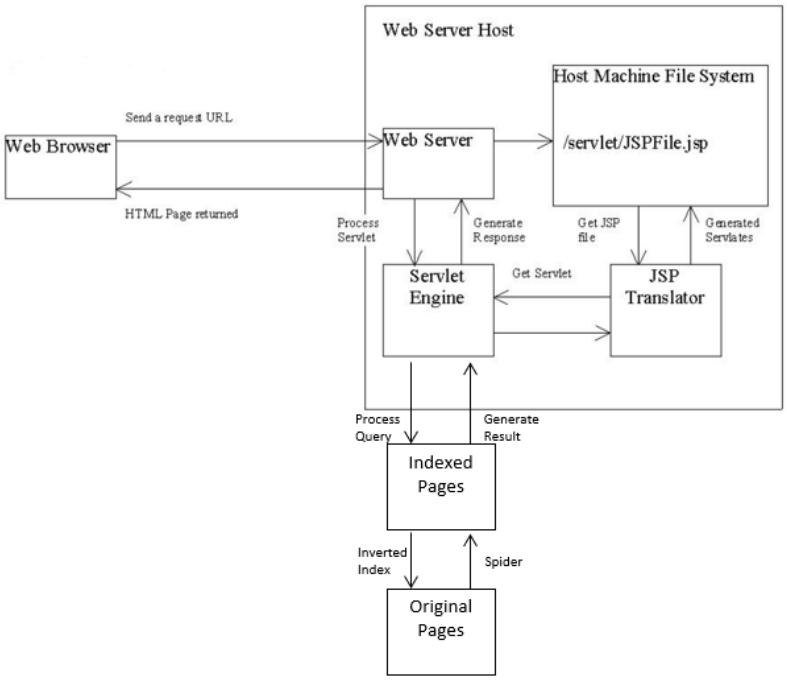
COMP 4321 Project
Final Report

CHAN, Hiu Lok Felix **SID:20177897**
OR, Ka Po **SID:20342179**
WANG, Yuan **SID:20175825**

1. Overall Design of the System



The design of our system is based on the above structure. JSP is adopted in our design as a server-side language to generate pages for our search engine. The design of the pages, as required, can achieve the following functions. There is a browser page written in htm which serves as a user interface and provides an input box for users to input queries. The web server host then intakes the inputs, retrieves queries from them and performs stop-word removal and stemming on the queries. It also retrieves indexed pages from database which is already crawled at the time of file compilation. After getting indexed pages according to document similarity which is based on cosine similarity measure, the web server host will also calculate page scores of queries and finally output the results.

2. File Structures used in Index Database

The database that we got after crawling is connected by different indexes. There are two types of indexers adopted in index database, keyword-based indexers and link-based indexers. Keyword indexers fall into more detailed categories which will be explained further. All indexers are designed based on hashing.

2.1 Keyword-based Indexes

Forward Index: Index the pages into keywords and their frequencies for further implementation of search engine.

- Page-ID to word with frequency (Variable Name: indexToWordWithFrequency)

Inverted Index: Index back to the pages so that keywords can be used to retrieve original pages.

- Word to Page-ID and position (Variable Name: indexToDocPos)

Mapping Indexes: Following indexes serve to facilitate our program by understanding our documents better and by enabling searching by different properties.

- Page-URL to Page-ID (Variable Name: visitedPage)
- Page-ID to Page-URL (Variable Name: indexToPageURL)
- Page-ID to Page-Title (Variable Name: indexToTile)
- Page-ID to Last-modified-date (Variable Name: indexToLastModifiedDate)
- Page-ID to Page-size (Variable Name: indexToPageSize)

2.2 Link-based Indexes

Following indexes are set up for potential calculations for PageRank.

- Parent-Page-ID to Child-Links (Variable Name: indexToChildLink)
- Child-Links to Parent-Links (Variable Name: linkToParentLink)

3. Algorithms Used

3.1 For Database

3.1.1 Links

Database file is formed by crawling at the time when the java program is compiled. The spider first initiates the crawler which starts with http://cse.ust.hk and then uses a java library to identify the links contained in the page. According to definition, these links are child links of the original link and they thus are recorded in database of indexToChildLink. Parent links cannot be found as straightforward as child links. So, an index which keeps track of which pages the link has visited is introduced to solve the problem. With the index, a link can be referred back when it has pointed to some other links. A parent link is thus identified and kept in the database. This index also helps to solve the potential cyclic links problem as visited pages will not be considered in the following crawling.

