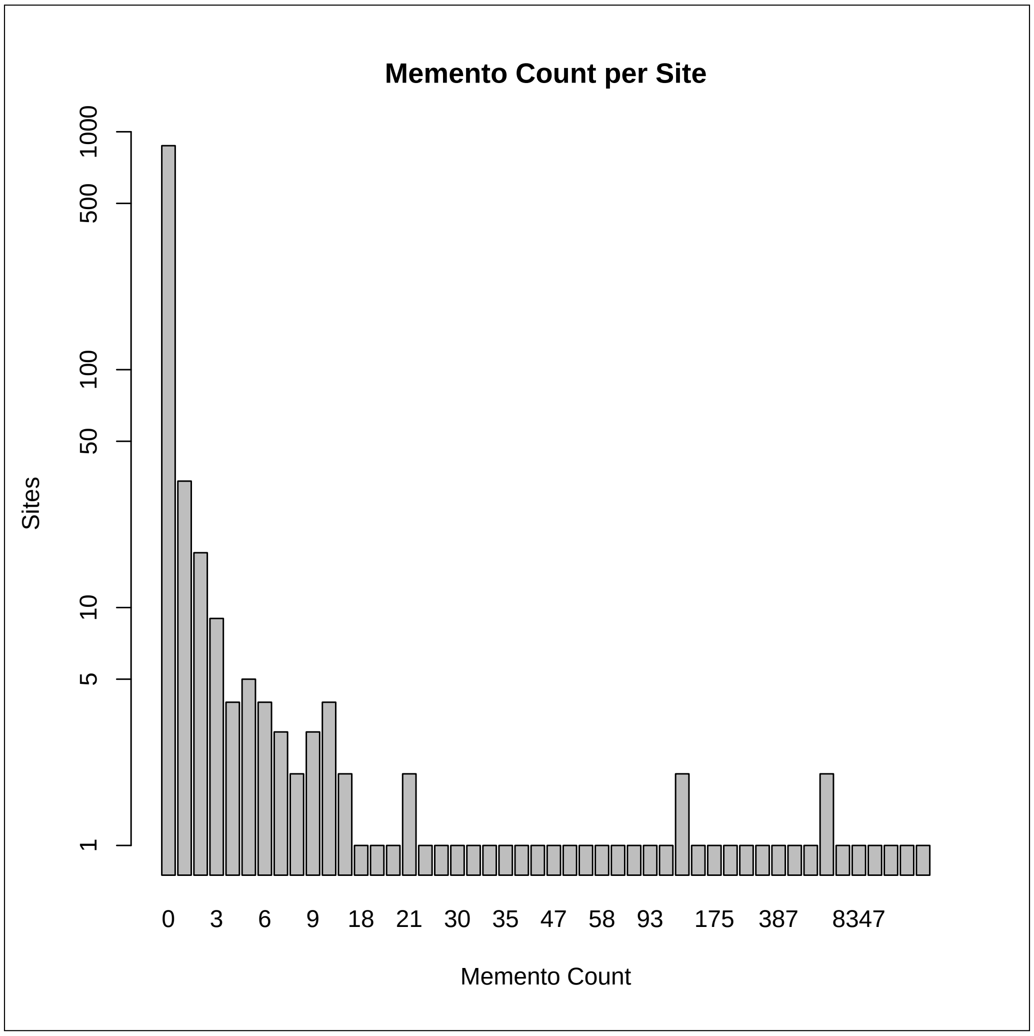

Figure 1: Histogram of Site Mementos

3 Question 3

3.1 Question

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.

3.2 Resources

- None: yet

3.3 Answer

Using the python script in Listing 3, along with the “Carbon Date” tool provided by H.S. Eldeen, estimated creation dates were found for 843 of the original URIs.

```
#! /usr/bin/python

from local import cd
import json

ECD = 'Estimated Creation Date'

def getdate(uri):
    uri_json = json.loads(cd(uri))
    if uri_json[ECD]:
        print 'Found creation date: %s' % uri_json[ECD]
        outfile.write('%s %s\n' % (uri, uri_json[ECD]))
    else:
        print 'Found no ECD'

if __name__ == '__main__':
    # Remove already completed links
    with open('output', 'r') as outfile:
        output = [line.rstrip('\n') for line in outfile]
    with open('results', 'r') as prevfile:
        prev = [line.split(' ')[0] for line in prevfile]
    output = [line for line in output if line not in prev]
    print "Starting on uri #%d" % len(output)

    # Work on the rest
    with open('output', 'r') as infile:
        with open('results','a') as outfile:
            for uri in infile:
                uri = uri.strip()
                print 'Searching for uri: %s' % uri
                getdate(uri)
```

Listing 3: carbondate.py