COMP4321 Project Report

Group 20
Name: PHAM Thanh Lam **E-mail: tlpham@connect.ust.hk**
Name: NGUYEN Kha Nhat Long **E-mail: knlnguyen@connect.ust.hk**
Url: http://vml1wk218.cse.ust.hk:8080/SearchEngine.html

Overview

The system mainly consists of the spider, index database, retrieval function and web server. The spider is responsible for crawling the web page recursively. The indexer will process the page crawled and store useful information in the database. The frontend will process the query inputted from the user. The backend will read the database and using the algorithm in retrieval function to compute relevant pages based on the similarity between the query and the page. Then, these pages will be rank and output to the user. Details of individual parts will be discussed later.

1. Overall design of the system

The system is divide into many folder each use for a specific task.

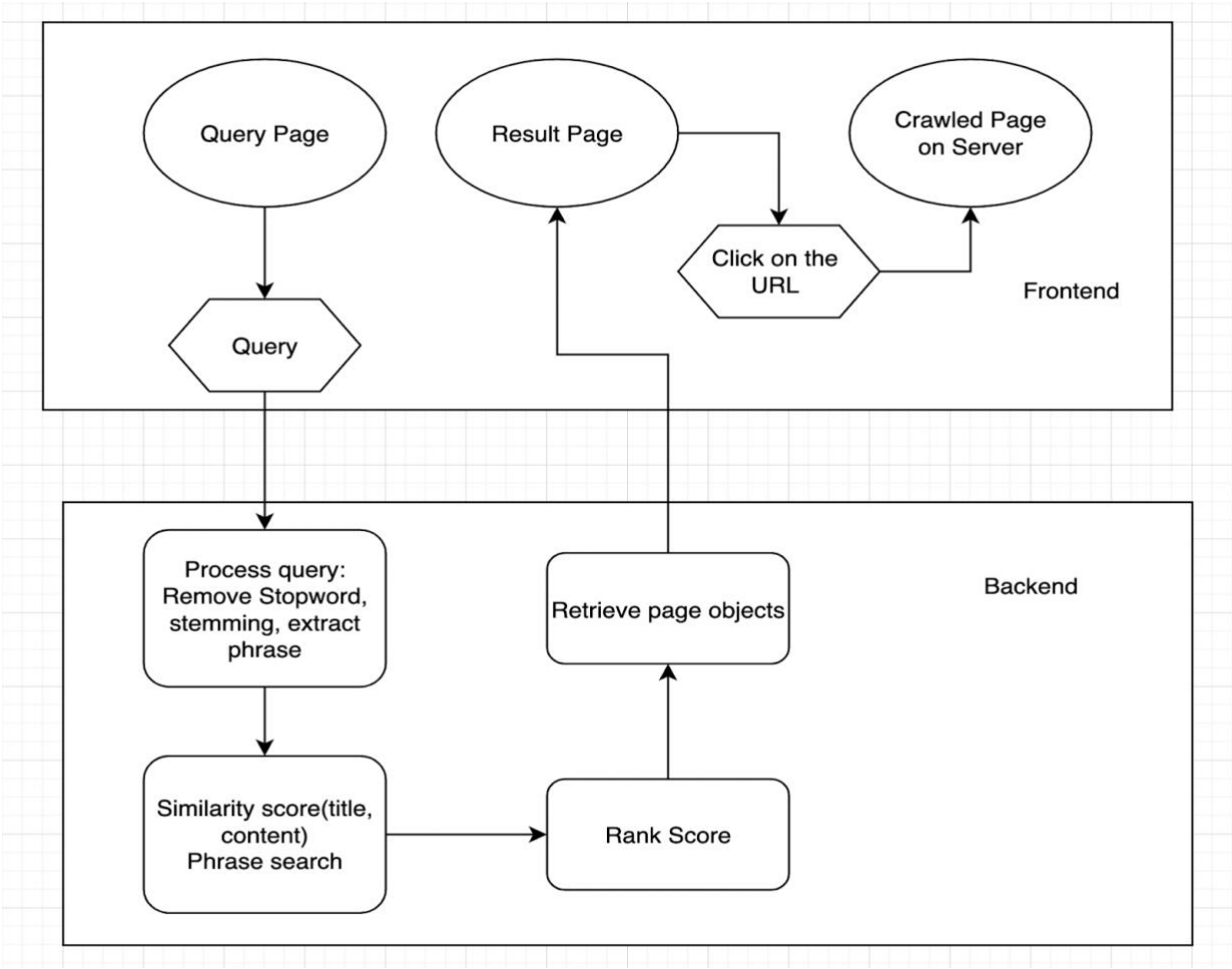
Folder info in the code system

spider	This folder contain the Crawler function and all data type support to build and run the Crawler. This class use to crawl and get data to form the database
database	This folder contain the Rocksdb class to run the spider and get the data. This folder also define interface to convert class to byte and byte to class to store data on Rocksdb This folder also define interface to write and read data to the database The write interface is class WriteData The read interface is class ReadData
forward	This folder define the class ForwardData which is the implement of the forward index. It also contain interface to create the ForwardIndex for each web page.

