Comp 4321
Project (Phase 2)

Name: Tang Ka Fu (20627414)
Chen Ka Wai (20604175)
Luk Yung Kwan (20601642)

Overall design

As an overview, it is a web search engine specializing in searching pages under the https://cse.ust.hk domain. We first crawl all the links with "https://cse.ust.hk" as the root using breadth-first search strategy. During crawling, we cumulatively update mapping tables for pages and keywords and continuously index all the processed words and processed link information into the local system. As in the local system, all the information is stored in RocksDB, which is a high performance embedded database for key-value data. After gathering all the information required for our search engine, we implemented a retrieval function on both keyword search and phrase search using cosine similarity while favoring the title matches with certain weight. After all the backend preprocessing and implementation, we built a web server with tomcat and provided a user-friendly web interface accepting query from all users who are accessible to our webpage.

The file structures used in the index database

All the data files are stored as key and value pairs in the disks with RocksDB as follow :

	key	value
1	URL	page id
2	page id	URL
3	parent id	child id
4	child id	parent id
5	word	word id
6	page id	{keyword , keyword frequency}
7	title word	{page id and position}
8	body word	{page id and position}
9	page id	page properties (including page tile, URL, last modification date, page size)

(Database 1-4)Mapping Tables for URL and PageID

There are 3 main purposes for mapping tables for url and pageID.

1) provide fast retrieval for specific url

2) provide fast retrieval for specific pageID
3) establish a parent-child relationship for a group of crawled links

With the above purpose, we have the following design:

	Key	Value
1	URL	page id
2	page id	URL
3	parent id	child id
4	child id	parent id

All of the above tables make use of data structure hashmap. Hashmap allows the fast retrieval in specific va lue data with a given key. The one-to-one relationship ensures the speed of data processing.

The first two tables are to create a mapping relationship between url and pageID. This ensures that pageID can be used to retrieve url, or vice versa.

The last two tables are to establish a tracing relationship of pages in parent-child relationship. In case the us er wants to get access to the relevant parent page, the child pageID can be used to retrieve the parent page i n a fast manner, or vice versa,.

Besides, the above design maintains the flexibility in extending our database schema design. For example, i f we want to store the information and metadata of a page, we can then develop a mapping table with the ke y of pageID and value of the page data.

(Database 5)Mapping table for word and wordID

There are 2 main purposes for mapping tables for word and wordID.

1) provide fast retrieval for specific word
2) provide fast retrieval for specific wordID

With the above purpose, we have the following design:

Table Name	Key	Value
5	word	word id

The table maps the relationship between word and wordID. This ensures that wordID can be used to retriev e words.

Forward(Database 6) and Inverted indexes(Database 7-8)

There are 2 main purposes for tables for forward and Inverted indexes

3) support deletion of pages and words
4) search for matched words which correspond to query

With the above purpose, we have the following design:

Forward indexes

Table Name	Key	Value
6	page id	{keyword , keyword frequency}

Inverted indexes

Table Name	Key	Value
7	title word	{page id and position}
8	body word	{page id and position}

Forward index and inverted index are separated into two sets of tables. More specifically, for each set, we c ould use a pageID to search for a set of words that it contains. Also, we could use the wordID to retrieve th e corresponding pageID and the word positions.

Inverted index are separated into two parts, including title and body. This facilitate the procedure of increas ing pages score when the page title contains any words of the query.

(Database 9)Page properties

The main purposes for tables for Page properties

1) record the general information of pages

With the above purpose, we have the following design:

Table Name	Key	Value
9	page id	page properties (including page tile, URL, last m odification date, page size)

This table stores pageID as the key and the information (i.e. title, URL, last-date-of-modification, size) of th at page as the value. By doing so, we could easily get the basic information of all the pages that we have cr awled through the crawler.

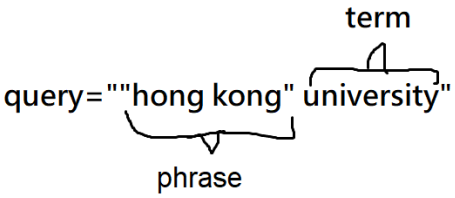
Algorithms Used

Vector Space Model

In retrieving relevant terms that match the query information, we used a cosine similarity algorithm to deter mine document-query similarity.

For a query, there are two data types, one is term and one is phrase.

Given an example query, “ “hong kong” university”, the word with no quotes is a term while double-q



uoted string “hong kong” is considered as a phrase.

To compute the cosine similarity between a query and a document, we develop two algorithms, one is for c omputing term cosine similarity, one is for computing phrase cosine similarity. Both the algorithm requires several variables such as term frequency tf, max term frequency max_tf, document frequency df, inverse do cument frequency idf.

