Python Web Crawler Documentation

Pratik Sahoo

June 14, 2023

1 Introduction

The link-following web crawler [3] is a Python program that extracts links from a given webpage and follows those links to extract further links from subsequent pages. It provides functionalities for crawling webpages up to a specified depth and sorting the extracted links based on file types. Furthermore, it also has the functionality to plot a site map based on links between pages.

2 Usage

To use the link-follower web scraper, run the Python script with the following command-line arguments:

- `-u <start_url>`: Specifies the starting URL for the web scraping process. If left empty, shows error message.
- `-t <depth>`: Specifies the depth to which the scraper will follow links (default is 2). If invalid threshold is given, shows error message.
- `-o <output_file>`: Specifies the output file to which the extracted links will be written. If not used, output written to `stdout`.
- `-g`: Optional flag to generate and display a (zoomable) site graph based on the extracted links, using textttmatplotlib.

3 Dependencies

The link-follower web scraper requires the following Python libraries:

- `requests`
- `beautifulsoup4` [2]
- `urllib.parse`
- `time`

- `getopt`
- `sys`
- `networkx` [1]
- `matplotlib`

Make sure you have these libraries installed in your Python environment before running the script.

4 Functionality

The link-follower web scraper provides the following functionality:

- Extracts links from a given http webpage, using a function utilizing beautifulsoup4 [2] functions.
- Follows the extracted links to subsequent http pages (within the same domain).
- Sorts the extracted links based on file types (e.g., HTML, CSS, images, scripts, etc.).
- Displays the sorted links or writes them to an output file, depending on the -o option.
- Optionally generates and displays a site graph based on the extracted links. The graph is plotted using the spring layout method provided within the `networkx` [1] module.

A point to be noted is that the majority of running time is spent in sending and receiving requests to the website involved, and most of the memory used is to store the URLs, both of which are essential to the program. Any minor optimisation (such as not storing edges when -g is not enabled) does not change the computational complexity of the base logic. Due to these reasons, it is non-beneficial to optimise this script further.

5 Example Usage

Here is an example command to run the link-follower web scraper:

```
python link_follower.py -u http://example.com -t 3 -o out.txt -g
```

This command starts the web scraping process from `http://example.com`, follows links up to a depth of 3, writes the extracted links to `out.txt`, and generates a site graph.

6 References

References

[1] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[2] Leonard Richardson. Beautiful soup documentation. *April*, 2007.

[3] Pratik Sahoo. Python web crawler. https://github.com/K1ngPat/WebCrawler/, 2023. Accessed on June 14, 2023.