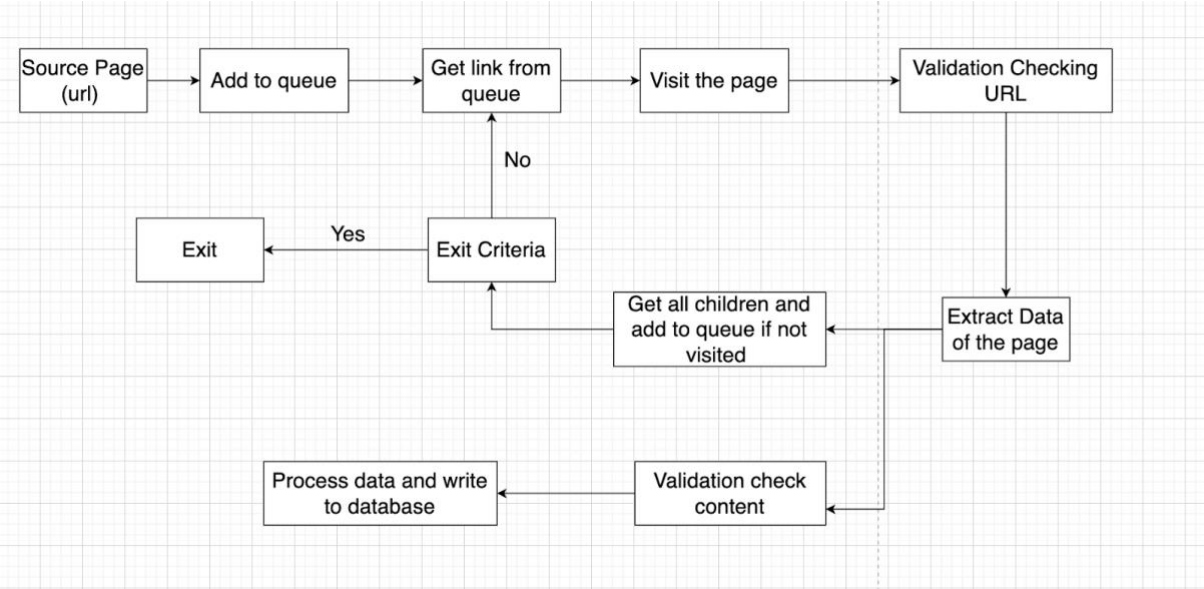
invert	This folder defind the class InvertData which is the implement of the invert index It also contain the interface to create the InvertIndex for each word.
processing	This folder contain all function to remove stop word and stem the word. This define the effective interface to handle it.
retrieval	This folder define the retrieval function of the search engine.
server	This folder is the Jsp server which will handle the query
test	This folder contain all the test case,debug that we use through implement the system.

Overall Design

Webserver Structure:



Spider structure

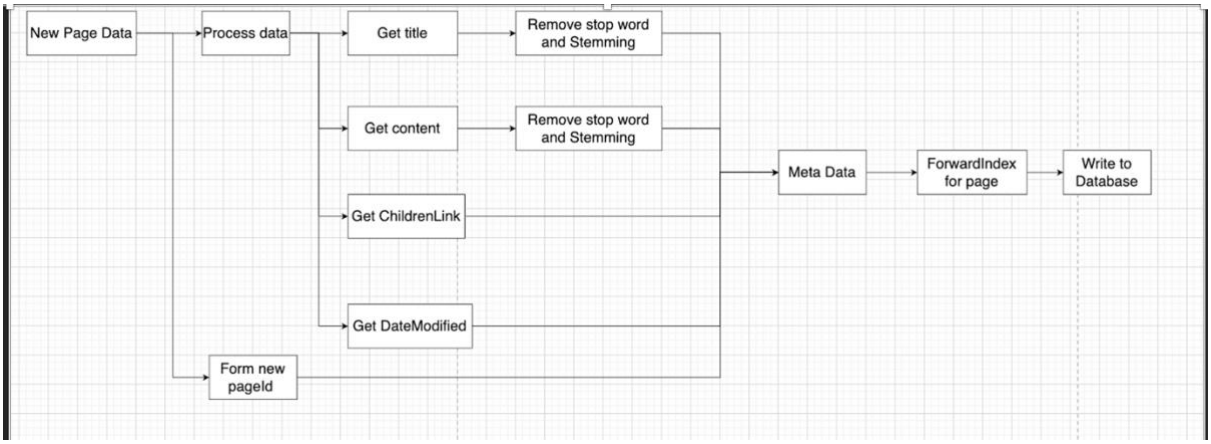


For Validation Checking URL we only check if the URL is valid and not visited before. If the url is valid but we cannot access the website due to some reason such as invalid credential, not authorizable. Exception will occur and handle by the spider and continue with other link.

