Web Scraping: Spidering

SCC.413: Applied Data Mining

Week 14, 6th February 2019.

Here you will perform some web spidering, also referred to as "Browsing" in the lecture.

1. Create a new folder for this lab. Your h-drive is probably the best place.
2. We are going to be using a web scraping tool developed at Lancaster for the annual NLP summer school (https://github.com/UCREL/web-corpus-construction). You can download and unzip the code, or simply clone the git repository as follows:

   ```
   $ git clone https://github.com/UCREL/web-corpus-construction.git
   ```

3. Running the web scraper is quite straight forward. You can see the usage instructions from the downloaded/cloned `web-corpus-construction` folder, with:

   ```
   $ ./spider.py -h
   ```

4. Perform a test run, using one of the provided list of seed URLs, as follows:

   ```
   $ ./spider.py -seeds seed_urls/blogs.txt -db blogs -loglevel INFO
   ```

5. The first step of the scraper is to read the list of seed urls and store these in an SQLite database. If the scrape is interrupted, it can be restarted by simply running on the same database, e.g.

   ```
   $ ./spider.py -db blogs -loglevel INFO
   ```

6. Once complete, a series of web pages will be downloaded in the output folder, along with a metadata database (`metadata.db`). You can review the metadata database with `sqlitebrowser`, or directly with sqlite directly if you are familiar with it (https://www.sqlite.org/cli.html).

   You may need to install sqlite3 first:

   ```
   $ sudo apt-get update
   $ sudo apt-get install sqlite3
   ```

   and then:

```
$ sqlitebrowser blogs/metadata.db
```
or
```
$ sqlite3 blogs/metadata.db
```

7. Examine the output and the log (you can also run with different log levels), and look at the code, starting from `spider.py` (either on the github repo or locally), and check you follow how the resulting "corpus" is built. If it helps, write out the steps and/or discuss with your neighbours. There are also some slides available that discuss the code design, and some related issues with spidering.

8. Try changing the fitness ranking mechanism to the `SimplicityURLRank`, and observe the difference made.

9. **Advanced:** Try implementing your own `SampleURLRank` to rank on some other factor.

10. Examine the end conditions present, it should be straightforward to collect a larger corpus (currently set to 100 documents).

11. **Advanced:** Try implementing your own `SampleEndCondition`, e.g. all URLs have a depth less than 3.

12. Finally, examine the Filter methods, try editing the threshold and examining the results.