Programming assignment 1

Filip Turk - Lan Biteznik – Tschimy Aliage Obenga

Introduction

This report outlines the implementation and functionality of a web crawler designed to extract URLs, handle various file types, and store relevant data in a PostgreSQL database. The crawler incorporates mechanisms for duplicate detection and priority search based on keywords and phrases.

The main objective of this crawler is to extract data about Erasmus exchange information from the Faculty of Computer and Information Science official website.

Implementation Details

HTTP Downloader and Renderer

- The crawler retrieves web pages using HTTP requests, adhering to domain-specific constraints and respecting robots.txt policies.
- 5 seconds between request and 3 for rendering.

Link and Image Extraction

- Detects hyperlinks and images by parsing HTML tags (<a> for links and <img> for images) and extracting the href or src attributes.
- Only images referenced in <img> tags are processed.
- Javascript based OnClick links are also searched.

Data Extraction

- Extracts text content, metadata, and file classifications.
- Identifies different file types (HTML, PDF, DOCX, PPTX) using file extensions and MIME types.
- Non-HTML files are processed separately, and their metadata is stored without binary content.

Duplicate Detection

- Uses MinHash to generate hash representations of page content.
- If the similarity score between two pages exceeds a predefined threshold, the URL is discarded.
- MinHashes of all processed pages are stored in a local runtime cache for efficient comparison.

URL Frontier and Prioritization

- Uses a priority queue (heapq) to manage URLs to be crawled. For this reason a database frontier and duplicates are not stored in the database.

- Prioritization is based on relevance using a bag-of-words model and cosine similarity.
- Pages matching multiple predefined keywords and phrases receive higher priority.

Preferential Crawling Strategy

- Extracted data, including URLs, metadata, and file classifications, is stored in a PostgreSQL database.
- Keyword-based prioritization ensures that pages relevant to Erasmus exchange programs are crawled first.

Challenges and Solutions

Challenge	***Solution Implemented***
Implementing MinHashing to avoid near duplicates	Storing MinHashes was not space-efficient, so we kept them in a cache used for comparison.
Ensuring the correct page is queued next for preferential crawling	Implemented a priority queue (heapq) for ordering URLs instead of saving pages as frontiers in the database, which worked efficiently with multiple workers.
Multi-threading complexities	Used a thread-safe queue and managed database access with proper locking mechanisms.

Evaluation and Performance

The web crawler was tested on the Faculty of Computer and Information Science website, focusing on Erasmus exchange-related pages. Key performance indicators include:

- Crawling Speed: The system efficiently processed and stored URLs with multi-threading support.
- Duplicate Removal: The MinHash technique successfully eliminated near-duplicate pages, reducing storage and processing overhead.
- Prioritization Accuracy: Keyword-based ranking ensured that Erasmus-related pages were crawled earlier.

Conclusion and results

number of sites	1
number of web pages	1455
number of duplicates	Were discarded before database insertion
number of binary documents by type	('DOC', 1), ('DOCX', 24), ('PDF', 483)
number of images	12600
average number of images per web page	8.69