To favor title matches, we put the terms present in the index and the terms present in the body in separate i ndexes. The terms in the body and title are represented as different vectors and the cosine similarity is com puted for both. The scores are combine according to the weighting:

$$Score = 2*Score(title) + Score(body)$$

Stopwords Removal and Stemming

The implementation of stopword detection uses the hash table data structure. We make use of the Porter alg orithm to do the stemming. We then use the stem words for computing cosine similarity for term and phras e.

Implementation of term cosine similarity

We make use of a hash map to compute and store the term cosine similarity. Since we already store the dat a in inverted index database, we extract the string value containing pageid and word position from the datab ase. We also get the max_tf from max_tf database. Analyzing the string value with string functions, we get all the information and then compute the term similarity.

Implementation of phrase cosine similarity

The implementation of phrase cosine similarity is similar to implementation of term cosine similarity. How ever, we need to compute the required variable after combining the inverted index of all the terms in the ph rase. Then we do analysis to check if the required variables are found. After getting all variables, we compu te the phrase cosine similarity.

Installation procedure

Please have Java with version 8 or above installed in your local machine. It is useful to also have Eclipse installed since we were working on Eclipse. Therefore, you can open the Eclipse project directly with all the classpath predefined.

In the submission, there are two folders, namely "Project" and "apache-tomcat-10.0.5_group11". The former one is the eclipse project that we are working on and the latter one is the tomcat server. To keep the submission file small, we have removed the rocksdbjni.jar, so please include it in the Project\manage_rocksdb directory and apache-tomcat-10.0.5_group11\webapps\example\project\lib directory. You can rebuild all the db files by running the Crawler_BFS.java and put generated db files inside apache-tomcat-10.0.5_group11\bin. After setting up, you can run the startup.sh under apache-tomcat-10.0.5_group11\bin. In case you are working on your local machine, you can visit http://localhost:8080/example/group11.jsp to see our system website.

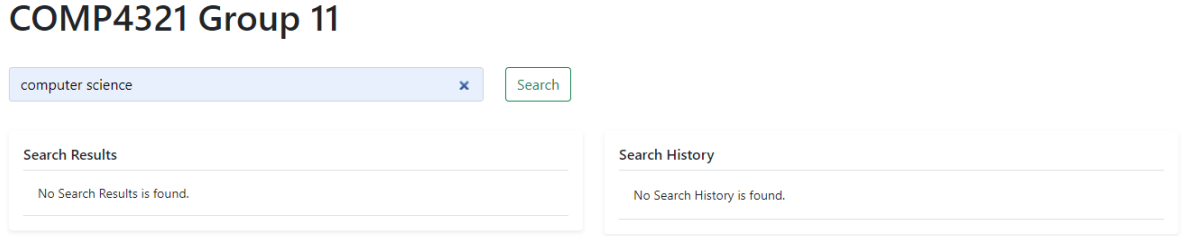
Highlight of extra features

Improvement of the web interface for better user experience

Usually for modern search engines, Google as an example, it well design the user interface to accommodate more useful information for users while striking a balance to have the least information shown for clarity. Therefore, we have added a number of features to improve the usability and readability.

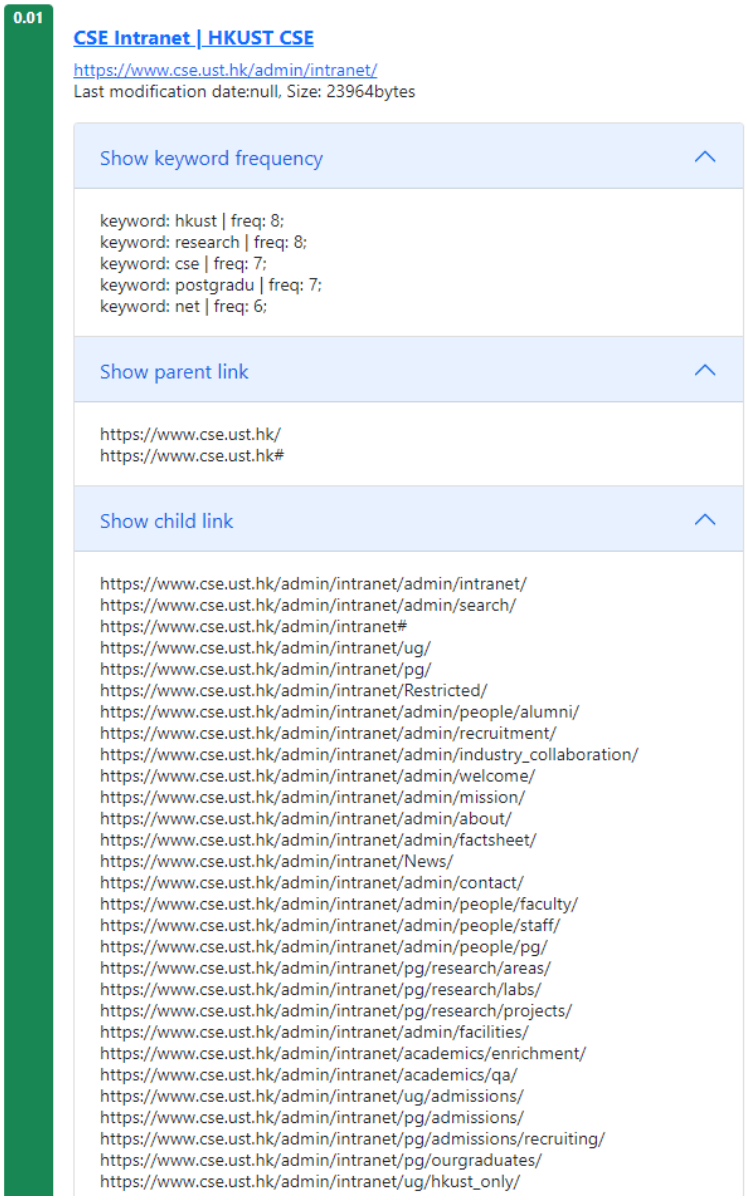
1. Search and view search result on the same page

