IESP: 1. Homework

Yannick Kuhar, Mark Vale, Rok Peterlin

March 2021

1 Crawler Architecture

The architecture of our crawler consists of six classes each fulfilling their own task.

Crawler is the top level class that takes the number of workers as an argument and initializes both the frontier managers and the database. It is also responsible for starting the process.

Spider is the main worker class. Each spider has its own **HTMLParser** and **RobotFileParser**. It parses **robots.txt** of each domain and extracts content from sites which is later written in the database. If an empty frontier is encountered spider falls asleep for 60 seconds and check the frontier every 5 seconds. In case any other spider adds a link to the frontier the spider that fell asleep wakes up and continues crawling.

The **FrontierManager** class controls all data structures shared by spider. Before any given URL is added to the frontier a basic duplicate detection check is performed - if either frontier or history contains that URL we simply reject it.

HTMLParser is a wrapper class for **BeautifulSoup** library. It is used for retrieval of links and images from HTML content.

The version of **RobotFileParser** class imported from **urllib.robotparser** for some reason didn't work, so the class was added manually.

The only extension we made to the **database** scheme is the hash table, where we store 4 hashed for each page.

Doc_similarity class was implemented as a solution of the bonus task. We implemented a version of *SimHash technique* which combines hashes of words on the given page. Each word was hashed using *adler*32 *checksum algorithm*, which was also implemented from scratch. Each page was defined by 4 different (8 bit) hash codes in the end. The class uses 2 hyperparameters: number of matching hash codes between two pages needed for the two pages to be duplicates of one another, and Jaccard similarity threshold. If two pages were deemed duplicates (just from comparing the 4 hash codes), the more in-depth inspection was carried out - Jaccard similarity between two documents or sets of words in our case. If the to-be-added page passed the Jaccard test (had similarity score bellow specified threshold) it was added to the database as an original page.

2 Crawler parameters and problems

The only parameter is the number of workers, which must be a positive integer. If the parameter is not specified or the value is inappropriate, the default value of 5 is used instead. Use case can be seen in **README**.

The latest and most important problem we've encountered is a very slow process of adding HTML files to the database (around 330 files were added in about 12 hours). Since this problem hasn't been solved yet, we documented the statistics only with what we had at the time.

Another problem we encountered was adding duplicate pages to the database. The problem lies in our method of adding pages to the database. In order to solve the problem we would have to rebuild the database from scratch, which was not feasible due to our aforementioned runtime problem.

3 General statistic

Out (incomplete) database consists of:

- 362 HTML files
- 0 duplicates (hyperparameters were set too strictly and 0 duplicates were consequently found),
- 211 sites,
- 7080 pages,
- 2 binary files,
- 1642 images,
- 6716 links in frontier left to be explored

Interactive visualization of links can be seen by opening the **links.html** file in your browser of choice. Please wait a few minutes for the graph to finish loading. Since our database was not yet full, we ”managed to fit it all” in a single visualization.