# Comp4321 project final submission

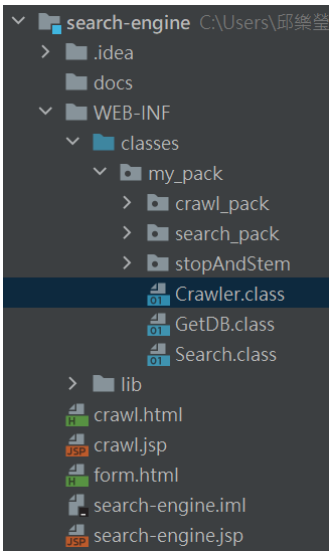
Chiu Lok Ying, SID: 20586347

## Overall design of the system

The crawler crawls information of webpages recursively using a breadth-first strategy. It stores the information including inverted index, forward index, parent and child links, title, last modified date, size, etc. into the corresponding database files.
After crawling the pages, a user can submit a query to the system simply through the user interface. The system then ranks the pages stored in the database by their similarity to the query, it returns the ranked similar pages and shows their information to the user.

To clear the database files, the user needs to delete them manually.

## File structures




*crawl_pack* contains classes that help performing crawling.
*search_pack* contains classes that help performing searching.
*stopAndStem* contains classes that help performing stopping and stemming.

*lib* contains the jar files that this program depends on.

crawl.html is used to crawl the pages by a click on the button.
form.html is used to submit query and get the related pages from the crawled documents.

**Database Schema:**

| Database file | Key | Value | Description |
|---|---|---|---|
| forwardIndex | doc_id | Posting(word_id, freq) | Use a document's ID as the key to get the words appear in this document and their frequencies. |
| invertedIndex (for page body) | word_id | Posting(doc_id, freq) | Use a word's ID as the key to get the webpages that contain this word in their body and its frequencies.. |
| titleInvertedIndex | word_id | Posting(doc_id, freq) | Use the word's ID as the key to get the webpages that contain this word in their title and its frequencies. |

| | | | |
|---|---|---|---|
| Page | pageId | PageInfo | Use a page's ID as the key to get the info (including title, last modified date, size, body max tf, and title max tf) of the page. |
| pageID_url | page_id | url | Use a page's ID as the key to get the corresponding url. |
| wordID_word | word_id | word_df | Use a word's ID as the key to get the corresponding word and its df. |
| childToParent | c_url_id | parentSet | Use the children page's id to get its set of parent pages. |
| parentToChild | p_url_id | childSet | Use the parent page id to get its set of children pages. |

## Algorithms

1. To deal with pages that redirect the user to another page:
   When performing recursive crawling, before crawling each page, the crawler checks if it redirects the user to another page. If the crawler sees a webpage (A) that redirects the user to another page (B), it crawls page B and marks both pages as expanded, so that neither page A nor page B will be crawled in the same run. Also, pages that are permanently moved to another URL will not be crawled into the database.

2. To perform crawling recursively:
   These are the tools used to help perform recursion and their data structures and purposes, the pages are represented by their ID in the implementation:
   For simplicity, I assume there is not redirection problem for now.
   a. *url_q*: a first-in-first-out queue that stores the pages that are waiting for being crawled (nodes waiting for being expanded).
   b. *crawled*: a set storing the crawled pages (expanded nodes).
   c. *seen*: a set of pages that have been in *url_q* before or are still in *url_q* (nodes waiting for being expanded + expanded nodes).

   First, the starting page is added into *url_q* and *seen*. It is then taken out from *url_q*, crawled, and added to *crawled* (marked as expanded). Afterward, its children links that are not in *seen* are added into *url_q* and *seen*.
   The first page in *url_q* is then crawled and the same actions done to the previous page are repeated on this page. The same procedures go on and on until the number of pages is met or there is no more page in *url_q.*

3. To crawl a page:
   Before crawling a page, it first checks if the page's information exists in the database.
   a. If it exists, it checks the last modified date of the webpage and compares it with that in the database.

- ✧ If it is modified after the last crawl, then it crawls the page and updates all the information.
- ✧ Otherwise, it does not continue crawling.

b. If it does not exist, it crawls the page.

4. To store words from a page:
   First, it takes all the one-gram tokens excluding stopwords from the page's words, and performs stemming to those one-grams. (Some words become an empty string after stemming, those are also excluded.)
   2-grams are formed from the one-grams without stopping and stemming. It checks if the 2-grams contain stopwords or words that become an empty string after stemming. If it does, discard the 2-gram.
   3-grams are formed in a similar way as 2-grams.