In this project, we are not required to have a very user-friendly design that users may need to trigger a search in one webpage and view the result on another webpage. Then, if the users want to trigger another search with another query, he/she may need to go back to the previous page manually which is tedious. As a result, we design our webpage in the way that users can search a query by clicking the search button while staying on the same page to view the search result. Then, when the user wants to submit another query, he/she can type in the query on the same page and submit the query directly without going forward and backward between web pages.



2. Group extra information together

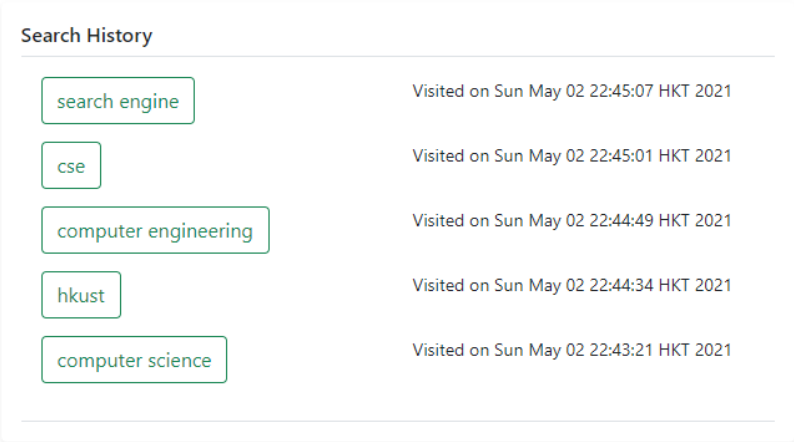
In this project, we are required to output not only the title of the webpage but other information like top-5 keywords, child links and parent links. Although some information like title and url are brief but keywords, child links and parent links are quite lengthy and not always appealing to normal users. Consequently, if all the information is listed. The search result would be very long and the information of the next retrieved page may be burdened by the child links and parent links of the previous page. In such a way, users may need to carefully scroll through the webpage which is time-consuming and extremely user-unfriendly. Therefore we designed to group these lengthy information into a collapsable menu. Users can view the information whenever they feel necessary to do so. After this UI modification, the search result is more clean and neat.



Search history

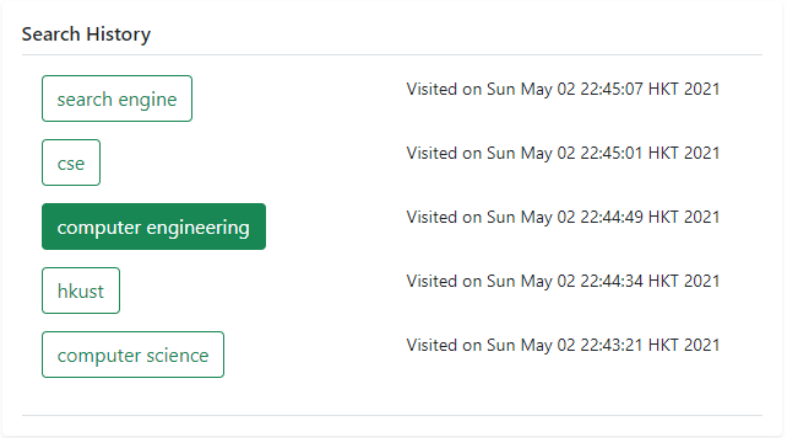
1. View search history

It is very common that commercial search engines like Google would let users keep track of their s earching history. For one reason, real-world users may want to return to the previous search result since they may not find useful information after further search. Therefore we have also included th is functionality in our search engine as well. We store the queries that users have searched in a pan el and the date and time that users search are also stored.



