University of Leeds **School of Computing**

COMP3011, 2023-2024

Web Services and Web Data

A Search Tool for a Simple Website

By

Abderrahamne Bennabet

201469614 - sc20ab2@leeds.ac.uk

Date: 09/05/2024

1. Introduction

In this coursework, I have successfully completed all the required tasks, including crawling web pages, creating an inverted index of all word occurrences on each page, and enabling users to search for pages containing specific terms with ranking features. These features are accessible through a user-friendly command-line interface.

2. The crawler

Implementing the crawler in my tool involves a Python-based application designed to browse and index web pages methodically. Using the requests library, the crawler makes ***HTTP*** requests to fetch web page content. Upon retrieval, the ***BeautifulSoup*** library from the ***bs4*** module is employed to parse the ***HTML*** content. This enables the extraction of textual data and hyperlinks for further navigation.

The crawling process begins from a base ***URL*** and uses a recursive approach to navigate through all accessible links on each page. Each visited ***URL*** is added to a ***visited*** set to prevent re-crawling and ensure efficiency. The crawler respects a politeness policy by pausing for ***six seconds*** between requests to avoid overloading the website's server.

3. The inverted index

The inverted index is a central data structure in our crawler, implemented as a ***dictionary*** in Python. It stores each word found on the crawled web pages as ***keys***, and each critical maps to another ***dictionary*** where ***URLs*** are keys and their corresponding values are the frequencies of the word at those URLs. This structure effectively supports quick look-ups for words across all indexed pages and is crucial for the search functionality.

When the crawler processes a page, it extracts text, normalizes it to lowercase, and splits it into words, filtering out common ***stopwords*** using the ***nltk*** library and removing punctuation marks so it just stores the word. Each relevant word is added to the inverted index with its associated ***URL*** and ***frequency count*** updated.

The inverted index is saved to a JSON file using Python's json for persistence***.dump*** function, which serializes the Python dictionary into ***JSON*** format, allowing the index to be saved to disk with readable formatting. The index can be reloaded using ***json.load***, which deserializes the ***JSON*** back into a ***Python dictionary***, ensuring that data can be efficiently loaded back into memory when needed without re-crawling.

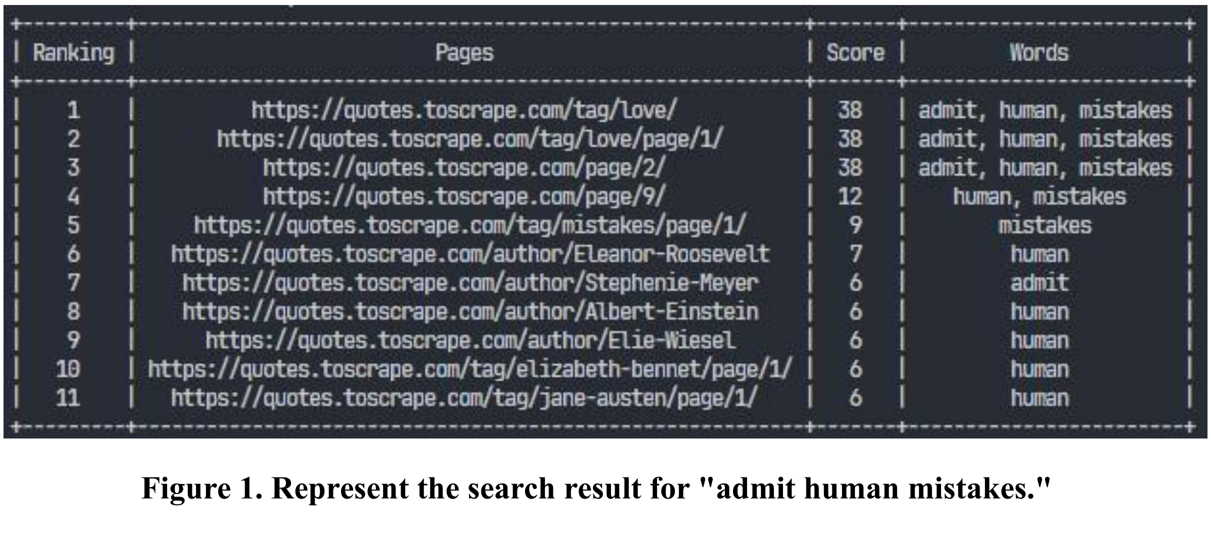
4. The ranking method

The search engine leverages a diverse approach to rank search results based on their relevance to a user's query. This approach incorporates the following techniques:

