CS595 Intro to Web Science, Assignment #2

Valentina Neblitt-Jones

September 19, 2013

1 Link Extraction from Twitter Exercise

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. You might want to use the search feature (Figure 1) 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 through the semester.**

The Code

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
import requests
from requests_oauthlib import OAuth1
from urlparse import parse_qs

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 = "YOUR_CONSUMER_KEY"
CONSUMER_SECRET = "YOUR CONSUMER_SECRET"

OAUTH_TOKEN = ""
OAUTH_TOKEN_SECRET = ""


def setup_oauth():
    """Authorize your app via identifier."""
    # Request token
    oauth = OAuth1(CONSUMER_KEY, client_secret=CONSUMER_SECRET)
    r = requests.post(url=REQUEST_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)

    resource_owner_key = credentials.get('oauth_token')[0]
    resource_owner_secret = credentials.get('oauth_token_secret')[0]

    # Authorize
```

```
    authorize_url = AUTHORIZE_URL + resource_owner_key
    print 'Please go here and authorize: ' + authorize_url

    verifier = raw_input('Please input the verifier: ')
    oauth = OAuth1(CONSUMER_KEY,
                   client_secret=CONSUMER_SECRET,
                   resource_owner_key=resource_owner_key,
                   resource_owner_secret=resource_owner_secret,
                   verifier=verifier)

    # Finally, Obtain the Access Token
    r = requests.post(url=ACCESS_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)
    token = credentials.get('oauth_token')[0]
    secret = credentials.get('oauth_token_secret')[0]

    return token, secret


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

if __name__ == "__main__":
    if not OAUTH_TOKEN:
        token, secret = setup_oauth()
        print "OAUTH_TOKEN: " + token
        print "OAUTH_TOKEN_SECRET: " + secret
        print
    else:
        oauth = get_oauth()
        r = requests.get(url="https://api.twitter.com/1.1/statuses/
            mentions_timeline.json", auth=oauth)
        print r.json()
```

The Execution

Figure 1:

2 TimeMaps Exercise

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://mementoproxy.cs.odu.edu/aggr/timemap/link/http://www.cs.odu.edu/`

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.

3 Carbon Date Exercise

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 to download and install; don't try to use the web service. For URIs that have >0 Mementos and an estimated date, create a graph with age (in days) on one axis and number of Mementos on the other.**