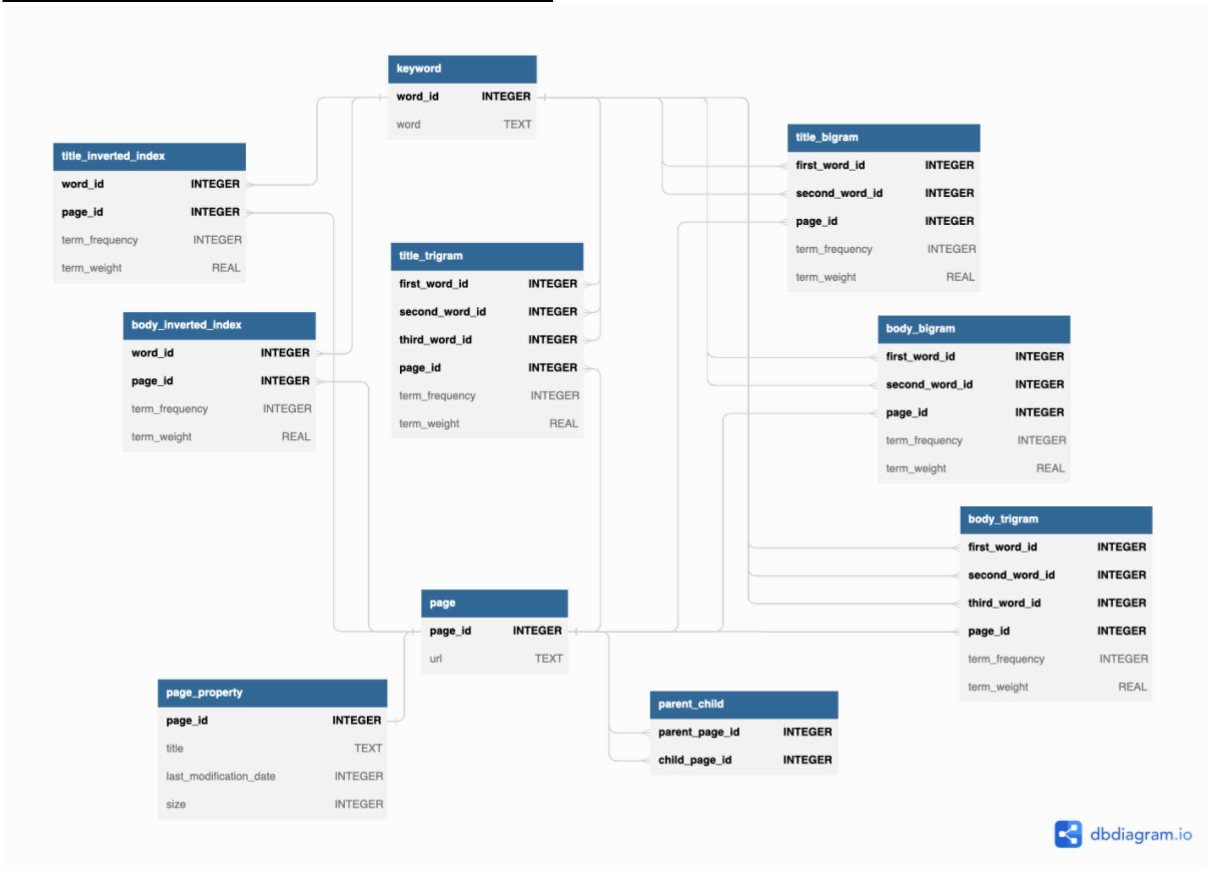
COMP4321 Project Report Phase 2
Name: Kwok Ying Kwan Kasey

<u>Overall design of the system</u>

The search engine system is designed with the combination of a SQLite database, Java classes and a JSP web server page. For the Java application Spider, it crawls pages from the root page, index all the pages and compute the term weights for all index record. It the stores all these information into the SQLite database for future retrieval. For the JSP webpage for the search engine, it inherits Java codes into HTML and allows me to call Java class functions, which parse query results, compute the similarity score and return the top 50 relevant pages to the JSP engine to display the result in the web server.

