CS532 Web Science: Assignment 3

Finished on February 18, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

Listings

List of Figures

Problem 1

Question

Download the 1000 URIs from assignment #2. "curl", "wget", or "lynx" are all good candidate programs to use. We want just the raw HTML, not the images, stylesheets, etc.

from the command line:

% curl http://www.cnn.com/ > www.cnn.com

% wget -O www.cnn.com http://www.cnn.com/

% lynx -source http://www.cnn.com/ > www.cnn.com

"www.cnn.com" is just an example output file name, keep in mind that the shell will not like some of the characters that can occur in URIs (e.g., "?", "&"). You might want to hash the URIs, like:

% echo -n "http://www.cs.odu.edu/show_features.shtml?72" |md5
41d5f125d13b4bb554e6e31b6b591eeb

("md5sum" on some machines; note the "-n" in echo - this removes the trailing newline.)

Now use a tool to remove (most) of the HTML markup. "lynx" will do a fair job:

% lynx -dump -force_html www.cnn.com > www.cnn.com.processed
Use another (better) tool if you know of one. Keep both files for each URI (i.e., raw HTML and processed).

Answer

Using the python script in Listing 1, 1000 unique URIs were dereferenced and their raw contents were stored in the `html/raw/` folder as a file with the filename as the md5-hashed URI. These were then stripped of all html elements and their processed contents were stored in the `html/processed/` folder as the same md5-hashed filename. For reference, the URIs were written as the first line of each of their content files.

```
1 #! /usr/bin/python

  import requests
  import concurrent.futures
  import md5
6 from bs4 import BeautifulSoup
  import pickle

  def convert(uri):
```

```
    return md5.new(uri).hexdigest()

def get_html(uri):
    print('Getting {}'.format(uri))
    response = requests.get(uri)
    return response.url, response.status_code, response.content

if __name__ == '__main__':
    with open('links') as infile:
        uris = [uri.rstrip('\n') for uri in infile]

    with concurrent.futures.ThreadPoolExecutor(max_workers=8) as executor:
        uri_futures = [executor.submit(get_html, uri) for uri in uris]
        for future in concurrent.futures.as_completed(uri_futures):
            try:
                uri, status_code, content = future.result()
            except Exception as exc:
                print('{} generated an exception: {}'.format(uri, exc))
                continue
            if status_code == 200:
                hashed_uri = convert(uri)
                print('Writing {} as {}'.format(uri, hashed_uri))
                try:
                    with open('html/raw/' + hashed_uri, 'w') as outfile:
                        outfile.write(uri + '\n')
                        outfile.write(content)
                    with open('html/processed/' + hashed_uri + '.processed.txt', 'w') as
                        outfile:
                        outfile.write(uri + '\n')
                        outfile.write(BeautifulSoup(content).get_text().encode('utf8'))
                except Exception as e:
                    print '**** ERROR **** --- ' + uri
                    print e
            else:
                print('Not writing {}, bad status code: {}'.format(uri, status_code))
```

Listing 1: get_html.py