Assignment 2

Fall 2014
CS595 Web Science
Dr. Michael Nelson

Mathew Chaney

September 19, 2014

Contents

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.

```
1  # −∗− encoding: utf−8 −∗−
2  import requests
3  from requests_oauthlib import OAuth1
4  from urllib import quote
5
6  REQUEST_TOKEN_URL = "https://api.twitter.com/oauth/request_token"
7  AUTHORIZE_URL = "https://api.twitter.com/oauth/authorize?oauth_token="
8  ACCESS_TOKEN_URL = "https://api.twitter.com/oauth/access_token"
9
10 CONSUMER_KEY = "PDBekXkvUto4V0XYZrrizcEub"
11 CONSUMER_SECRET = "OElKNfpNWF8Eh4iwbDFYFKDMBSouni3uRZrpsoGhbJcLZZmnBq"
12
13 OAUTH_TOKEN = "2560074793-g7ESlsQmwl3YKAfCJnIBa0lh3wHLjmPqj96XFuV"
14 OAUTH_TOKEN_SECRET = "tGYCQa9LL2i6wmApJzbGzHdVIVA65xiwVffPmbqWJwZPs"
15
16 SEARCH_URI = "https://api.twitter.com/1.1/search/tweets.json?q="
17
18 SEARCH_ITEMS = map(quote, [ 'space x',
19                            'elon musk',
```

```python
20                                    'richard garriott',
21                                    'starcraft 2',
22                                    'ebola virus',
23                                    'world cup',
24                                    'singularity',
25                                    'rick and morty',
26                                    'iphone 6',
27                                    'android',
28                                    'robin williams',
29                                    'tony stewart',
30                                    'bitcoin',
31                                    'game of thrones',
32                                    'facebook',
33                                    'youtube',
34                                    'google',
35                                    'chris roberts',
36                                    'hyper light drifter',
37                                    'golang'])
38
39 def get_oauth():
40     return OAuth1(CONSUMER_KEY,
41                   client_secret=CONSUMER_SECRET,
42                   resource_owner_key=OAUTH_TOKEN,
43                   resource_owner_secret=OAUTH_TOKEN_SECRET)
44
45 def find_uris(uris):
46     for search_item in SEARCH_ITEMS:
47         result = requests.get(SEARCH_URI + search_item + '&filter%3Alinks&count=1000', auth=
                oauth)
48         for status in result.json()['statuses']:
49             for url in status['entities']['urls']:
50                 if len(uris) == 1000:
51                     return
52                 if 'expanded_url' in url:
53                     try:
54                         result = requests.get(url['expanded_url'])
55                         # only add expanded uris if they aren't in the list already
56                         if result.status_code == 200 and result.url not in uris:
57                             add_uri(uris, result.url)
58                     except Exception as e:
59                         print e
60                         continue
61
62 def add_uri(uris, uri):
63     uris.add(uri)
64     print 'added uri #%d: %s' % (len(uris), uri)
65
66 if __name__ == "__main__":
67     oauth = get_oauth()
68     uris = set()
69     # read in previous set of uris
70     with open('output', 'r') as infile:
71         for line in infile.readlines():
72             add_uri(uris, line.strip())
73     find_uris(uris)
74     with open('output', 'w') as outfile:
75         outfile.writelines('%s\n' % uri for uri in 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.

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

import requests
import pickle

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

if __name__ == '__main__':
    with open('output', 'r') as f:
        mementos = {}
        for uri in f.read().split('\n'):
            if uri is '':
                continue
            count = 0
            result = requests.get(MW_URI + uri)
            if result.ok:
                count = result.text.count('rel="memento"')
            mementos[uri] = count
            print 'found %d mementos for uri: %s' % (count, uri)
    with open('results', 'w') as o:
```

```
22          for item, num in mementos.iteritems():
23              o.write('%s %d\n' % (item, num))
```

Listing 2: mementofinder.py