<u>File structures used in the index database</u>



keyword	
word_id	INTEGER
word	TEXT

title_inverted_index	
word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

body_inverted_index	
word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

title_trigram	
first_word_id	INTEGER
second_word_id	INTEGER
third_word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

title_bigram	
first_word_id	INTEGER
second_word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

body_bigram	
first_word_id	INTEGER
second_word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

body_trigram	
first_word_id	INTEGER
second_word_id	INTEGER
third_word_id	INTEGER
page_id	INTEGER
term_frequency	INTEGER
term_weight	REAL

page	
page_id	INTEGER
url	TEXT

page_property	
page_id	INTEGER
title	TEXT
last_modification_date	INTEGER
size	INTEGER

parent_child	
parent_page_id	INTEGER
child_page_id	INTEGER

There are a total of 10 tables, where 6 of them are table for indexes, including inverted index, bigram and trigram tables for each of title and body. The page, page property and parent child relationship tables define the main characteristics of the page. While the table keyword reduces the data redundancy of storing the same words over and over in 6 different index tables. A new field "term_weight" is added as compared to Phase 1 because it is found that term weight would not change according to the query but to the current status of the database, so storing it permanently can significantly reduce the time used for computing similarity scores.

Algorithms used

Computing similarity scores

The similarity score used for ranking the pages aim to rank matches in title more than that in body, and matches trigram > bigram > single word (inverted index). First of all, the term weights are calculated when the pages are being indexed. The term (inverted index, bigram or trigram) weight equation is given by

$$Term\ weight\ w_{ij} = \left(0.5 + 0.5 \times \frac{tf_{ij}}{\max_k(tf_{ik})}\right)\log_2\left(1 + \frac{N}{df_j}\right)$$

Where $tf_{ij} =$ frequency of term j in document I,

$df_j =$ the number of documents containing term j,

$\max_k(tf_{ik}) =$ the maximum term frequency of all terms in document I, and

$N =$ the total number of documents in the system.

The term weight are calculated separately for body inverted index, bigram, trigram, and title inverted index, bigram and trigram respectively.

Then, cosine similarity score is computed separately among all the 6 index tables as well. The cosine similarity score equation for each table is given by

$$Sim_{index}(D,Q) = \frac{\sum_j (w_{ij} \times q_j)}{\sqrt{\sum_j {w_{ij}}^2}\sqrt{query\ length}}$$

Where $w_{ij} =$ term weight of term j in document i calculated above, and

$q_j =$ query j's binary weight, i.e. if it is in the query then it is 1, else 0.

Lastly, 6 similarity scores are computed, which the combined similarity score will be

$$Sim(D,Q) = Sim_{bii}(D,Q) + 1.1 \times Sim_{bb}(D,Q) + 1.2 \times Sim_{bt}(D,Q) + 1.8 \times Sim_{tii}(D,Q) + 1.9 \times Sim_{tb}(D,Q) + 2.0 \times Sim_{tt}(D,Q)$$

Where bii = body inverted index, bb = body bigram and bt = body trigram, and the other three are for title respectively. This allows title and trigram match to rank higher than others.

And this is the final similarity score of each document and query, computed mainly in the database query server.

Installation procedure

Please refer to the readme.md file in the zip folder for details.

Highlight of features beyond the required specification

1. Fast Query
   The query search is extremely fast as all the term weights are calculated and stored in the database after each time the spider crawl all pages. And the similarity scores are calculated using SQL query to avoid large memory overhead between the Java program and the database. So the query results could show up almost instantly after searching.

2. Thorough update of indexes
   I have implemented the spider crawler in a way such that during recrawling, if a child page is modified but the parent page is not, it would ignore the parent page but still crawl through all the child pages to see if anything has changed. This can ensure a thorough update of indexes instead of some pages not being indexed even though they are updated.

3. Get Similar search
   I have implemented the get similar search function for each result page. When the button is clicked, it will automatically add the page's top 5 most frequent keyword into the original query and display the result right away.

4. Enable showing more stemmed words to add to query
   On click of the "View All Stemmed Words" button, the table of all keywords in frequency descending order will show. User can simply click the "Add" button of that row to add this word to the query without initiating a new search.

