Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

TCC/TSI/TECC: Information Retrieval

Programming Assignment #1
Web Crawler

Deadline: May 8th, 2022 23:59 via Moodle

Overview The goal of this assignment is to implement a crawler capable of fetching a mid-sized corpus of webpages in a short time frame while respecting the politeness constraints defined by each crawled website. In addition to the source code of your implementation and the actual crawled documents, your submission must include a characterization of these documents.

Implementation You must use Python 3 for this assignment. Your code must run in a virtual environment **using only the libraries included** in the provided `requirements.txt` file. Execution errors due to missing libraries or incompatible library versions will result in a zero grade. To make sure you have the correct setup, you can test it in one of the Linux machines provided by the Department of Computer Science[1] using the following commands:

```
$ python3 -m venv pa1
$ source pa1/bin/activate
$ pip3 install -r /path/to/requirements.txt
```

Execution Your implementation should include a `main.py` file, which will be executed in the same virtual environment described above, as follows:

```
$ python3 main.py -s <SEEDS> -n <LIMIT> [-d]
```

with the following arguments:

- `-s <SEEDS>`: the path to a file containing a list of seed URLs (one URL per line) for initializing the crawling process.

[1]`https://www.crc.dcc.ufmg.br/infraestrutura/laboratorios/linux`

- `-n <LIMIT>`: the target number of webpages to be crawled; the crawler should stop its execution once this target is reached.
- `-d`: (optional argument) run in debug mode (see below).

Debugging When executed in debugging mode (i.e. when `-d` is passed as a command-line argument), your implementation must print a record of each crawled webpage to standard output[2] as it progresses. Such a record must be formatted as a JSON document containing the following fields:

- `URL`, containing the page URL;
- `Title`, containing the page title;
- `Text`, containing the first 20 words from the page visible text;
- `Timestamp`, containing the Unix time[3] when the page was crawled.

The following example illustrates the required debugging output format for the first webpage fetched during a crawling execution:

```
{ "URL": "https://g1.globo.com/",
  "Title": "G1 - O portal de notícias da Globo",
  "Text": "Deseja receber as notícias mais importantes em
      ↪ tempo real? Ative as notificações do G1! Agora não
      ↪ Ativar Gasto familiar Despesa",
  "Timestamp": 1649945049 }
```

Crawling Policies Your implementation must keep a frontier of URLs to be crawled, which must be initialized with the seed URLs provided as input to you with the `-s` argument. For each URL consumed from the frontier, your implementation must fetch the corresponding webpage, parse it, store the extracted HTML content in the local corpus, and enqueue the extracted outlinks in the frontier to be crawled later. In addition to this standard workflow, **your implementation must abide by the following crawling policies**:

1. *Selection Policy.* Starting from the provided seed URLs, your implementation **must only follow discovered links to HTML pages** (i.e. resources with MIME type `text/html`). To improve coverage, you may optionally choose to limit the crawling depth of any given website.
2. *Revisitation Policy.* Because this is a one-off crawling exercise, you **must not revisit a previously crawled webpage**. To ensure only new links are crawled, you may choose to normalize URLs and check for duplicates before adding new URLs to the frontier.

[2] `https://en.wikipedia.org/wiki/Standard_streams#Standard_output_(stdout)`
[3] `https://en.wikipedia.org/wiki/Unix_time`

3. *Parallelization Policy.* To ensure maximum efficiency, you **must parallelize the crawling process across multiple threads**. You may experiment to find an optimal number of threads to maximize your download rate while minimizing the incurred parallelization overhead.

4. *Politeness Policy.* To avoid overloading the crawled websites, your implementation **must abide by the robots exclusion protocol**.[4] Unless explicitly stated otherwise in a `robots.txt` file, you must obey a delay of at least 100ms between consecutive requests to the same website.

5. *Storage Policy.* As the main target for this assignment, your implementation **must crawl and store a total of 100,000 unique webpages.** The raw HTML content of the crawled webpages must be packaged using the WARC format,[5] with 1,000 webpages stored per WARC file (totalling 100 such files), compressed with gzip to reduce storage costs.

Deliverables Before the deadline (May 8th, 2022 23:59), you must submit a package file (`zip`) via Moodle containing the following:

1. Source code of your implementation;

2. Link to your crawled corpus (stored on Google Drive);

3. Documentation file (`pdf`, max 5 pages).

Grading This assignment is worth a total of 15 points distributed as:

- 10 points for your *implementation*, assessed based on the quality of your source code, including its overall organization (modularity, readability, indentation, use of comments) and appropriate use of data structures, as well as on how well it abides by the five aforementioned crawling policies.

- 5 points for your *documentation*, assessed based on a short (pdf) report[6] describing your implemented data structures and algorithms, their computational complexity, as well as a discussion of their empirical efficiency (e.g. the download rate throughout the crawling execution, the speedup achieved as new threads are added). Your documentation should also include a characterization of your crawled corpus, including (but not limited to) the following statistics: total number of unique domains, size distribution (in terms of number of webpages) per domain, and size distribution (in terms of number of tokens) per webpage.

Teams This assignment must be performed **individually**. Any sign of plagiarism will be investigated and reported to the appropriate authorities.

[4] `https://en.wikipedia.org/wiki/Robots_exclusion_standard`
[5] `https://en.wikipedia.org/wiki/Web_ARChive`
[6] Your documentation should be no longer than 5 pages and use the ACM LaTeX template (sample-sigconf.tex): `https://www.acm.org/binaries/content/assets/publications/consolidated-tex-template/acmart-primary.zip`