Web Information Extraction and Retrieval
Programming Assignment 1

Rok Cej, Metodija Bucevski

1. Crawler implementation

The crawler was implemented in Python. It uses a PostgreSQL database running as a Docker container. The crawler can be (remotely) shut down by sending it a kill signal on port 2803.

1.1. Frontier

The frontier was implemented in the database, using the crawldb.page table. Pages that are in the frontier are labeled with the type code FRONTIER. When a page is retrieved from the frontier, its type code is changed to PROCESSING. Pages in the frontier are sorted by their timestamps, which enables breadth-first crawling strategy. As instructed, only *.gov.si URLs are allowed to be added to the frontier.

1.2. Robots Exclusion Standard

To respect the robots exclusion standard, we used the library reppy. The crawler supports the following robots.txt directives: User-agent, Allow, Disallow, Crawl-delay and Sitemap along with wildcards (* and $) for pattern matching. If a crawl delay isn't specified, a minimum delay of 5 seconds is used.

To enforce crawl delay, we keep track of when each domain was last visited. If the crawler attempts to process a page before the crawl delay has passed, the page is added to the back of the frontier.

1.3. Getting pages

First, the crawler sends a HEAD request in order to obtain the response code and the page content type. If the page is not a binary file, we use Selenium to download and render the HTML with Geckodriver.

1.4. Duplicate detection

To avoid URL duplication, we use the URL canonicalization library url-normalize. It takes care of capitalization, consistent character encoding, relative paths (. and ..), adds a trailing / to URLs without a path and removes empty fragments (#). To extend this functionality, we also manually remove fragments with a specified value (e.g. #text), as it is generally not used to modify the content of a page.

In order to detect duplicate pages with different URLs, we extended the database with the html_hash field. For each page, we first compute a hash of its HTML content. If the same hash is already present in the database, we know that the page is a duplicate.

1.5. Link detection

To find links on a page, we first check the `href` attribute of all `<a>` and `<area>` tags. Then, we check the `onclick` attribute of all HTML elements inside the body and look for links in the JavaScript code. This is done by finding either `document.location`, `window.location`, `self.location` or `location.href` and then extracting the string that is being assigned.

Once a link is found, we first check if it starts with `javascript:`, `mailto:` or `tel:`. Email addresses and telephone numbers are stored in the database. Next, we check if the link has a supported file extension, in which case, we store it as a crawldb.page_data entry. The remaining links are normalized and added to the frontier.

1.6. Image detection

To find images on a page, we check the `src` attribute of all `<img>` tags. We ignore any empty URLs and Base64 encoded images (URLs that start with `data:`). Content type is extracted from the image's headers by looking into Selenium's request history (Seleniumwire). If no entry is found, a manual HEAD request is sent. If that fails, the content type is extracted from the image's name. The image is then stored into the database.

2. Problems during development

- Race conditions due to multithreading
- JavaScript `alert()` and `print()` causing Selenium to timeout
- Library url-normalize crashing the crawler when parsing invalid URLs
- Built-in library urllib.robotparser not parsing robots.txt correctly
- Getting responses to requests made by Selenium while loading pages

3. Results

3.1. Sites and pages

In total, the crawler found 131,686 pages across 290 sites. Excluding the pages that were unavailable, disallowed by robots.txt and that are still in the frontier, the crawler processed a total of 30,340 pages in 70 hours.

Page type	Count
HTML	26766
Binary	880
Duplicate	2694
Unavailable	501
Disallowed	10400
Frontier	90445
Total	**131686**

Table 1. Number of pages by type.

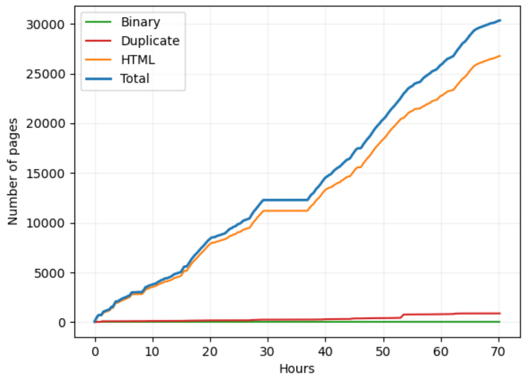

Figure 1. Number of pages by crawl time.

There were a total of 2,888,682 links in the database. Among these, 1,936,828 links connected pages that were processed.

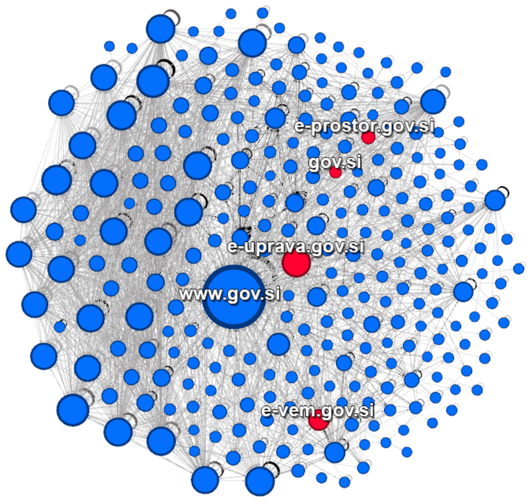

Figure 2. Visualization of links between sites.

3.2. Documents and images

We found a total of 42,776 documents, which add up to an average of 1.60 documents per page.

Document type	File extensions	Count
PDF	.pdf	26481
Word	.doc, .docx, .docm	7425
PowerPoint	.ppt, .pptx, .pptm	161
Excel	.xls, .xlsx, .xlsm	2332
Archive	.zip, .rar	5899
CSV	.csv	478
Total		**42776**

Table 2. Number of documents by type.

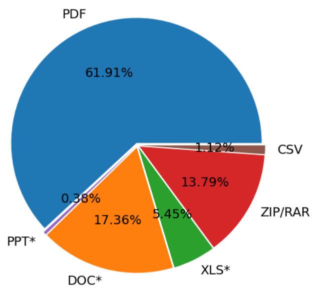

Figure 3. Ratio of documents by type.

There were 304046 images, which add up to an average of 11.36 images per page.

Image type	File extensions	Count
JPG	.jpg, .jpeg, .pjpeg	26940
PNG	.png	104241
GIF	.gif	77716
SVG	.svg	95040
BMP	.bmp	22
TIFF	.tif, .tiff	87
Total		**304046**

Table 3. Number of images by type.

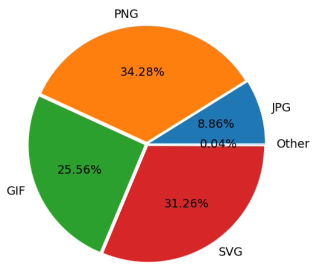

Figure 4. Ratio of images by type.

4. Other Information

A total of 1794 telephone numbers and 1338 email addresses were collected.