5. To rank the pages:
   a. Convert the query into 1-grams, 2-grams, and 3-grams, and extract the phrases enclosed by quotation marks.
   b. Calculate the cosine similarity between a page and the query:
      - ✧ First, set the similarity score = 0.
      - ✧ Iterate through all the x-grams of the query, if the page contains the x-gram in the body, add the word's tfidf to the similarity score, if the page contains it in the title, multiply the word's tfidf by 10 and add the product to the similarity score. Otherwise do nothing.
      - ✧ Iterate through all the phrases extracted from the query. If the page contains the phrase in its body, add the word's tfidf to the similarity score. If the page contains it in the title, multiply the word's tfidf by 10 and add the product to the similarity score. Otherwise do nothing.
      - ✧ TF*IDF score of a term here is calculated as:
        - ■ TF (term frequency, normalized by the max tf in the document)
        - ■ IDF (inverse document frequency): log(1/n), where n is the number of documents that contains the specific term. It was supposed to be log(N/n), where N is the total number of documents, however, since N is a constant number, I ignored it for simplicity.
      - ✧ The final score of the page is divided by number of different x-grams of the page*number of different x-grams of the query.
   c. Put the pages into a priority queue when the priorities are the similarity scores of the pages.

## Installation procedure

1. Unzip the project.
2. Put the whole project under "webapps" of tomcat.

## Highlight of features beyond the required specification

1. It deals with the redirection problem to avoid crawling deleted or moved pages, which is unmeaningful.

## Testing of the functions implemented

1. To see if the database files are storing the expected data, I printed out the information stored to txt files. Since the documents are too long, I am only showing small parts of them.

inverted_index - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
48 = doc-851779856 1 doc-78792779 1 doc-1464143762 1 doc-1867925106 1 doc-1839295955 1 doc-751388217 1
49 = doc1530562810 10 doc-363883119 1 doc600765933 1 doc872078288 1 doc929336590 1 doc957965741 1 doc98659489
107421930 1 doc-78792779 3 doc655602704 2 doc712861006 4 doc770119308 1 doc798748459 1 doc827377610 3 doc8846
1 doc196346996 1 doc224976147 1 doc253605298 1 doc282234449 1 doc310863600 1 doc-1896554257 2 doc-1867925106
2 doc1108911952 1 doc1137541103 2
50 = doc1530562810 13 doc-363883119 1 doc600765933 1 doc872078288 1 doc900707439 1 doc957965741 1 doc10152240
881291 1 doc1743510442 2 doc-1864357230 2 doc-1807098928 5 doc-1778469777 1 doc-1721211475 1 doc-1635324022 3
5 1 doc1911122826 1 doc-1696744846 7 doc-1668115695 1 doc-1639486544 1 doc-1610857393 1 doc-1553599091 2 doc-
51 = doc1530562810 3 doc1101111496 1 doc1267966610 4 doc1325224912 2 doc1439741516 1 doc1525628969 1 doc-2110
6 1 doc-951792274 1 doc-808646519 1 doc-35659442 4 doc851844239 1 doc909102541 1 doc1023619145 1 doc165346046
52 = doc1530562810 1 doc900707439 1 doc1015224043 1 doc1101111496 1 doc1267966610 2 doc1325224912 1 doc-12233
doc1767977071 1 doc1853864524 2 doc-1696744846 1 doc-1639486544 1 doc-809241165 1 doc-666095410 1 doc-6088371
```

forward_index - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
doc7094674 = -275648109 1 2086319711 1 3715123 2 -52518701 1 1310559763 1 110327427 1 -1415026758 9 -24915425
6 -2013696117 1 -537404540 1 2075333575 6 118 2 -2139111416 1 153351508 1 60358627 1 -303887976 1 -1227952974
1 658301402 1 1892800056 1 -1421971500 1 -536925419 1 1929323028 1 1596380104 1 3208415 2 1516263 1 -20084652
2840 3 1826455473 1 1816727847 1 3198785 2 739057020 2 -1571173592 1 -537080177 1 -1668204059 1 3445 1 96631
1 -1268958287 2 -1184792947 1 50511086 1 1984153006 1 382823231 1 1729655542 1 1057052910 1 -1856867651 1 166
doc25166736 = -275648109 1 -1141238786 6 2086319711 1 3715123 2 1310559763 1 110327427 1 -249154254 1 -537659
```

