Overall Design of the System

In this project, a **web-based search engine** is built.

On the web interface, user can submit request of fetching pages given a starting url and number of pages to fetch. User can also check the existing database, and clear the database. On the other hand, user can search by submitting a search query, even with quotation marks on phrases to boost ranking of pages containing the phrases. Pages will be ranked by descending scores, and at most 50 pages will be displayed. Page Title, url, last modified date, size of page, top 5 keywords and frequency, and parent and child links will be displayed.

Class Spider, Indexer, Database, StopStem, Porter, SearchEngine are used to develop the search engine with several jsp and html files for supporting the web interface.

Class Spider will fetch pages recursively given a starting url and number of pages to fetch. When fetching a page, Spider will call class Indexer to extract words, titles, bigrams, and trigrams, and insert them to the database through class Database. Dealing with the words, stop words will be removed and then stemmed by class StopStem and Porter. After crawling the pages, the tfxidf/max(tf) matrix will be calculated for each Page and Word, ready for the search process. Class SearchEngine given a query will then calculate scores for each page with favors in matches in title, and matches of phrases, and get the top 50 scores pages.

JDBM Database Scheme of the Indexer

1. PageID – URL Mapping Tables:

PageIDtoURL(PageID, URL)

A mapping table for the interchange of PageID and URL. The key is PageID. The PageID is an index provided by the Spider, starting from 0.

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
URL	String	The URL of a web page

URLtoPageID(URL, PageID)

A mapping table for the interchange of PageID and URL. The key is URL.

Key	Data Type	Description
URL	String	The URL of a web page
Value	Data Type	Description
PageID	String	The PageID of a web page

For simplicity of storing the URL in other databases, an ID is given to the page and recorded on this mapping table. Both directions are recorded, for efficiency of retrieval.

2. Table for title of each web page:

PageIDtoTitle(PageID, Title)

A table for recording the title of a Page when fetching a page. The key is PageID. If the page is not fetched yet, there will not be an entry for it.

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
Title	String	The title of a web page

The Table is for recording the title of the fetched page.

3. Table for LastModifiedTime of each web page:

PageIDtoTime(PageID, LastModifiedTime)

A table for recording last modified time of a Page when fetching a page. The key is PageID. If the page is not fetched yet, the initial value of last modified time would be (Thu Jan 01 08:00:00 HKT 1970).

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
LastModifiedTime	String	The last modified time of a web page Format: EEE MMM dd HH:mm:ss zzz yyyy Example: Thu Jun 16 16:47:39 HKT 2022

The LastModifiedTime is for recording the last modified time of the pages fetched. When the Spider has fetched that page before, and the page is not modified after that, the last modified time from the page and from the table would be the same. Then the Spider will skip this page this time, as it is not modified. This can help handle the cyclic links.

4. Table for Length of each web page:

PageIDtoLength(PageID, Length)

A table for recording the length of a Page when fetching a page. The key is PageID. If the page is not fetched yet, there will not be an entry for it..

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
Length	String	The content-length, html length and number of words extracted from a web page Format: ContentLength;HTMLLength;NumberofWords

The length is for recording the length of the pages fetched. All three Content-Length, HTML Length, and Number of Words will be recorded as some may be missing fetching a page.

5. WordID – Word Mapping Tables:

WordIDtoWord(WordID, Word)

A mapping table for the interchange of WordID and Word. The key is WordID. The WordID is an index provided by the Spider, staring from 0.

Key	Data Type	Description
WordID	String	The WordID of a word
Value	Data Type	Description
Word	String	The word

WordtoWordID(Word, WordID)

A mapping table for the interchange of WordID and Word. The key is Word.

Key	Data Type	Description
Word	String	The word
Value	Data Type	Description
WordID	String	The WordID of a word

For simplicity of storing the word in other databases, an ID is given to the word and recorded on this mapping table. Both directions are recorded, for efficiency of retrieval.

6. Forward and Inverted Indexes for storing Parent – Child relationship:

ParenttoChild(ParentPageID, ChildPageID)

A forward index for storing the Parent-Child relationship. The key is Parent’s PageID.

Key	Data Type	Description
Parent’s PageID	String	The PageID of the parent page
Value	Data Type	Description
Child’s PageID	String	The PageID of the child page Format: ChildPageID1;ChildPageID2;ChildPageID3;…

ChildtoParent(ChildPageID, ParentPageID)

A backward index for storing the Parent-Child relationship. The key is Child’s PageID.