2. Return to previous search with one click

Addition to the search history, we found that it is tedious for users to re-typing their query and sub mit. Therefore we allow users to just click on the query and return back to the previous search inst antly. Below is the example to search: computer engineering by clicking the search history button.



3. Keep previous search query on the search bar

For real-world users, they may search once with the query that they think is good. But sometimes, the result is not so relevant to the user so he/she may want to modify the query. Therefore, we sav e the previous query and forward to the current input box, so the user can modify the previous and search without typing the full query.

search engine

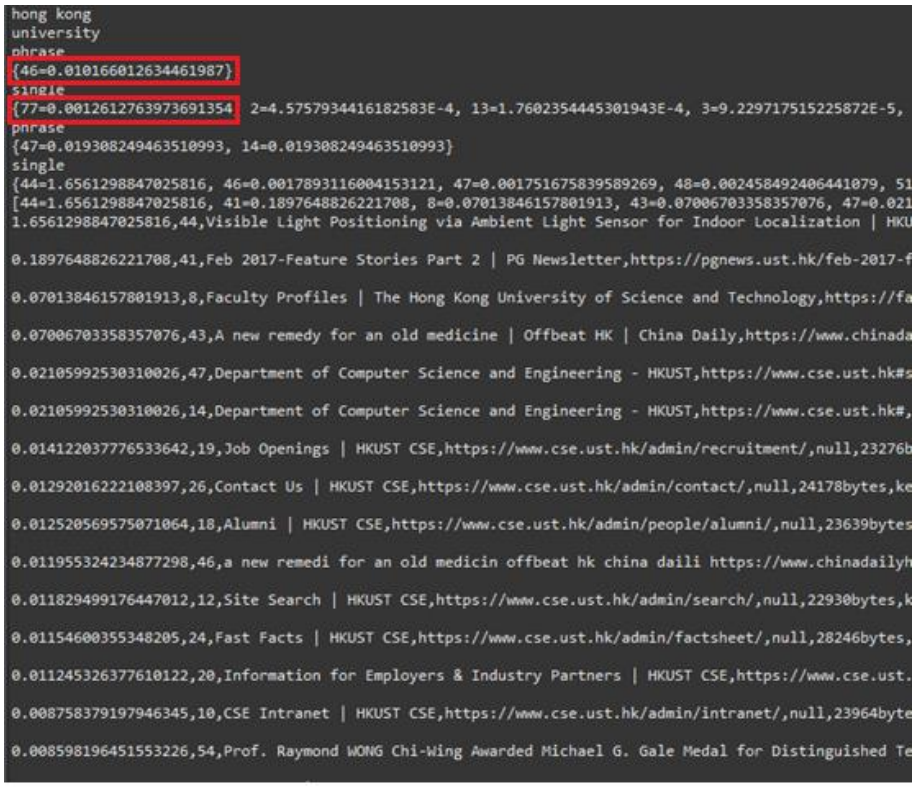
Search

Testing of the functions implemented

After all the functions have been implemented, test cases are implemented to test the functionality of the system. Below is one of the testing cases where the query is “hong kong university”.

```
String test = "\"hong kong\" university";

RetrievalFunction rf = new RetrievalFunction(test);

for(Object obj:rf.get_retrieval_result() ) {
    System.out.println(obj+"\n");
}

for(Object obj:rf.get_retrieval_result() ) {
    System.out.println(obj);
}

for(Object obj:rf.get_c_links() ) {
    System.out.println(obj);
```

The result shows pages with “hong kong university” as a phrase have a higher score and thus a higher ranking. Page 46 contains the phrase “hong kong university” so its score supasses page 7 7 after the phrase scanning function has been implemented.



Conclusion

1. Strengths

Our search engine is quite robust in searching web pages within a reasonable speed. Apart from th e search results itself, the web interface is improved to better the user experience on a large scale. I t is because how users can retrieve their favorable information is very important, so we try our bes t to simplify the search experience by hiding lengthy parent links and child links. Also we let users to smoothly search in our search engine within a webpage and the fewest typing is needed for sub mitting a query.

2. Weakness

As the algorithm behind our system mainly depends on the cousine similarity measure, the ranking result does not depend on the link relationship and user preference. Therefore, the ranking results might not show the importance of pages.

3. Possible Improvement

If we could re-implement the whole system, we would have included algorithms that consider the l ink relationship between different pages, like Google PageRank algorithm to adjust the score. Thu s, more accurate results could be given.Relevant feedback may be also useful to help fine-tune the performance of our system.