parent_child_link - 記事本

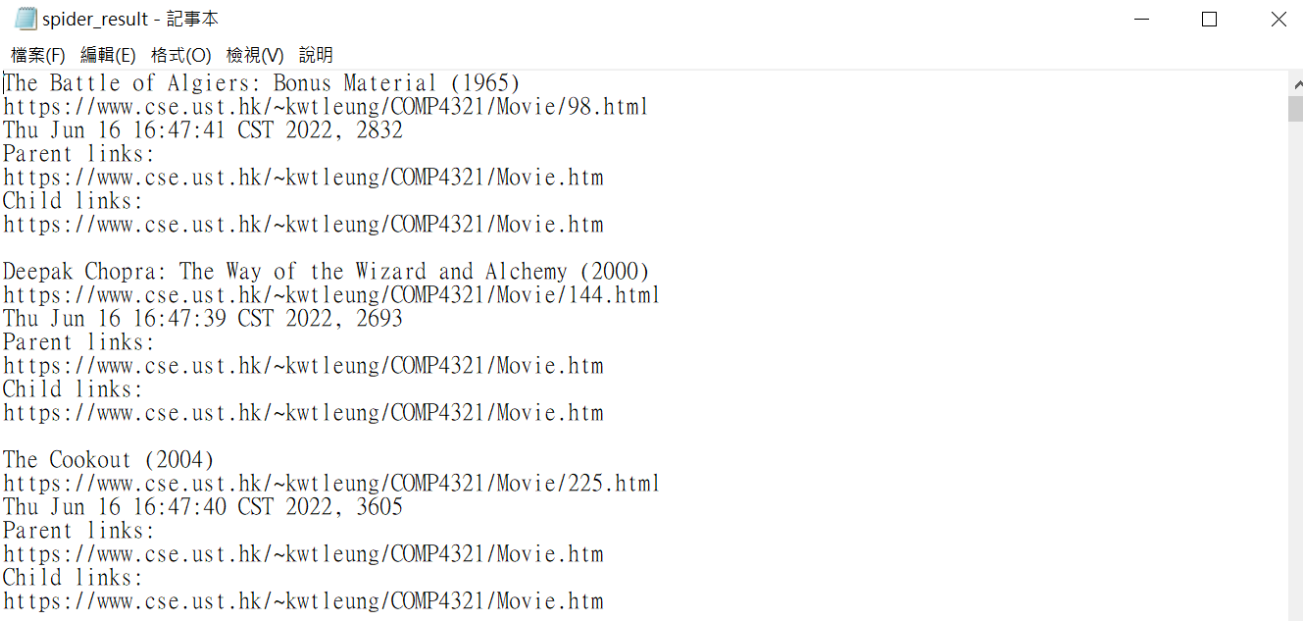
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
1439146870 = 1530562810
1468370667 = 1530562810
1467776021 = 1530562810
1496999818 = 1530562810
1496405172 = 1530562810
1525628969 = 1530562810
1525034323 = 1530562810
1530562810 = 107486313 110459543 -894533972 -1496340789 794585937 192779120 -576640081 -994330965 35723825 25
839 -135456435 -833707794 49633365 168312491 -1782037653 -1223364173 1543106385 -780017368 253605298 -1696744
0 608901491 -3462415 -1967722099 1714881291 221408271 -980421425 -193309383 -808646519 1108911952 1023619145
1005008957 -751982863 -1052183913 741490157 -1911058443 909102541 1857432400 1772139593 -1009050576 -19963512
1543106385 = 1530562810
1553663474 = 1530562810
1571735536 = 1530562810
1582292625 = 1530562810
1609635318 = -507028874 1005008957 -363883119
```

pageid_url - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

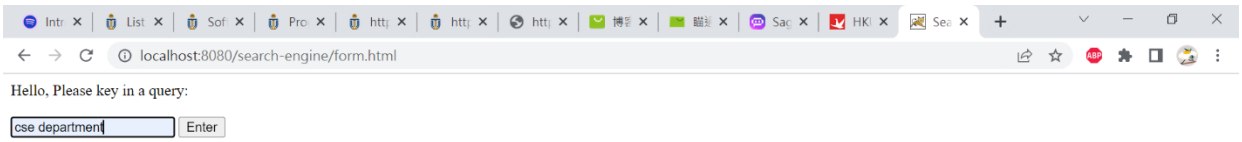
```
7094674 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/98.html
25166736 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/144.html
21598860 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/225.html
21004214 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/270.html
35723825 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/99.html
50228011 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/226.html
49633365 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/271.html
53795887 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/145.html
53201241 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/190.html
82425038 = https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/146.html
```




2. I tested if the database files are not generated again if no new information should be crawled when I run the crawler each time by checking the last modified time on my device.

3. Tests of the results of search:

Query: cse department




Result:

The query you entered is:
cse department

Below are the relevant pages:

0.168519
CSE department of HKUST
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Sun Apr 30 18:26:32 CST 2023, 392
admis 2; ug_admis_back 1; admis_ug 1; hkust_pg 1; hkust_pg_admis 1
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

0.004981
Test page
https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Sun Apr 30 18:26:31 CST 2023, 603
new 2; here 1; crawler_befor 1; list_new 1; book 1
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link(s):
None

0.000078
PG
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Sun Apr 30 18:26:35 CST 2023, 3267
postgradu 9; comput 8; admis 8; program 8; applic 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

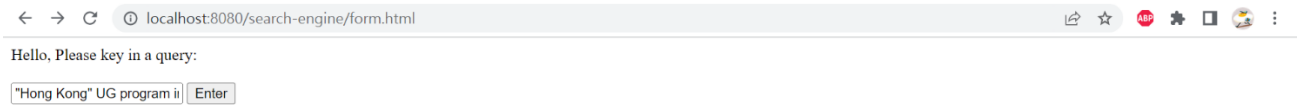
0.000055
UG
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
Sun Apr 30 18:26:35 CST 2023, 4070
program 16; comput 13; comput_scienc 7; scienc 7; student 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

0.000006
