HomeWork-3 Report

Name: Karthik Chintamani Dileep

1. Crawling [Note down steps implemented for each of the below]

    a. URL Canonicalization –

    i. The canonicalize_url function takes a URL and an optional base URL as input.
    ii. It converts the URL to lowercase and splits it into different components (protocol, hostname, port, and path).
    iii. It then constructs a canonical URL by combining the components back together, removing any unnecessary parts, and handling edge cases such as relative URLs and URLs without a protocol.
    iv. The function also maintains a cache (canonicalize_url_map) to avoid redundant computations for the same URL.
    v. It returns the canonicalized URL if it is a valid HTTP/HTTPS URL, and None otherwise.

    b. Frontier Management

    i. The crawler uses a deque (frontier) to manage the frontier.
    ii. Initially, the seeds are added to the frontier after canonicalization and checking the robots.txt file.
    iii. During the crawling process, new URLs extracted from the fetched pages are added to the end of the frontier.
    iv. URLs are popped from the front of the frontier for processing.
    v. After a certain number of documents have been fetched (controlled by num_of_docs_in_each_file), the frontier is sorted based on the link scores, giving higher priority to more relevant URLs.

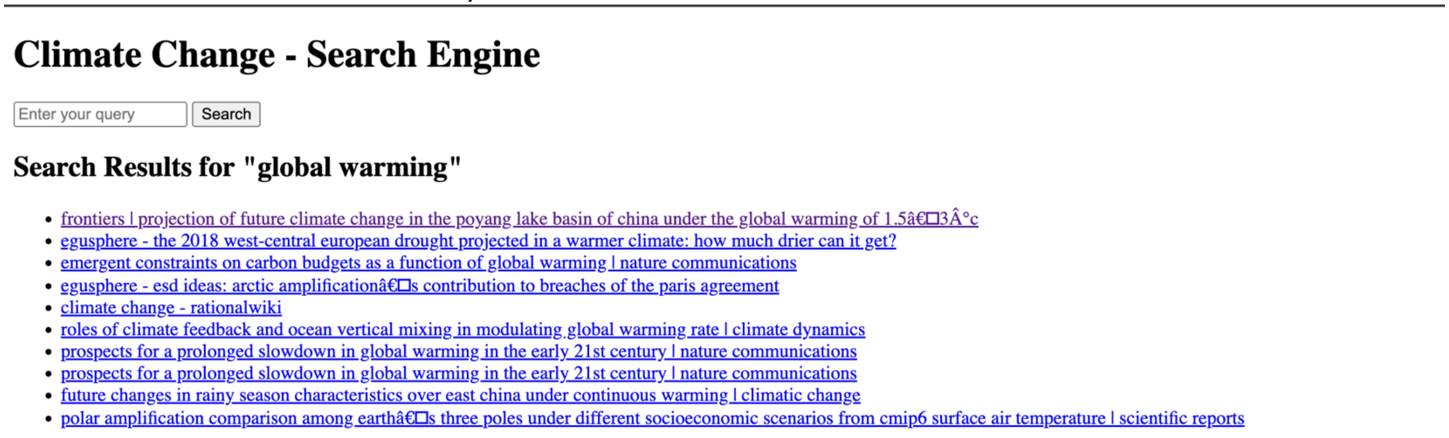
    c. Politeness Policy

    i. The robots_file_allowed function checks the robots.txt file for a given URL to determine if the crawler is allowed to fetch the URL and what the specified crawl delay is.
    ii. The function maintains a cache (robots_map) to avoid redundant robots.txt file fetching for the same domain.
    iii. The check_delay function ensures that the crawler respects the crawl delay specified in the robots.txt file for each domain.
    iv. Some domains are blacklisted (blocked_domains) and skipped entirely.

    d. Document Processing

    i. The save_response function fetches the content of a URL, checks if the response is valid (HTTP 200 and HTML content type), and processes the HTML content.
    ii. The process_html_content function extracts the title, links, and text content from the HTML using BeautifulSoup.
    iii. The extracted data is stored in various maps (title_map, links_map, data_content_map) for later use.
    iv. The function also checks if the page is in English.

2. Vertical Search
    a. Add a Screenshot of your Vertical Search UI



Climate Change - Search Engine

Enter your query [Search]

Search Results for "global warming"

- frontiers | projection of future climate change in the poyang lake basin of china under the global warming of 1.5â€□3Â°c
- egusphere - the 2018 west-central european drought projected in a warmer climate: how much drier can it get?
- emergent constraints on carbon budgets as a function of global warming | nature communications
- egusphere - esd ideas: arctic amplificationâ€□s contribution to breaches of the paris agreement
- climate change - rationalwiki
- roles of climate feedback and ocean vertical mixing in modulating global warming rate | climate dynamics
- prospects for a prolonged slowdown in global warming in the early 21st century | nature communications
- prospects for a prolonged slowdown in global warming in the early 21st century | nature communications
- future changes in rainy season characteristics over east china under continuous warming | climatic change
- polar amplification comparison among earthâ€□s three poles under different socioeconomic scenarios from cmip6 surface air temperature | scientific reports

    b. Explain briefly how you implemented it.
        i. Used Flask to run the application.
        ii. Identified that the search should be matched with “text” in a doc.
        iii. Got the search results from ES and displayed the title for each doc.
        iv. When clicked on the doc it navigates to the page.

3. Extra Credits Done [Note done what was done for each extra credit]