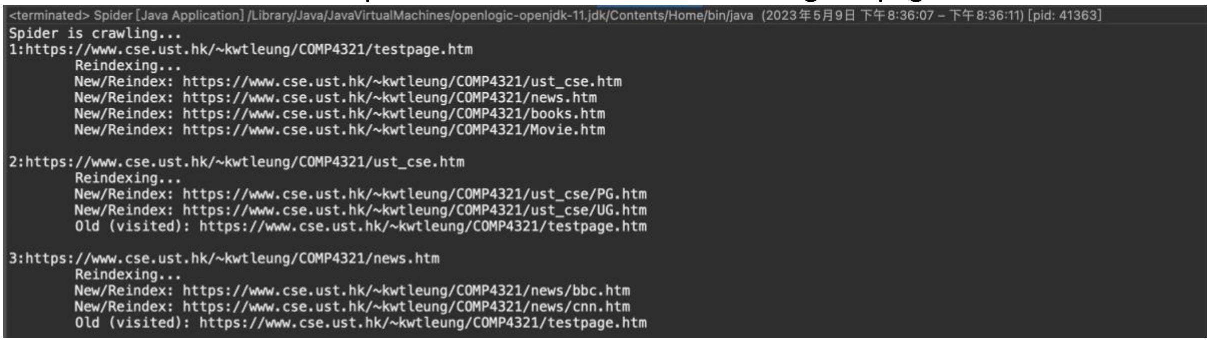
5. Elegant user interface
   I have built the web interface with simplicity and aesthetic. When the user scrolls down the page, the search box will stick on top of the website so that users can immediately check what was their current query looking like. It is especially useful when users are using the "View more stemmed words" function.

Testing of functions implemented

1. Spider
   To test the spider, you can refer to the readme.md file for the set up for running the application. If the application run successfully after clicking "Run As > Java Application", the console will show the crawling progress. The URL without indentation is the parent page URL, and those indented are the child page URLs to be processed. There are 5 types of texts printed to identify what should be done to the child links
   - If the page never existed or has been updated and needs to be reindexed, then "New/Reindex" will be shown.
   - If the page is visited during the BFS crawling process, it shows "Old (visited)".
   - If the page is not visited during BFS but it exists in the database and was not updated, then it shows "Old (no update)"
   - If the page cannot secure a SSL connection, it shows "Ignored url"

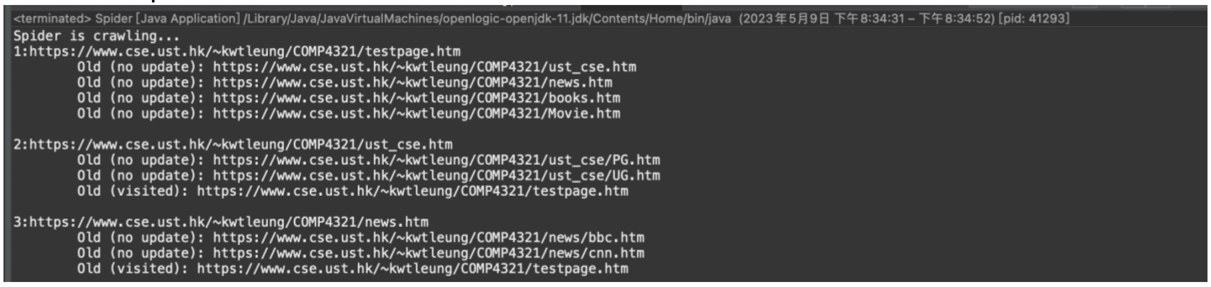
- If the page has been crawled as the child of the same parent page for the second time, then it shows “Repeated child”

This is what the console output looks for the first time crawling the pages.



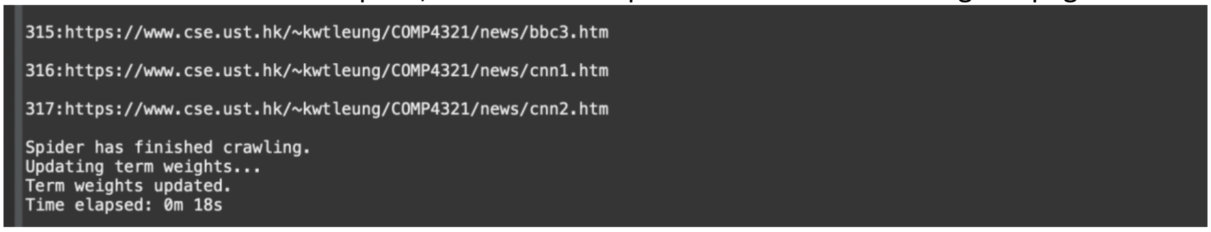
```
<terminated> Spider [Java Application] /Library/Java/JavaVirtualMachines/openlogic-openjdk-11.jdk/Contents/Home/bin/java  (2023年5月9日 下午8:36:07 – 下午8:36:11) [pid: 41363]
Spider is crawling...
1:https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
        Reindexing...
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

2:https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
        Reindexing...
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
        Old (visited): https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

3:https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
        Reindexing...
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm
        New/Reindex: https://www.cse.ust.hk/~kwtleung/COMP4321/news/cnn.htm
        Old (visited): https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
```