Key	Data Type	Description
Child’s PageID	String	The PageID of the child page

Value	Data Type	Description
Parent's PageID	String	The PageID of the parent page Format: ParentPageID1;ParentPageID2;ParentPageID3;…

These tables store the Parent – Child relationships. Both directions of retrieval are recorded, for efficiency.

7. Forward and Inverted Indexes for page and title words on page:

PageIDtoTitleWordID(PageID, WordID)

A forward index for storing words crawled from the title of a page. The key is PageID. The frequency of each word is also recorded.

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
WordID	String	The WordID of a word in the title on the page Format: word1 freq1;word2 freq2;word3 freq3;…

TitleWordIDtoPageID(WordID, PageID)

An inverted index for storing pages that contains the word. The key is WordID. The frequency of each word on that page is also recorded.

Key	Data Type	Description
WordID	String	The WordID of a word
Value	Data Type	Description
PageID	String	The PageID of a web page that contain the word in title Format: page1 freq1;page2 freq2;page3 freq3;…

These tables store the title words crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

8. Forward and Inverted Indexes for page and word on page:

PageIDtoWordID(PageID, WordID), WordIDtoPageID(WordID, PageID)

(Similar to 7.)

These tables store the words crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

9. Forward Index for page and top five word on page:

PageIDtoTopFiveWordID(PageID, WordID)

(Similar to 7. But only forward index)

The table store the top five words crawled from the web pages.

10. Forward and Inverted Indexes for page and title word bigrams on page:

PageIDtoBiTitleWordID(PageID, WordID), BiTitleWordIDtoPageID(WordID, PageID)

(Similar to 7. But each with 2 wordID)

These tables store the title word bigrams crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

11. Forward and Inverted Indexes for page and title word trigrams on page:

PageIDtoTriTitleWordID(PageID, WordID),TriTitleWordIDtoPageID(WordID, PageID)

(Similar to 7. But each with 2 wordID)

These tables store the title word trigrams crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

12. Forward and Inverted Indexes for page and word bigrams on page:

PageIDtoBiWordID(PageID, WordID), BiWordIDtoPageID(WordID, PageID)

(Similar to 7. But each with 3 wordID)

These tables store the word bigrams crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

13. Forward and Inverted Indexes for page and word trigrams on page:

PageIDtoTriWordID(PageID, WordID), TriWordIDtoPageID(WordID, PageID)

(Similar to 7. But each with 3 wordID)

These tables store the word Trigrams crawled from the web pages. Both directions of retrieval are recorded, for efficiency.

14. Table for TFxIDF:

PageIDtoTFxIDF(PageID, TFxIDF)

A table for storing the TFxIDF of the pages and words. The key is PageID.

Key	Data Type	Description
PageID	String	The PageID of a web page
Value	Data Type	Description
TFxIDF	String	The TFxIDF of that page and each word Format: tfxidf1;tfxidf2;tfxidf3;…

The table stores the TFxIDF weights of web pages and words.

15. Table for SearchHistory:

SearchHistory(Index, SearchHistory)

A table for storing search history. The key is index.

Key	Data Type	Description
Index	String	Index starting from 0
Value	Data Type	Description
SearchHistory	String	The query

The table stores the past search history inputted by the user.

Algorithms

1. Crawl

If a page is not fetched before, it will not be in the database URLtoPageID, then it will be fetched. If a page is fetched before, it will be in the database URLtoPageID and last modified time will be extracted from the database PageIDtoTime. If the new last modified time is earlier than or same as the one in database, then it will be ignored, and next page in pages_queue will be tried to be fetched. If new last modified time is later, then it will be fetched.

Fetching a page, it will be added to visited_pages (for this run) to avoid circular loop. Titles, words, bigrams, trigrams will be extracted and stored after stop word removal and stemming. At the same time, the Indexer will also keep track of the top five frequent words. Links will be stored and added to pages_queue, if not visited.

After fetching the required number of pages, TFxIDF/max(TF) matrix will be calculated and stored to the database.

2. Search

After extracting phrases from the query, and performing stop word removal and stemming, scores of each page will be calculated using the formula below.