3.1.2 Keywords

After crawling, the spider performes stop-word removal and stemming to the pages crawled using Porter's algorithm. It looks at the words one by one to identify if it is a stop word or if it needs to be stemmed. At the same time, the spider also counts the frequency of the particular word and records the position of it. Therefore, word frequency and positions of each word is also found and recorded. The details are then saved in the inverted file.

3.1.3 Number of links in database

To get more links as required, such algorithm is recursively applied to the child links to get as much links as possible. However, the final decision is to stop the crawling process once 1000 links have been crawled. This number is chosen after testing different stop lines ranging from 100 to infinity. The reason why it is set to 1000 at the end is because with the increase of the number, pages being crawled have decreasing relation with cse.ust.hk. This will decrease the Recall and Precision of the searching result and is not what expected to be seen.

3.2 For Queries

3.2.1 Transformation

After users submit their inputs through the interface, the inputs are being processed into queries to be used for search engine. The inputs are first stop-word removed and stemmed so as to match with the keywords saved in the database.

3.2.2 Retrieval function

The transformed queries are then being compared with inverted files to retrieve the most related file. Whether the queries are related to a page or not is decided by similarity. For our design, we adopted cosine similarity for the comparison. And the weighting formula involved is based on tfxidf/max(tf).

Besides, to emphasize the importance of those links which have anchor text, we add 1 to the final similarity to the links which contain exact words that we are looking for in the title. As each word matched in anchor text accounts for 1 point, the maximum for final similarities could be more than 2.

3.3 User interface

The pages are mainly using JSP as its foundation. The first page contains an input box which can get inputs from users. The information is then transformed and compared within JavaServer Pages (JSP) by the embedded functions. After getting the score for comparison between queries and inverted files, JSP connects to the database to retrieve the desired pages according to the scores calculated. It then displays the outputs calculated or fetched on a final page to users. The outputs that we include in our final page are keywords, similarity score, URL title (hyperlinked), parent links and child links. The keywords shown here is untransformed ones. They are shown unprocessed because transformed ones are not user-friendly (not easy for users to understand).

4. Installation procedure

In personal computer:

1. Download Tomcat 7.0.86 and setup permission of Tomcat's files
2. Install Java EE - Eclipse IDE for Java EE Developers
3. Create a **Java EE Dynamic Web Project** in Eclipse and add the path of the downloaded tomcat folder as **New Runtime**
4. Drag all *.jar files and drop them in "WebContent/WEB-INF/lib" node in the Eclipse [Project Explorer] window
5. Drag all "stopword.txt" file and drop them in "WebContent/WEB-INF/lib" node in the Eclipse [Project Explorer] window
6. Drag "IRUtilities" and "searchingRelated" **Java package folders** and drop them in "Jave Resources" node in [Project Explorer] window
7. Set "**Publish module contexts to separate XML files**"
8. Specify the server path (i.e. Catalina.base) and deploy path as "**Use Tomcat Installation**"
9. Run spider.java in eclipse to generate **"database.db"** and **"database.lg"**
10. Drag **"database.db"** and **"database.lg"** and drop them in "WebContent/WEB-INF/lib" node in the Eclipse [Project Explorer] window
11. Export the project as .war file

In VM:

12. Upload Tomcat 7.0.86 folder to VM and setup permission of Tomcat's *.sh files by **"chmod 700 *.sh"** command
13. Upload the .war file to VM and place it in the webapps directory in Tomcat's distribution folder
14. Change directory to tomcat-7.0.86/bin and run the command "./startup.sh" to run the tomcat
15. The WAR file content will be extracted into the folder by Tomcat
16. Set the permission of all .jsp file by "**chmod 700 *.jsp"**
17. Set the permission of the lib files by **"chmod 700 *.db"**, **"chmod 700 *.lg"** and **"chmod 700 .txt"**

5. Highlight of unique features

In our project, we implemented some unique functions that could be considered as bonus.