And this is what the console output looks like for the second time crawling the pages if there is updates done to it at all.



```
<terminated> Spider [Java Application] /Library/Java/JavaVirtualMachines/openlogic-openjdk-11.jdk/Contents/Home/bin/java  (2023年5月9日 下午8:34:31 – 下午8:34:52) [pid: 41293]
Spider is crawling...
1:https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

2:https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
        Old (visited): https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

3:https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm
        Old (no update): https://www.cse.ust.hk/~kwtleung/COMP4321/news/cnn.htm
        Old (visited): https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
```

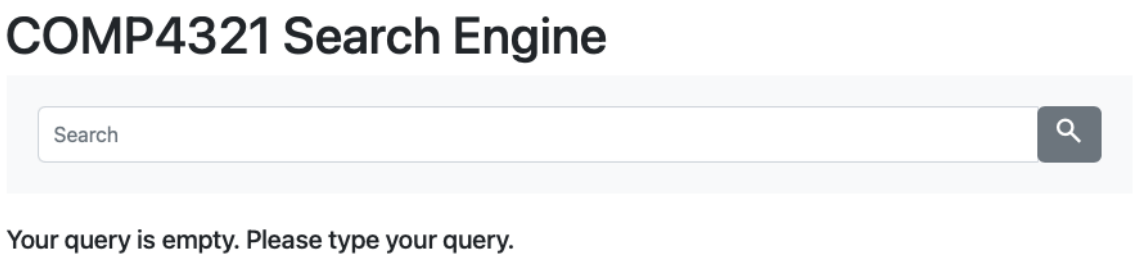
When it shows the time elapsed, it means that spider has finished crawling the page.



```
315:https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc3.htm

316:https://www.cse.ust.hk/~kwtleung/COMP4321/news/cnn1.htm

317:https://www.cse.ust.hk/~kwtleung/COMP4321/news/cnn2.htm

Spider has finished crawling.
Updating term weights...
Term weights updated.
Time elapsed: 0m 18s
```

2. Search Engine Web Interface

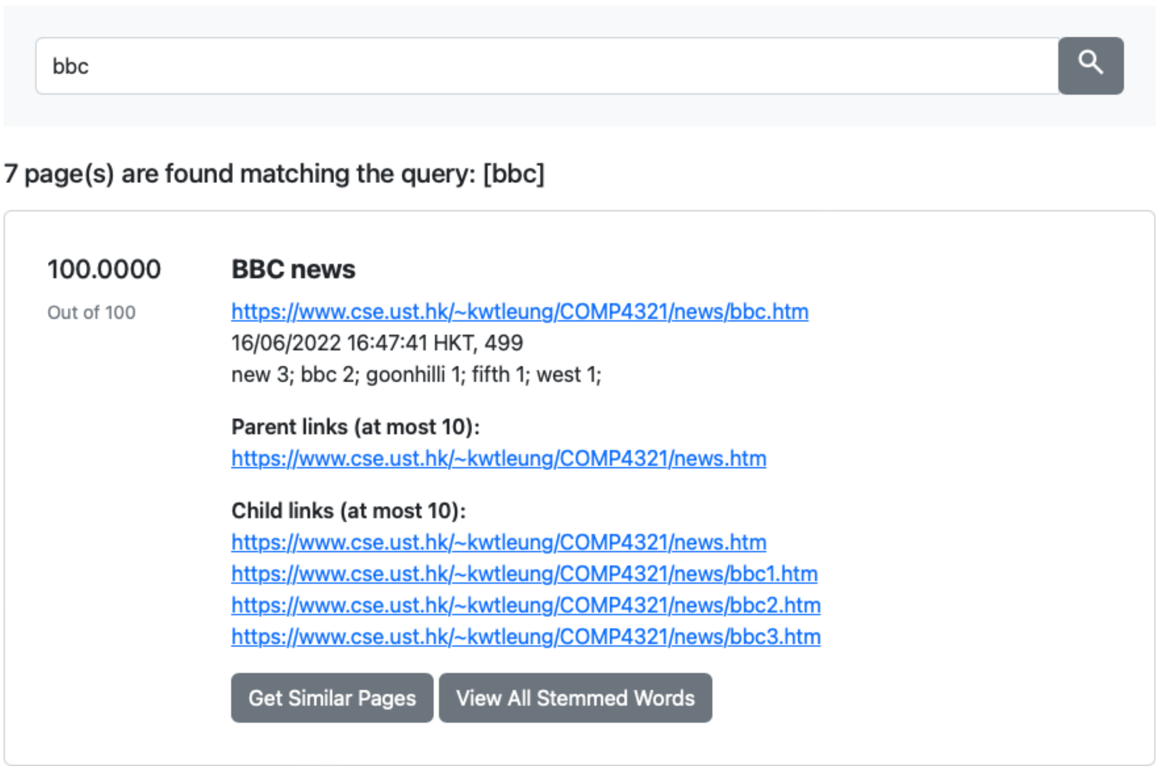
After running the JSP file on server, you can visit http://localhost:8080/WebApp/search.jsp to view the web interface. The appearance of the web interface looks like this on start.