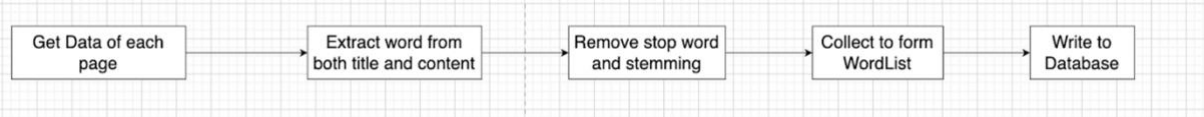
For Validation check content we check to make sure that both the title and content are not null value. And also check if the page is move or not. If the title or content is null, or the page is moved, we won't process data and don't write to database

Indexers structure (Database)

Forward index

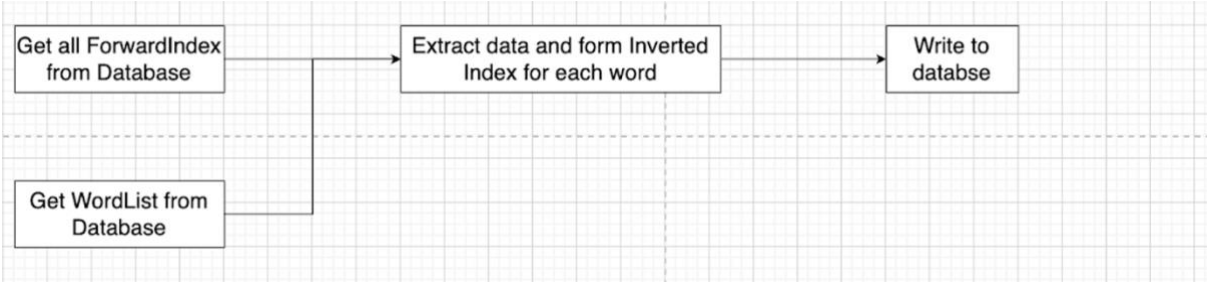


WordList (List of all word of all documents that crawling)

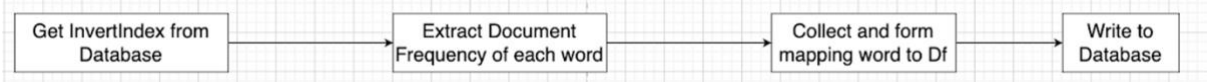


MaxTf (We store MaxTf of content and title of each document separately but the method are the same)

Inverted Index



Document Frequency of each word that in the WordList



2. File structure of the index database

Forward Index

Description:

The WordData store the local data of the word in the title or content of the page

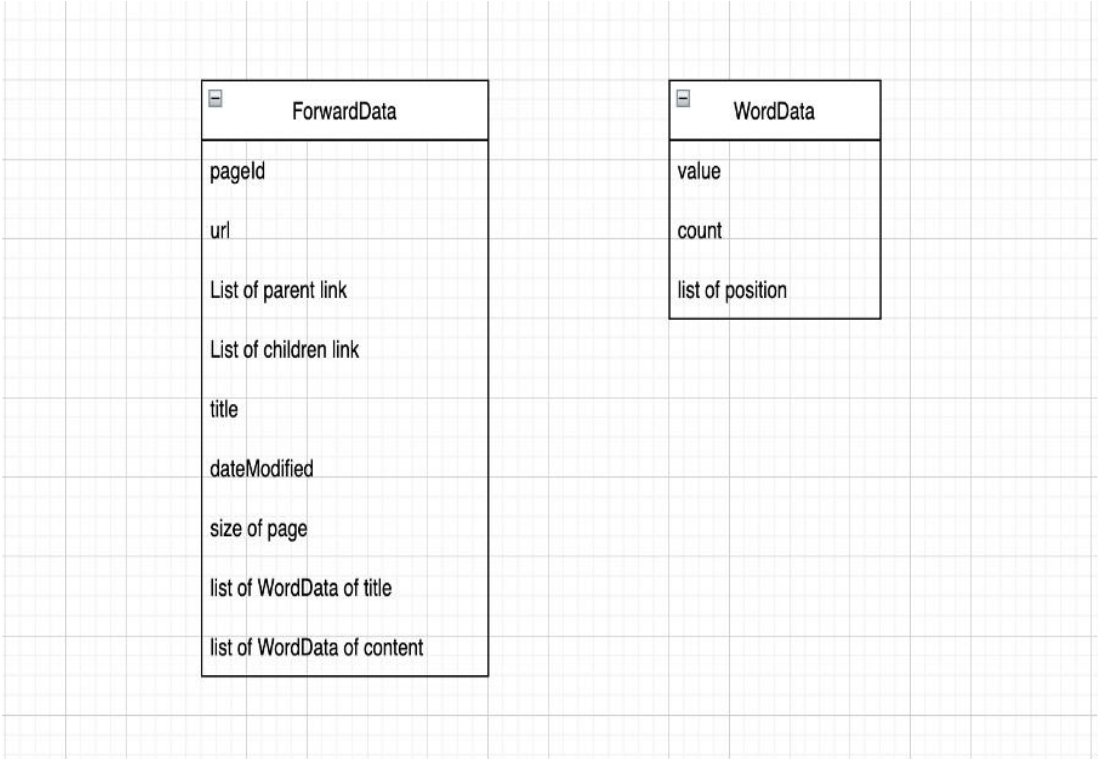
It store the value of the word as well as the number of time as the word appears in the title or content is the variable count
It also store the list of position that the word appears in the title or content