Search similar page

Under each score of similarity page, there is a button which enable users to find similar pages of the pages being found. The similarity is decided by the top 5 most frequent keywords of the found pages. By clicking the button, the program will fetch the top 5 keywords as shown on result page and use the keywords to compare with the pages back in database. The new result page is thus made up of the similar pages of the 5 appointed keywords.

Search by stemmed keywords

There is a dropdown list on the right side of the result page which contains all the stemmed keywords. By selecting the desired stemmed keywords and press the submit button, the keywords will be sent as an input to JSP and be regarded as the queries.

Using + to replace "" function

In the input page, + can be used to indicate that the words connected together by plus sign is a phrase, which is the same function as "".

Button back to input page

There is a return button on the top of the result page so that users can easily go back to the input page to initiate another new search.

6. Testing of functions

As most of the functions have been detailed explained and discussed in above sections, the main contents for this section will be demonstrated mainly through cap-screens of our project. Keywords with top 5 word frequency are shown together with the frequency number. Besides, parent links and child links are also displayed in outputs.

Search Engine: http://143.89.130.11:8080/comp4321-testing1.0/comp4321.html

Example 1

Inputs: *the hkust "residential hall"*

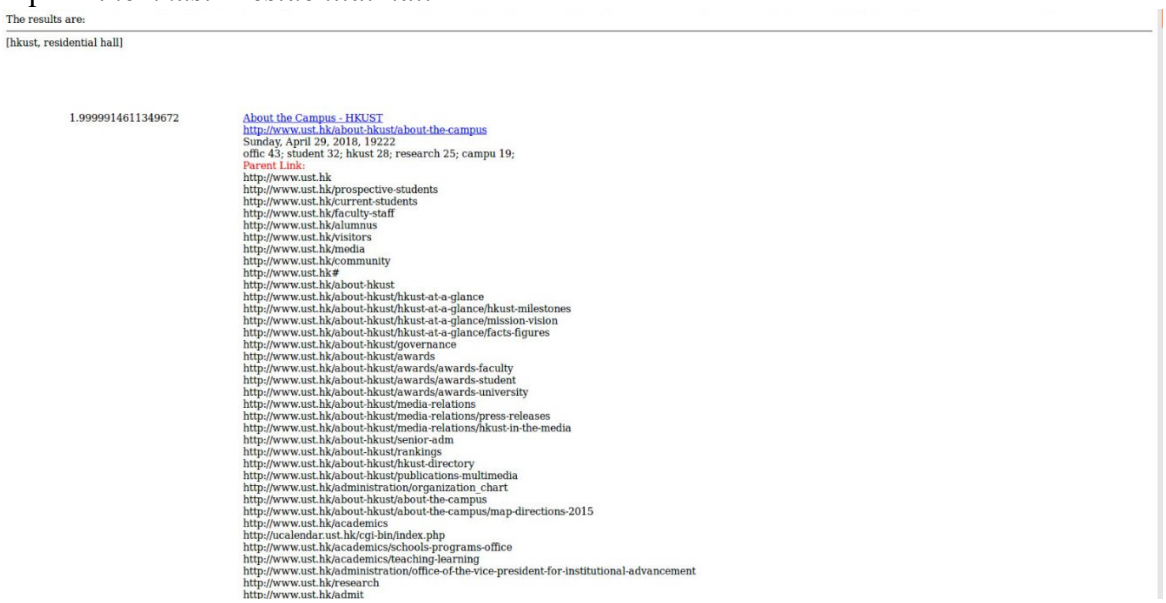


Figure_1 Outputs for the first inputs

Keywords identified (untransformed): hkust; residential hall
Highest Similarity (after adding anchor text weight): 1.999991461134967 – as "hkust" appears
Last modified date: Aprill 29, 2018
Size of page: 19222