COMP4321 Search Engine

Your query is empty. Please type your query.

If I search [bbc], then it will show the matching pages in the required format.



bbc

7 page(s) are found matching the query: [bbc]

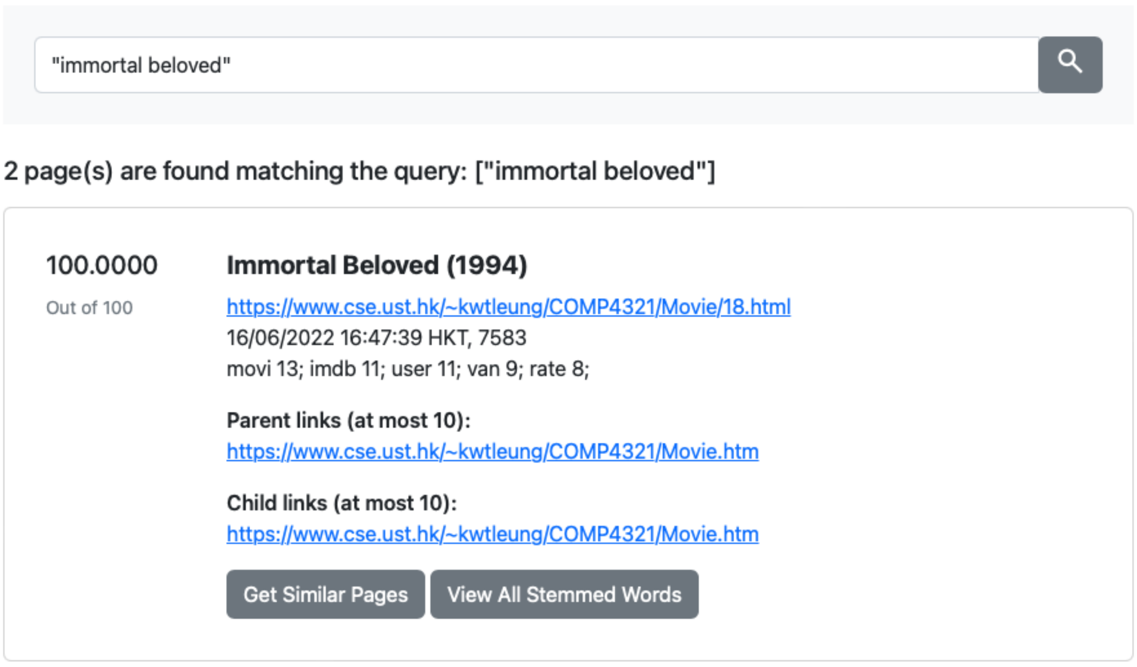
100.0000
Out of 100

BBC news
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm
16/06/2022 16:47:41 HKT, 499
new 3; bbc 2; goonhilli 1; fifth 1; west 1;

Parent links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm

Child links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc1.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc2.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc3.htm

If I enter a phrase in double quote, it can also do phrase search successfully



"immortal beloved"

2 page(s) are found matching the query: ["immortal beloved"]