For each matching of query word and title word,

$$AddScore = 1 * \log_2\left(\frac{N}{df_i}\right)$$

N is the number of documents
df_i is the number of documents with title containing word i

For each matching of query word and word in page,

$$AddScore = \frac{tf_{ij}}{tfmax_i} * \log_2\left(\frac{N}{df_j}\right)$$

tf_{ij} is the frequency of term j in document i
$tfmax_i$ is the max term frequency in document i
df_j is the number of documents with title containing word j

As tf/max(tf) is upper bounded by 1, the score added from matching in title would usually be larger (if similar df). Also the scores from matching in title is considered as a bonus besides of the score from tfxidf/max(tf), which gives favors in matching in title.

For each matching of query phrase and the title word/bigram/trigram,

$$AddScore = 2 * 1 * \log_2\left(\frac{N}{df_i}\right)$$

N is the number of documents
df_i is the number of documents with title containing phrase i

For each matching of query phrase and word/bigram/trigram in page,

$$AddScore = 2 * \frac{tf_{ij}}{tfmax_i} * \log_2\left(\frac{N}{df_j}\right)$$

tf_{ij} is the frequency of phrase j in document i
$tfmax_i$ is the max phrase frequency in document i
df_j is the number of documents with title containing phrase j

Similar to the score from matching in title, the scores from matching phrases in title/page is considered as a bonus besides of the score from tfxidf/max(tf), which gives favors in matching in phrases. The coefficient 2 is to weight more on phrase match than normal title/page match.

Finally, after calculating the scores for each page, pages will be sorted in descending order of scores. Top 50 Pages with non-zero scores will be normalized to 0 to 100 and printed on the web interface.

Installation Procedure

Download the files on GitHub and unzip it

Add environment var:

CATALINA_HOME = "your tomcat path (e.g. /Users/ABC/apache-tomcat-9.0.74)"
JAVA_HOME = "your jdk path (e.g. /Users/ABC/jdk-20.0.1.jdk/Contents/Home)"

Go to $CATALINA_HOME/webapps and create a folder "Search"
Put the files under $CATALINA_HOME/webapps/Search

Then build the program by:

```
> javac -cp combined.jar WEB-INF/classes/project/*.java
```

Then start it by:

```
> $CATALINA_HOME/bin/startup.sh
```

Open in your browser:
http://localhost:8080/Search/Search.html
to test the program

Shut it down by:

```
> $CATALINA_HOME/bin/shutdown.sh
```

Highlight of Features

1. **Get Similar Pages**: equivalent to search pages with the top 5 most frequent keywords from that page.

2. **List of Stemmed Keywords (Dictionary)**: you can browse through and select keywords to search.

3. **Query Search History**: you can browse through and select queries to search.

4. **Delete Database and Delete History Button**: you can delete database and past search history by one click.

5. **Sensitive Words**: if you add words in sensitivewords.txt, and if your query submitted contains any of the sensitive word, you will get nothing, but a reminder of words that violates the regulation.

Conclusion

The strengths of my system are the number of extra features, the algorithm of flavoring title phrase matching, a reasonable time (2-3mins) to fetch 300+ pages, and everything can be done using the web interface.

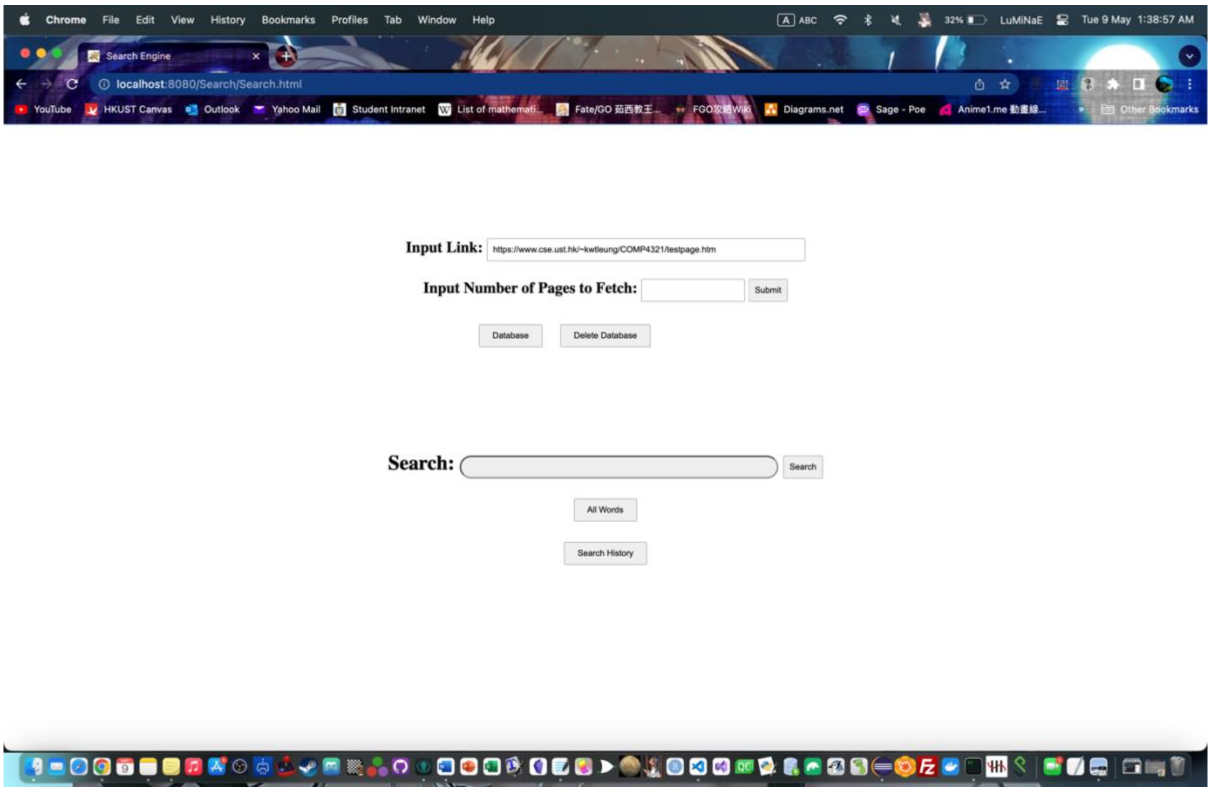
The weaknesses of my system are improvable outlook, code can be simplified.

