CSC326 Final Lab Report

Yufei Wang 1001623843
Jason Wu 1001462865

Design

Backend - Database

In this lab, we created 4 tables in the database. Schema shown as below:

- lexicon(word_id INTEGER, word TEXT, PRIMARY KEY (word_id));
- document_index(doc_id INTEGER, url TEXT, PRIMARY KEY (doc_id));
- inverted_index(word_id INTEGER, doc_id INTEGER, PRIMARY KEY (word_id));
- page_rank_score(doc_id INTEGER, rank_score INTEGER, PRIMARY_KEY (doc_id));

After the crawler is finished crawling the urls, a check is done on the database to see if tables exist before inserting new values. If tables exist already, they are deleted and a new table is created. This is to ensure that the database only has the data related to the urls crawled in current run.

Tables populated from crawler will be queried from frontend, when the user searches for a word. The pages that contain the searched word will be found and ranked through the page_rank function. If the word is not found in the database, user is directed to the error page.

Frontend

Some features that were added are returning back to home page after clicking home logo, having a search icon inside the search bar, highlight over hovered buttons/links, toggle button for search history(if logged in) and allow multiple word search.

Depending on how many words the user searches, the number of urls returned differs.

- If the user searches only one word, then all the urls that contain the word will be found from querying the database.
- If the user searches for more than one word, urls are found for each word and stored into a list, it is then compared with the urls of the next word and all common urls(with their pagerank score) are retracted and added into a set.

After retrieving a list of urls, it is then sorted and displayed on the results page on the website.

Proposed vs Completed Design

<u>Backend</u>

For persistent storage on the backend, initially in backend storage, when crawler crawls a url, it checks if the document has been seen or not, if not it will get the doc_id and check if it exists in the database. If not exist in database, it will insert doc_id and corresponding url into the document_index table. Same for each word, if word is not seen and does not exist in database, the word and word_id will be inserted into the lexicon table. The newly inserted doc_id and word_id would be returned.

However, this did not work out and crawler was not able to crawl the urls normally. After changes, the completed crawler parses through all words in documents from urls, all information is stored in corresponding dictionaries: _word_id_cache, _doc_id_cache, _inverted_index_cache, _doc_id_score. Once crawler is finished crawling, it checks to see if tables already exist in database or not, and drops if they exist. Tables are created and dictionaries are looped through and inserted into their corresponding tables.

<u>Frontend</u>

Initially, we wanted to implement spell check, night mode, calculator and animated logo. But due to time restrictions and other priorities, they were not implemented. For spell check, existing libraries only supported Python 3 and could not be runned on Python 2.7. In order to implement spell check and animated logo, more time was needed.

For Google login, we initially planned to enable multiple users to login to their account and logout successfully. However, user was not able to log out properly through the Google API. In the end, only one user can login and logout.

Testing Strategy

<u>Frontend</u>

Testing for frontend features were done manually, such as

- Count number of words in searched string correctly
- Making sure special character will not be counted as one word
- Login status displays correctly and session is saved
- Search history display according to login status
- Pagination displays urls in correct page rank order
- Error Page displays correctly

Backend

For backend, testing was completed by running crawler against test pages:

- Test crawler for crawling zero, one, two and more urls
- Created simple test pages to check if inverted_index and resolved_inverted_index contained the correct data
- PageRank
    - Check for when page doesn't have a score
    - Print urls and it's pagerank score, make sure it is printing in order
- Persistent Storage
    - Check to see if data is inserted into database correctly
    - Data can be retrieved correctly

Lessons Learned

- Creating webpage using Python
- Learning HTML and CSS to create Frontend of website
- Working with AWS, launching and managing a remote server instance

What you would do differently

- Use javascript, as it can dynamically create tables and have more control over frontend features
- Implement multithreading on backend

Helpful Course Material

Python related lectures were very useful: imperative programming, object oriented programming, lambda expressions

Amount of time spent on labs

Each partner put in around 7-10 hours of work for each lab.

Which part of the project you think is useful and you believe the labs should spend more time on it.

Crawler and page rank were very useful algorithms to know and we wished it was explained better and taught in class.

Which part of the project you think is useless and you think it should be removed from the labs when this course is being offered in the future.

The benchmarking aspect of the lab was not explained very well and we were not sure what the results were trying to show. Lab2 and Lab3 results differed due to the words searched and also due to use of database.

Other feedback or recommendations for the course

In the future, it would be better to have a more organized lab marking structure. During use of AWS, websites were deployed during time it was not being marked.

Work Distribution

Work was distributed evenly between both teammates. To get experience in both frontend and backend, we alternated between labs. For Lab1, Jason did frontend while Yufei did backend crawler. For Lab2, Yufei did frontend and Jason did AWS setup and deployment/benchmarking. Lab3, Jason worked on frontend and Yufei did backend. For last lab, Jason worked on features and deployment and Yufei worked on lab report and a feature.