The structure of the ForwardData is simply as below

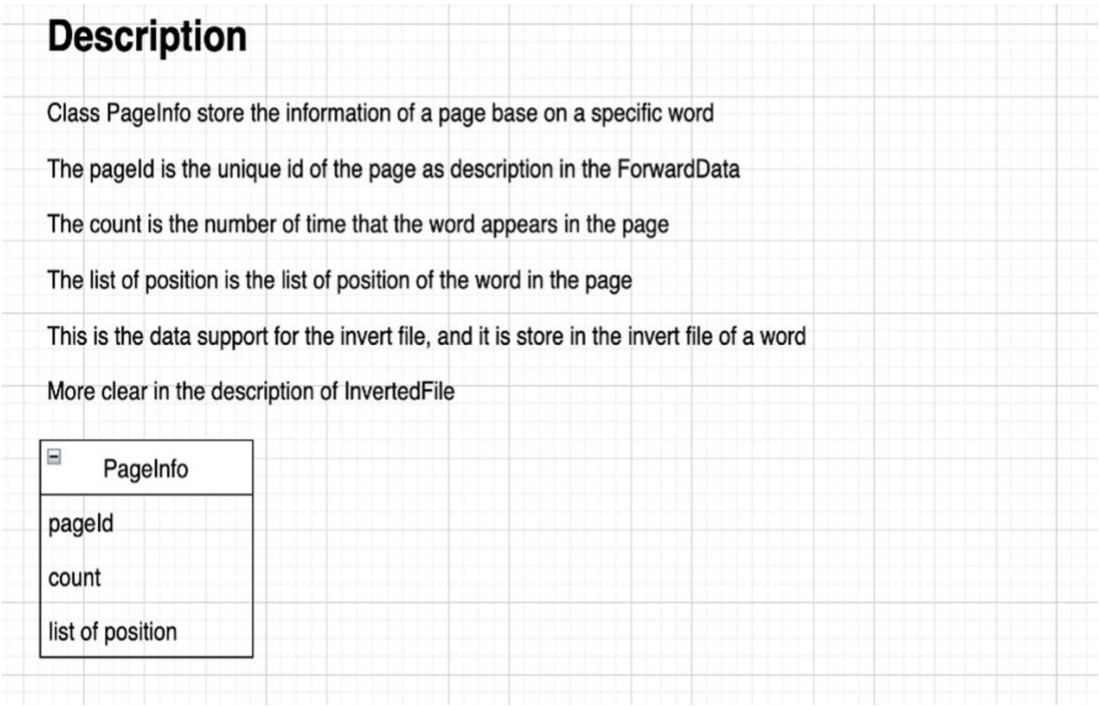
The list of WordData of title is that we process the title of the page to be the list of WordData for each word that in the title

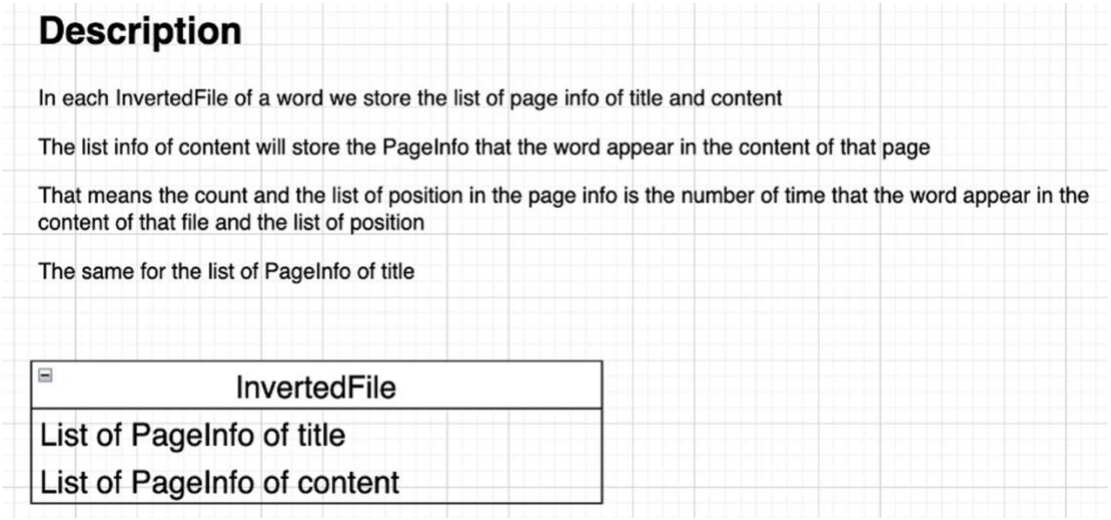
The list of WordData of content is that we process the title of the page to be the list of WordData for each word that in the content

In particular, we separate the content and the title, and the WordData is the information of the word in the title and content respectively.