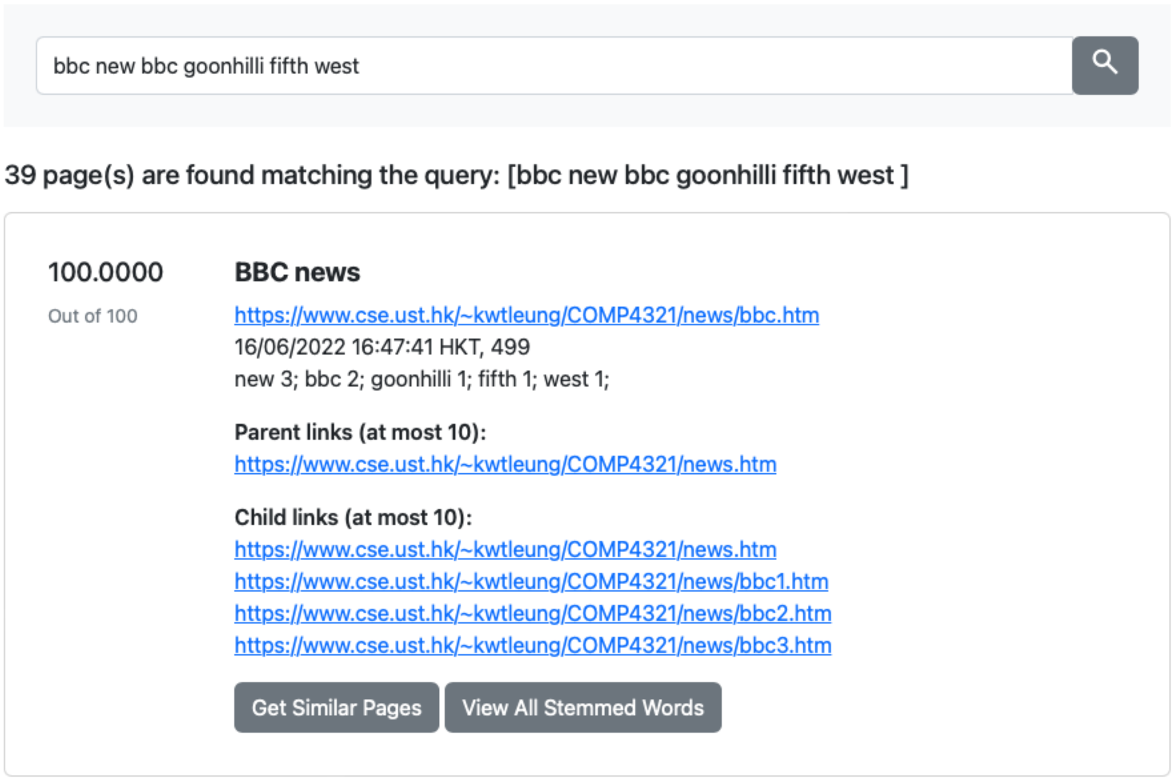
100.0000
Out of 100

Immortal Beloved (1994)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/18.html
16/06/2022 16:47:39 HKT, 7583
movi 13; imdb 11; user 11; van 9; rate 8;

Parent links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Child links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

If I click "Get Similar Pages", the top 5 most frequent keywords of the page will be added to the original query and display the new search result immediately.



bbc new bbc goonhilli fifth west

39 page(s) are found matching the query: [bbc new bbc goonhilli fifth west]