- **Word Frequency:** The core principle is that pages containing the search terms more frequently are deemed more relevant. The crawler counts each instance of the search terms across all indexed pages. Pages with a higher word count for the specific terms are prioritized and positioned higher in the ranking.
- **Unique Terms Bonus:** The system goes beyond just raw word frequency. It recognizes the value of pages that comprehensively address the search topic. Pages containing a wider variety of the search terms, even if the individual word counts are lower, receive a significant score boost. This bonus rewards content that provides a more well-rounded exploration of the search query.
- **Full Match Bonus:** The system places a premium on exact matches. Pages that contain all the search terms specified in the user's query are awarded an ***additional 20 points***. This helps the ranking to present results that directly correspond to the complete query, offering the most relevant and focused information.

- **Extra Bonus:** There is an extra bonus when the query is found on a title, heading 1, heading 2, or heading 3.

Combining these techniques, the ranking system aims to surface the most pertinent and informative pages in response to a user's search query. Figure 1 illustrates the search results for the query ***"admit human mistakes."*** The ***URLs*** are listed in ***descending*** order of their relevance scores, with the highest-scoring pages appearing at the top. The "Words" header indicates the specific terms found on each webpage**.**



```
+---------+--------------------------------------------------------+-------+-------------------------+
| Ranking |                         Pages                          | Score |          Words          |
+---------+--------------------------------------------------------+-------+-------------------------+
|    1    |           https://quotes.toscrape.com/tag/love/        |  38   | admit, human, mistakes |
|    2    |       https://quotes.toscrape.com/tag/love/page/1/     |  38   | admit, human, mistakes |
|    3    |           https://quotes.toscrape.com/page/2/          |  38   | admit, human, mistakes |
|    4    |           https://quotes.toscrape.com/page/9/          |  12   |     human, mistakes     |
|    5    |     https://quotes.toscrape.com/tag/mistakes/page/1/   |   9   |        mistakes         |
|    6    |  https://quotes.toscrape.com/author/Eleanor-Roosevelt  |   7   |          human          |
|    7    |  https://quotes.toscrape.com/author/Stephenie-Meyer    |   6   |          admit          |
|    8    |  https://quotes.toscrape.com/author/Albert-Einstein    |   6   |          human          |
|    9    |    https://quotes.toscrape.com/author/Elie-Wiesel      |   6   |          human          |
|   10    | https://quotes.toscrape.com/tag/elizabeth-bennet/page/1/ |   6   |          human          |
|   11    |  https://quotes.toscrape.com/tag/jane-austen/page/1/   |   6   |          human          |
+---------+--------------------------------------------------------+-------+-------------------------+
```

Figure 1. Represent the search result for "admit human mistakes."

5. Using the tool

The tool can be used as the following:

- Create and activate the environment, then install the necessary libraries through:
  pip install -r requirements.
- Run the application through *python ./search.py –[command]* Command list:

- ***python ./search.py --build :*** This crawls the website, build the index, and save the resulting index into a single file *inverted_index.json*.
- ***python ./search.py --load :*** Loads the index from the file system, which only works after building the inverted.
- ***python ./search.py –print [word]*** : This command prints the inverted index for a particular word.
- ***python ./search.py –find [word] or [sentence] :*** Finds a certain query phrase in the inverted index and returns a list of all pages containing that phrase. The sentence has to be inside quotes as an example:
  python ./search.py —find "human mistakes"