Assignment 2

Fall 2014
CS595 Web Science
Dr. Michael Nelson

Mathew Chaney

October 1, 2014

Contents

List of Figures

Listings

1 Question 1

1.1 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

Keep both files for each URI (i.e., raw HTML and processed).

If you're feeling ambitious, "boilerpipe" typically does a good
job for removing templates:

https://code.google.com/p/boilerpipe/
```

1.2 Resources

- md5: https://docs.python.org/2/library/md5.html
- requests: http://docs.python-requests.org/en/latest/
- futures: https://pypi.python.org/pypi/futures

1.3 Answer

Using the python script in Listing 1, 1000 unique URIs were dereferenced and their contents were stored into a file with the filename as the md5-hashed URI. Another file was created, called `uri_map` that was used to map the URIs to their md5-hashed filenames.

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

import requests
import futures
import md5

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('uris') as infile:
        uris = [uri.rstrip('\n') for uri in infile]

    with futures.ThreadPoolExecutor(max_workers=8) as executor:
        with open('html/uri_map', 'w') as map_file:
            uri_futures = [executor.submit(get_html, uri) for uri in uris]
            for future in 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))
                    map_file.write('{} {}\n'.format(uri, hashed_uri))
                    with open('html/' + hashed_uri, 'w') as outfile:
                        outfile.write(content)
                else:
                    print('Not writing {}, bad status code: {}'.format(uri, status_code))
```

Listing 1: get_html.py