100.0000

Out of 100

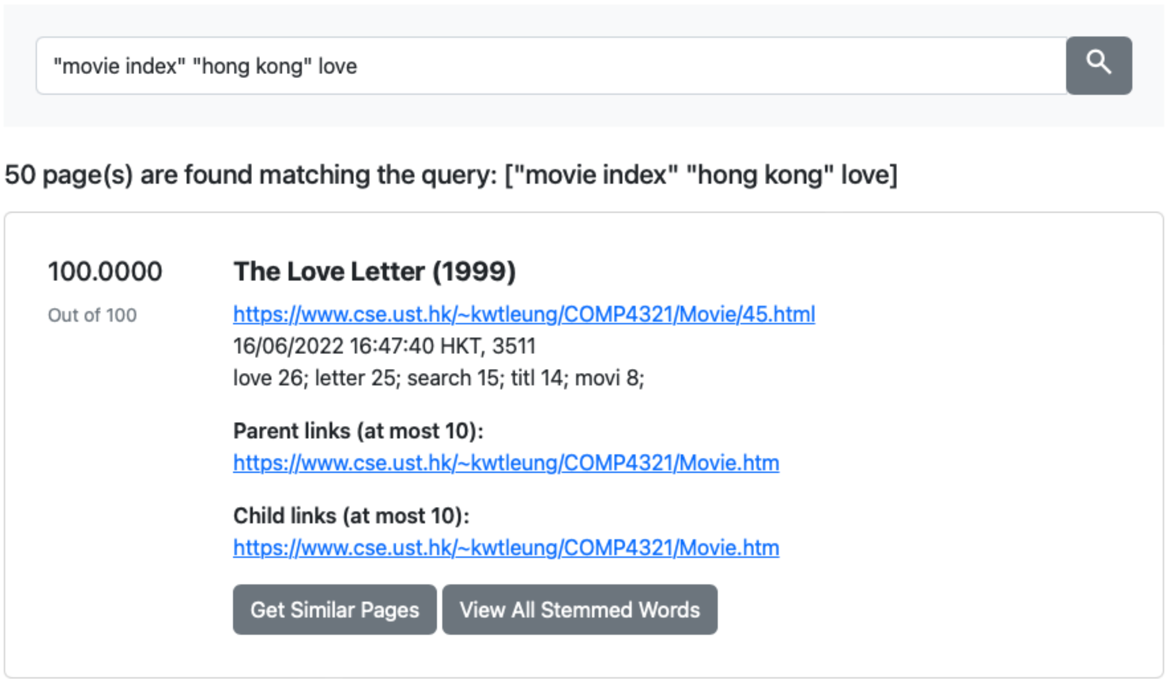
BBC news

https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm
16/06/2022 16:47:41 HKT, 499
new 3; bbc 2; goonhilli 1; fifth 1; west 1;

Parent links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm

Child links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc1.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc2.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc3.htm

If I enter multiple phrase and words, the search is also possible.



"movie index" "hong kong" love

50 page(s) are found matching the query: ["movie index" "hong kong" love]

100.0000

Out of 100

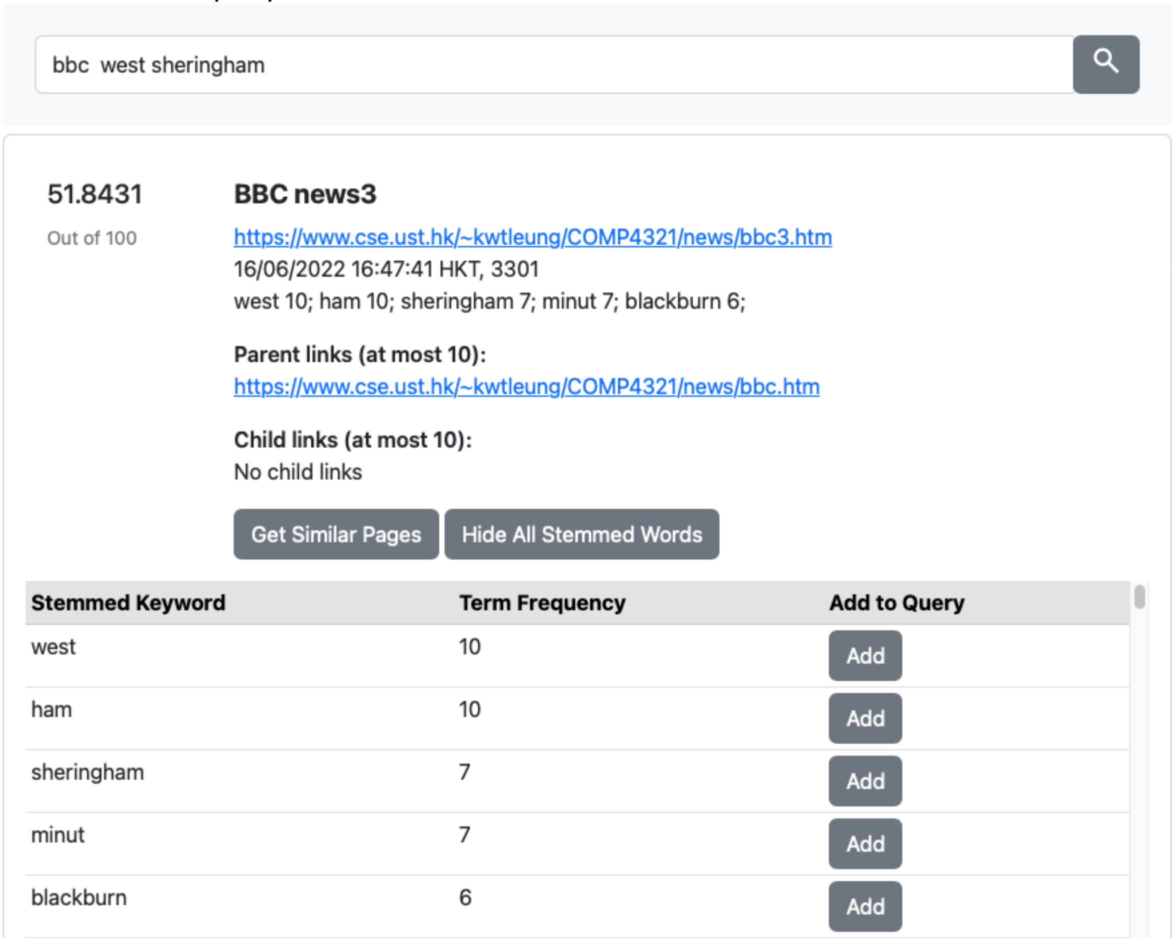
The Love Letter (1999)