Inverted Index





We also store some global information to be more effective when doing the retrieval function

We store the wordlist as the stem of all word in all document. We also remove stopword

We store the value of df (document frequency) for each word

We store the value of max_tf (max term frequency) for each document

3. Algorithm

3.1 Processing

Porter's algorithm is used to stem word for processing. Firstly, we remove stopword by using the stopword list after that we apply Poster. It applies the same set of rules (5 steps to remove suffixes) to all words. The stem of a word can be expected so the stem is consistent when there are changes to the corpus.

3.2 Ranking

Cosine similarity is a ranking function which can be used in search engines. We transfer both the document and the query into the vector in the vector space. Each entry in the vector is the weight of the word corresponding to the document or query. For particularly, we use tf-idf with normalize with tf-max to determine the weight of the word in the document and the number of time that the word in the query to be weight of word in the query

TF-IDF is the function to compute weight of a term in the document

The formula is $\frac{tf}{tf\_max}.\log_2\left(\frac{N}{df}\right)$

With N is the number of documents

Ranking algorithm we want to favor the title over the content. For each page, we store separately the content and the title of the page as two separate document. For each query we do cosine similarity of the query with the title and the content separately. After that the similarity between the query and the page is 5* sim(Query,title)+sim(Query,content)

To compute the score of each documents we need to compute the cosine similarity of each page with the query

Firstly, the query is process to the list of word that in the WordList. After that, for each document we compute the vector space base on the list of word of query after processing.

For example: Query is " food in hong kong" then the list of word is ["food","hong", "kong"]

Then for each document we form a vector with 3 entry each is the weight of three word ["food","hong", "kong"] for that page.

We use inverted file of these word to compute the weight.

We need to go through the inverted file, extract data and compute the corresponding weight for each page in the inverted file. The data of max_tf and df can extract from the database

This is why we need inverted file and also store meta data separately on database.

The page is sorted base on score and then output base on ranking.

3.3 Phrase

Filter algorithm is used to search for phrases if the query contains a phrase such as "Hong Kong".

The parameters are wordlist, which is a list of words in a phrase query ("Hong Kong" => ["hong", "kong"]), and invertedData, which is a list of inverted data for each word in the wordlist.

We assume that each word in the wordlist is in the database and not a stopword, if it is, we set the word to be an empty string.

First we try to find the document that all the non-empty words in wordlist are in the document.

We keep track of the current index of invertedData for all words in the wordlist.

Since the documents in invertedData are sorted in ascending order, then we check from the first document to the last document. If the current document has an id greater than the docId in the invertedData of some words, then we can set the index of that word to check the next docId in the inverted Data. If there is a word that has the current docId in the invertedData is greater than current document Id, then the current document will not contain the phrase, we skip this document by setting the current document to the current docId in the invertedData of that word.

At the end of the outer while loop, we increase the value of the document id by 1. Since the document Id always increases, eventually, the loop will terminate by the condition that the document id exceeds some all the docId in the inverted data of a word.

If all the non-empty words are in the current document, we check if all the non-empty words preserve the order as in the original phrase. If they preserve, then we add the document to the result. We ignore the empty string in all the steps, the reason to keep empty string in wordlist is to get the original order of the phrase.

We prepare parameters as the following. First get the phrase inside double quotes by regular expression. Then we remove stopwords and stem each word in the phrase. If a word is a stopword or not in the database, then we change the word to an empty string and add it to the wordlist.

Supporting phrase search: After calling the filter, we get the list of all documents containing the phrase. After calculating the similarity scores for all the documents compared to the query, if the query contains a phrase, for now, our model is simple, we penalize all documents that do not have the phrase by multiplying its score by 0.5. We are planning to recalculate the similarity with a phrase as a term.

3.2 Do query (user search)

The algorithm for the this is first we process the query by remove stopword and stemming

We also extract all phrase input from user

We keep track of the list of SatisfyDocument that satisfy the phrase search

First, for each phrase, we do filter and get the new list SatisfyDocument

Sencond, after all phrase, we compute score and do ranking on these SatisfyDocument

Note that in the score we don’t favour the phrase and in here the query is treated as the query without phrase

4. Installation procedure

1. Put below files in /apache-tomcat-10.0.20/webapps/ROOT/
   -SearchEngine.html
   -SearchEngine.jsp
   -SearchEngine.css

2. Put below folder in /apache-tomcat-10.0.20/webapps/ROOT/WEB-INF/classes
   resource/
   lib/

3. Put lib/ folder in /apache-tomcat-10.0.20/webapps/ROOT/

4. In directory /apache-tomcat-10.0.20/webapps/ROOT/WEB-INF/classes:
   Run the following command to crawl webpages:

```
make all
```

5. In directory /apache-tomcat-10.0.20/webapps/ROOT/WEB-INF/classes:
   Run the following command to start the Search Engine:

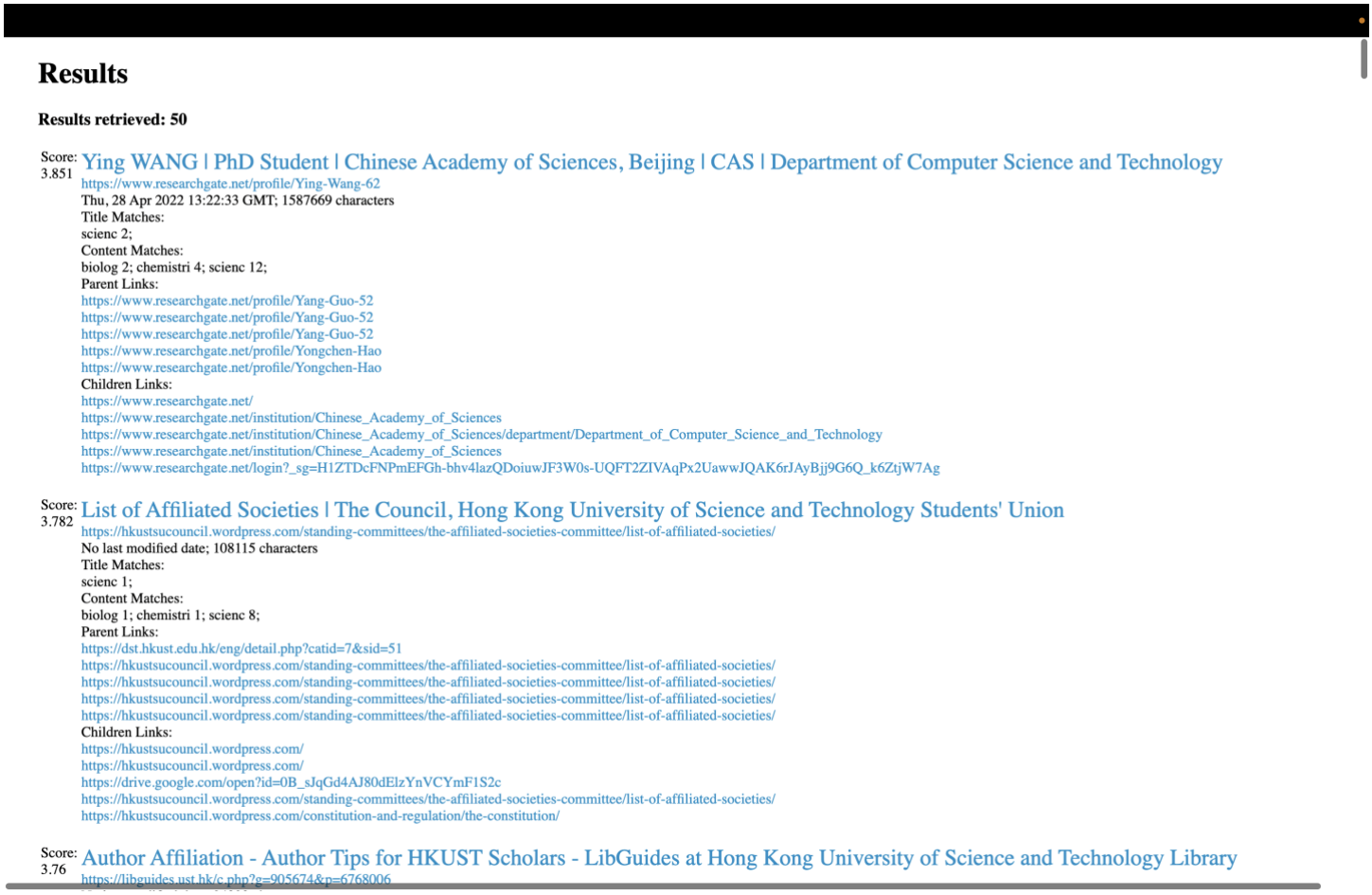
```
make server
```

5. visit VM_location:8080/SearchEngine.html to use the search engine

5. Testing of the functions implemented

Query without phrase search:

Result:



Results

Results retrieved: 50

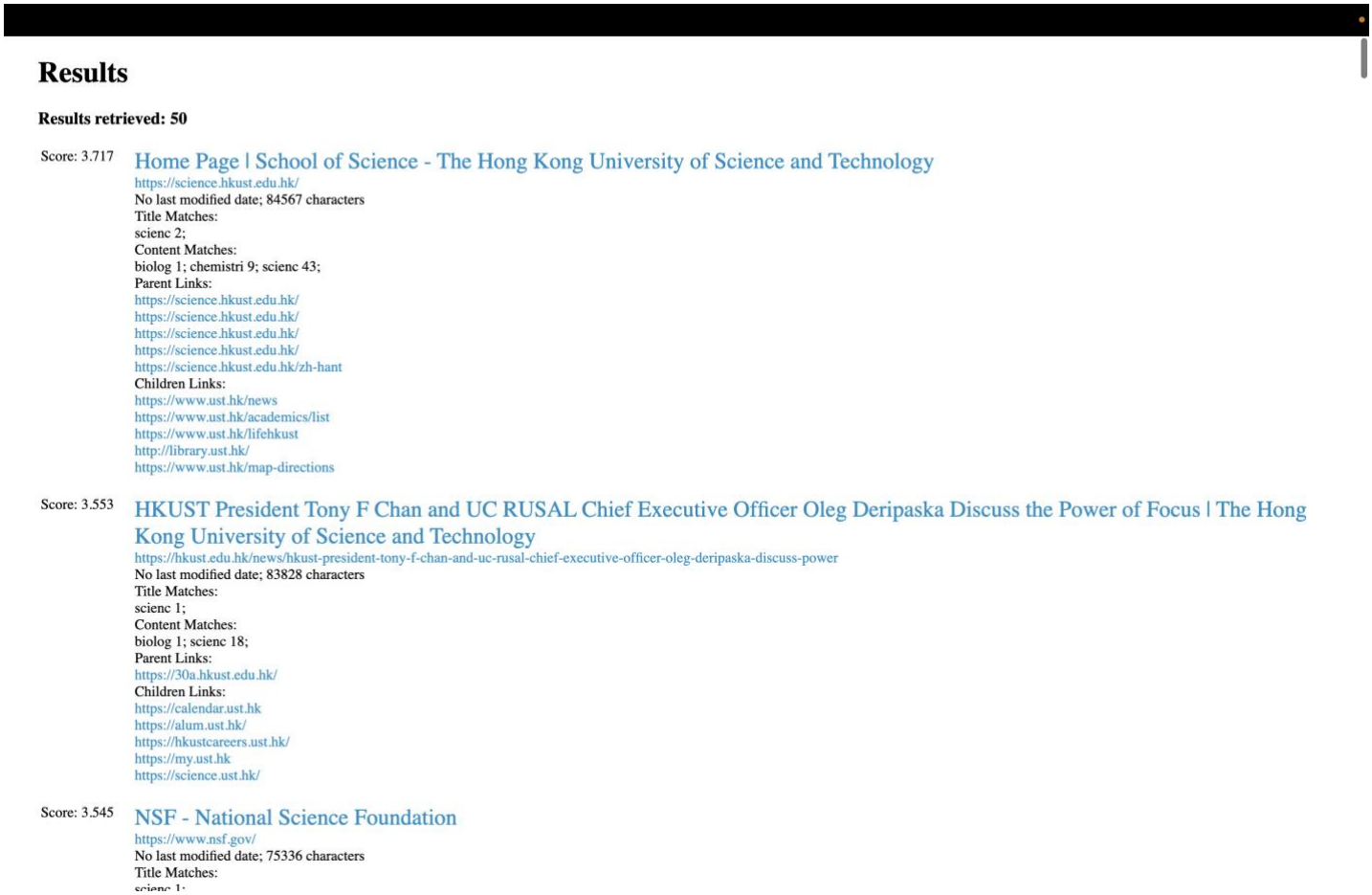
Score: 3.851 Ying WANG | PhD Student | Chinese Academy of Sciences, Beijing | CAS | Department of Computer Science and Technology
https://www.researchgate.net/profile/Ying-Wang-62
Thu, 28 Apr 2022 13:22:33 GMT; 1587669 characters
Title Matches:
scienc 2;
Content Matches:
biolog 2; chemistri 4; scienc 12;
Parent Links:
https://www.researchgate.net/profile/Yang-Guo-52
https://www.researchgate.net/profile/Yang-Guo-52
https://www.researchgate.net/profile/Yang-Guo-52
https://www.researchgate.net/profile/Yongchen-Hao
https://www.researchgate.net/profile/Yongchen-Hao
Children Links:
https://www.researchgate.net/
https://www.researchgate.net/institution/Chinese_Academy_of_Sciences
https://www.researchgate.net/institution/Chinese_Academy_of_Sciences/department/Department_of_Computer_Science_and_Technology
https://www.researchgate.net/institution/Chinese_Academy_of_Sciences
https://www.researchgate.net/login?_sg=H1ZTDcFNPmEFGh-bhv4lazQDoiuwJF3W0s-UQFT2ZIVAqPx2UawwJQAK6rJAyBjj9G6Q_k6ZtjW7Ag

Score: 3.782 List of Affiliated Societies | The Council, Hong Kong University of Science and Technology Students' Union
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
No last modified date; 108115 characters
Title Matches:
scienc 1;
Content Matches:
biolog 1; chemistri 1; scienc 8;
Parent Links:
https://dst.hkust.edu.hk/eng/detail.php?catid=7&sid=51
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
Children Links:
https://hkustsucouncil.wordpress.com/
https://hkustsucouncil.wordpress.com/
https://drive.google.com/open?id=0B_sJqGd4AJ80dElzYnVCYmF1S2c
https://hkustsucouncil.wordpress.com/standing-committees/the-affiliated-societies-committee/list-of-affiliated-societies/
https://hkustsucouncil.wordpress.com/constitution-and-regulation/the-constitution/

Score: 3.76 Author Affiliation - Author Tips for HKUST Scholars - LibGuides at Hong Kong University of Science and Technology Library
https://libguides.ust.hk/c.php?g=905674&p=6768006

Query with phrase:

Result



Results

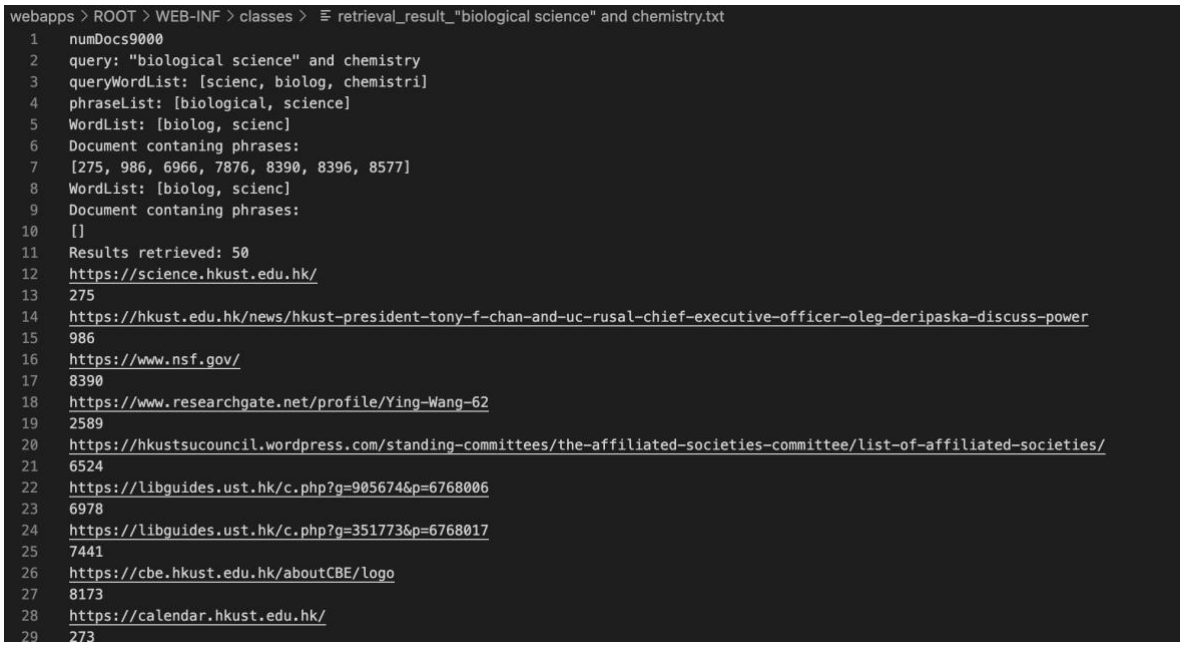
Results retrieved: 50

Score: 3.717 Home Page | School of Science - The Hong Kong University of Science and Technology
https://science.hkust.edu.hk/
No last modified date; 84567 characters
Title Matches:
scienc 2;
Content Matches:
biolog 1; chemistri 9; scienc 43;
Parent Links:
https://science.hkust.edu.hk/
https://science.hkust.edu.hk/
https://science.hkust.edu.hk/
https://science.hkust.edu.hk/
https://science.hkust.edu.hk/zh-hant
Children Links:
https://www.ust.hk/news
https://www.ust.hk/academics/list
https://www.ust.hk/lifehkust
http://library.ust.hk/
https://www.ust.hk/map-directions

Score: 3.553 HKUST President Tony F Chan and UC RUSAL Chief Executive Officer Oleg Deripaska Discuss the Power of Focus | The Hong Kong University of Science and Technology
https://hkust.edu.hk/news/hkust-president-tony-f-chan-and-uc-rusal-chief-executive-officer-oleg-deripaska-discuss-power
No last modified date; 83828 characters
Title Matches:
scienc 1;
Content Matches:
biolog 1; scienc 18;
Parent Links:
https://30a.hkust.edu.hk/
Children Links:
https://calendar.ust.hk
https://alum.ust.hk/
https://hkustcareers.ust.hk/
https://my.ust.hk
https://science.ust.hk/

Score: 3.545 NSF - National Science Foundation
https://www.nsf.gov/
No last modified date; 75336 characters
Title Matches:

Process phrase:



6. Conclusion

Strength:

- Quite fast retrieval because the tf-idf and document norm are computed using inverted file, and meta data needed is stored separately as global data in the database
- High relevance if the query matches the title of a page because of favor title 5 times as content.

Well structure system, easy to extend and add additional feature and improve.

Weakness:

- Limited of word and phrase because each query requires computing the tf-idf of all documents
- Title matching dominant the similarity score. Page content is less important
- No return result if only stop word are search
- Ignore link base between the documents

Things to improve:

- Using multithread to speed up the query process, because the inverted data are independent and can get separately
- Include link in the structure, compute score for each page and use this to favor important page.

7. Contribution

NGUYEN Kha Nhat Long: implemented database, forward, invert, processing and retrieval function, improved crawler. ~ 50%

PHAM Thanh Lam: implemented crawler, interface, server, phrase search algorithm, query pre-processing (extract phrase from double quotes and remove word not in database) ~ 50%