The Big Clock (1948)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/261.html
Sun Apr 30 18:29:16 CST 2023, 6735
imdb 11; user 10; movi 9; rate 8; plot 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

0.000003
Chappelle's Show: Season 1 (2003)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/223.html
Sun Apr 30 18:28:50 CST 2023, 7877
chappel 17; rate 13; episod 13; imdb 11; user 11
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

0.000002
Reservoir Dogs (1992)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/175.html
Sun Apr 30 18:28:20 CST 2023, 11108
film 18; the 13; rate 12; i 11; imdb 11

Query: “Hong Kong” UG program in “Computer Science”




Result: (only showing the top five since the full result is too long)

The query you entered is:
"Hong Kong" UG program in "Computer Science"

Below are the relevant pages:

0.002116
UG
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
Sun Apr 30 18:26:35 CST 2023, 4070
program 16; comput 13; comput_scienc 7; scienc 7; student 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

0.001473
PG
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
Sun Apr 30 18:26:35 CST 2023, 3267
postgradu 9; comput 8; admis 8; program 8; applic 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

0.001157
CSE department of HKUST
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
Sun Apr 30 18:26:32 CST 2023, 392
admis 2; ug_admis_back 1; admis_ug 1; hkust_pg 1; hkust_pg_admis 1
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

0.000132
The Tricky Master (2000)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/142.html
Sun Apr 30 18:27:56 CST 2023, 6415
movi 16; wong 13; imdb 11; user 10; rate 8
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

0.000125
Sex and the Beauties (2003)
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/59.html
Sun Apr 30 18:27:08 CST 2023, 6428
movi 17; imdb 11; user 10; rate 8; titl 7
Parent link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm
Child link(s):
https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

## Conclusion

If I were to re-implement the system, I would store the positions of words in the inverted index instead of the frequencies of x-grams. This approach would avoid the issue of storing many meaningless x-grams, resulting in a table with numerous unnecessary entries. Furthermore, given that users are expected to submit queries with phrases enclosed in quotation marks, storing positions of words would align better with the design. The current design, which stores frequencies of x-grams, works without quotation marks because it uses the phrases in the inverted index as a key.

Additionally, to simplify the code and make it easier to manage, I would create a super class for the classes that perform similar tasks in managing the database files. This would lead to cleaner code and more efficient management.

Overall, this project was a nice and fun one that helps me with practice designing the system, coding in Java, and understanding how a search engine works in general.