https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/45.html
16/06/2022 16:47:40 HKT, 3511
love 26; letter 25; search 15; titl 14; movi 8;

Parent links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Child links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

If I click "View all stemmed words", the table of stemmed words and their corresponding frequency will be shown in decreasing order of the frequency. If I would like to add the keyword to the query, I can simply click "Add", and it will add the word to the query.



bbc west sheringham

51.8431
Out of 100

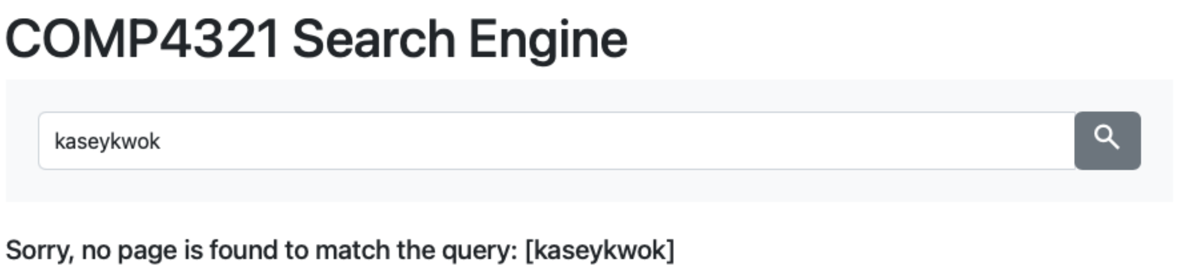
BBC news3
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc3.htm
16/06/2022 16:47:41 HKT, 3301
west 10; ham 10; sheringham 7; minut 7; blackburn 6;

Parent links (at most 10):
https://www.cse.ust.hk/~kwtleung/COMP4321/news/bbc.htm

Child links (at most 10):
No child links

Stemmed Keyword	Term Frequency	Add to Query
west	10	Add
ham	10	Add
sheringham	7	Add
minut	7	Add
blackburn	6	Add

Now if I search my own name, it will show that there is no page found.



COMP4321 Search Engine

kaseykwok

Sorry, no page is found to match the query: [kaseykwok]

<u>**Conclusion**</u>

The strength of this whole system is that it can display search results very quickly, and crawling of pages is done in a careful manner to make sure all updated pages are reindexed. The user interface is clean and modern, which is much more attractive than the usual HTML components. And the database is well-structured to facilitate the quickest and minimize data redundancy among the tables.

However, the weakness of this system is that the crawling of pages is quite slow without multi-threading. For the web interface, there is no way to keep track of the query. It does not allow user to reuse the query. Also, PageRank was not implemented in this submission due to the lack of time, so the page ranking is lacking the authority.

If I could re-implement the whole system, I would implement cookies to store the query and allow users to review their past queries on the web interface. Another thing is that I would like to implement PageRank and add it to the similarity score so that the ranking of the page can better reflect their authorities.

I believe that it would be interesting to add query suggestion while user enters the query. If possible, it would also be nice to include a button such that users can initiate the crawling of webpages themselves. And I would suggest adding a user login system such that recent queries can be stored permanently in the database, and when the user logs in the next time, I can still show her the past queries she has asked. I think personalizing the user experience will be nice, just like how Google is currently.