Example 2

Inputs: *hong+kong+student*

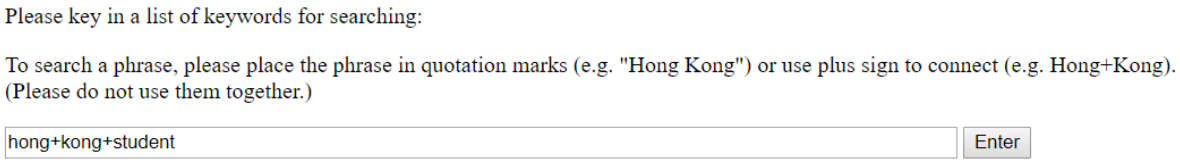


Figure_2 Input Page

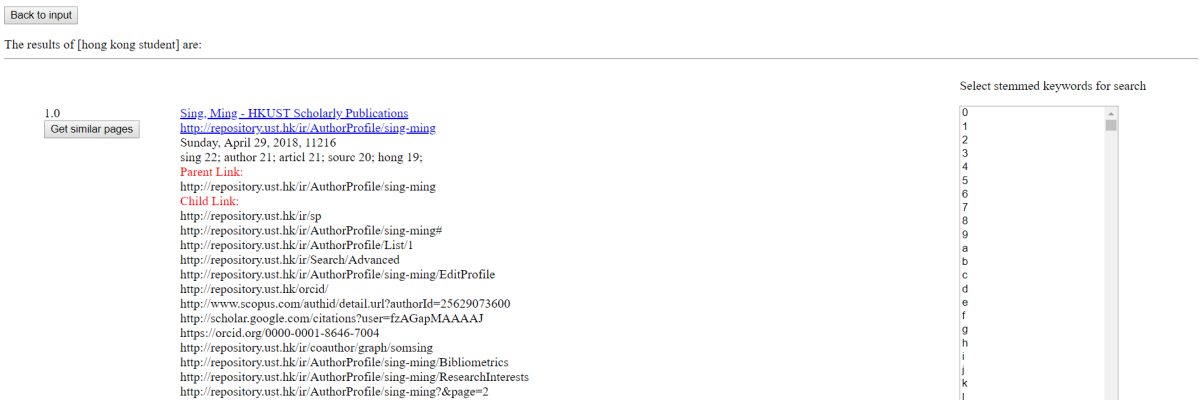


Figure_3 Output page for second try

Keywords identified (untransformed): hong kong student
Highest Similarity (after adding anchor text weight): 1
Last modified date: Aprill 29, 2018
Size of page: 11216

On this output page there are the new functions that we have adopted. The Back to input button redirect users back to the input page. The Get similar pages button enables users to choose their interested topics to search similar pages. And the stemmed keywords list provide users with a chance to know what kind of keywords the database has. Users can therefore choose their most interested topics accordingly.

7. Conclusions

7.1 Strengths

Our program is able to deal with phrases being specified in double quotes. This means users can specify the combination of words at their call. Besides, our program is also able to handle URL inputs, which provides flexibility to users.

As the difference between cosine similarities could be small among for different links, so we add a large portion to the URL which include desired words in their title. Therefore, the first link output of our program is always the most related one and the most desired one.

Because of the program design, our program can process queries fast. The expected time duration from data inputs to result outputs takes less than 1 second.

In the result page, we choose to display the first 50 child links that we got in case there are more links that distract users' attention. Besides, we add markers to differentiate parent links from child links to make the presentation clearer.

7.2 Weakness

In the links that we crawled, there is a domain name called "LinkedIn". This domain can not be crawled due to some encrypted functions. As a result, there is usually a time run-out problem when the program trying to get details of such pages. After tests and considerations, we decided to handle it by skipping it because it is the most efficient and least cost way.

The stemmed keyword list function is very long. It has some meaningless words such as single alphabet and numbers. This would lead to the confusion of users because they need to roll down a lot to find the desired words.

7.3 Potential improvements

We thought of adding hyperlinks to the child links as well. However, since all the outputs mainly consist of different links, the layout will be all in blue and be underlined. It looks very distractive and is too sharp as a result page.

We also considered adding description to each child link to better describe the properties of them. But the problem we encountered is that it is hard to find the precise description by crawling the page using our function. It will involve much more word processing and even need to understand the words. So, we have not come up with a proper way to include that.

In terms of language, our program can not handle Chinese characters now. It would be interested if this function could be added.

For similarities, we think simply using cosine similarity and anchor weighting may not be enough. The whole searching process will be more accurate if we could use PageRank algorithm to enhance the precision and recall results.

We also tried to implement a function called "show more" because some child lists are extremely long. As we would like the page presentation to be neat and clear for users, we designed a function to fold links larger than 5. However, due to some syntax problems, it is not carried out yet.