If I could re-implement the whole system, I would have put more effort on visualization (outlook of the web) and factoring the code.

Interesting features I would like to add to my feature are pagerank, but it would require much more effort.

Testing of Functions (Appendix)

Web Interface

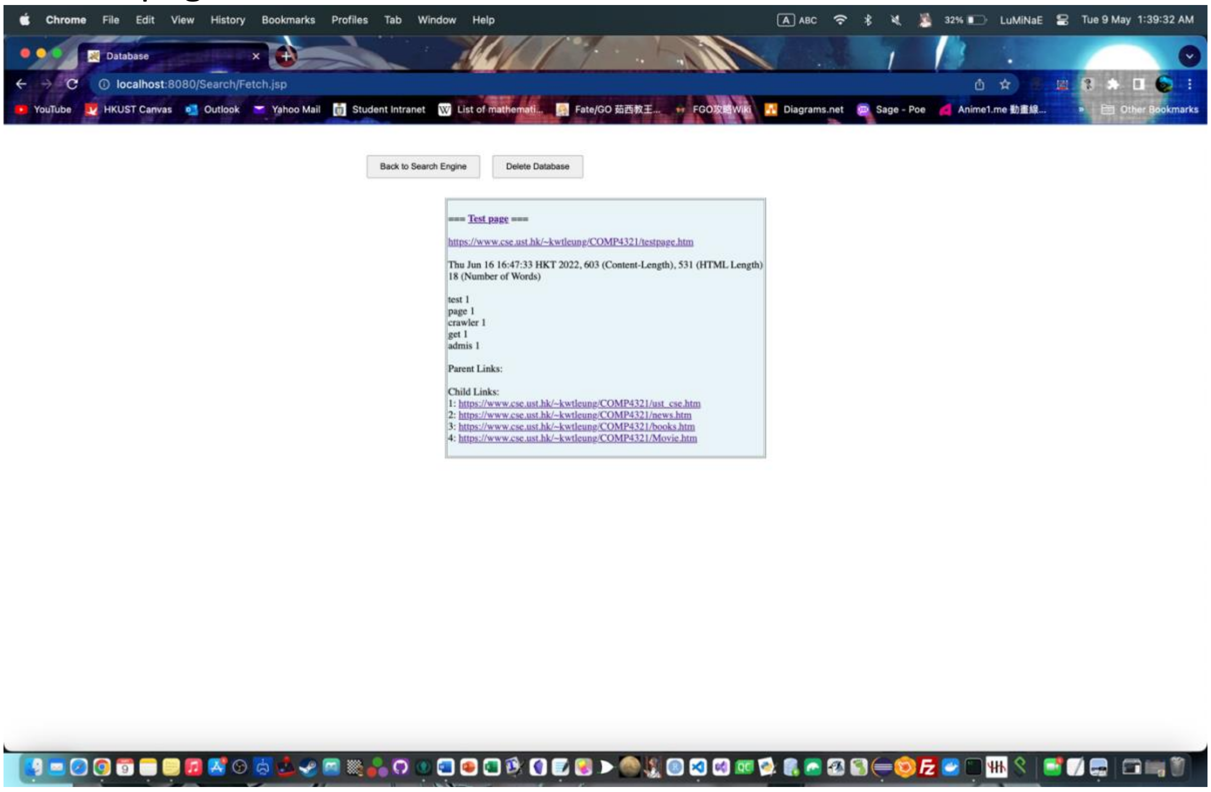


Input Link: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Input Number of Pages to Fetch:

Search:

Fetch 1 page



=== Test page ===

https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

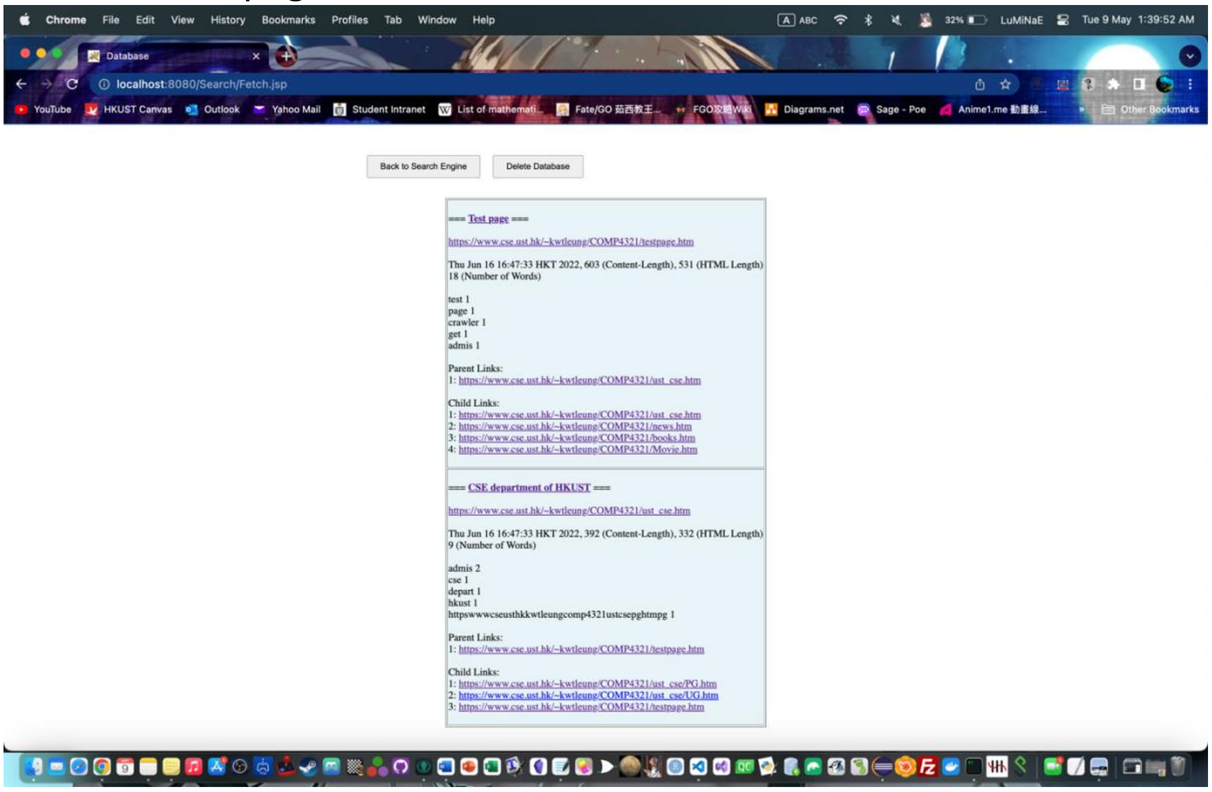
Thu Jun 16 16:47:33 HKT 2022, 603 (Content-Length), 531 (HTML Length)
18 (Number of Words)

test 1
page 1
crawler 1
get 1
admis 1

Parent Links:

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Fetch 1 more page



=== Test page ===

https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Thu Jun 16 16:47:33 HKT 2022, 603 (Content-Length), 531 (HTML Length)
18 (Number of Words)

test 1
page 1
crawler 1
get 1
admis 1

Parent Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

=== CSE department of HKUST ===

https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

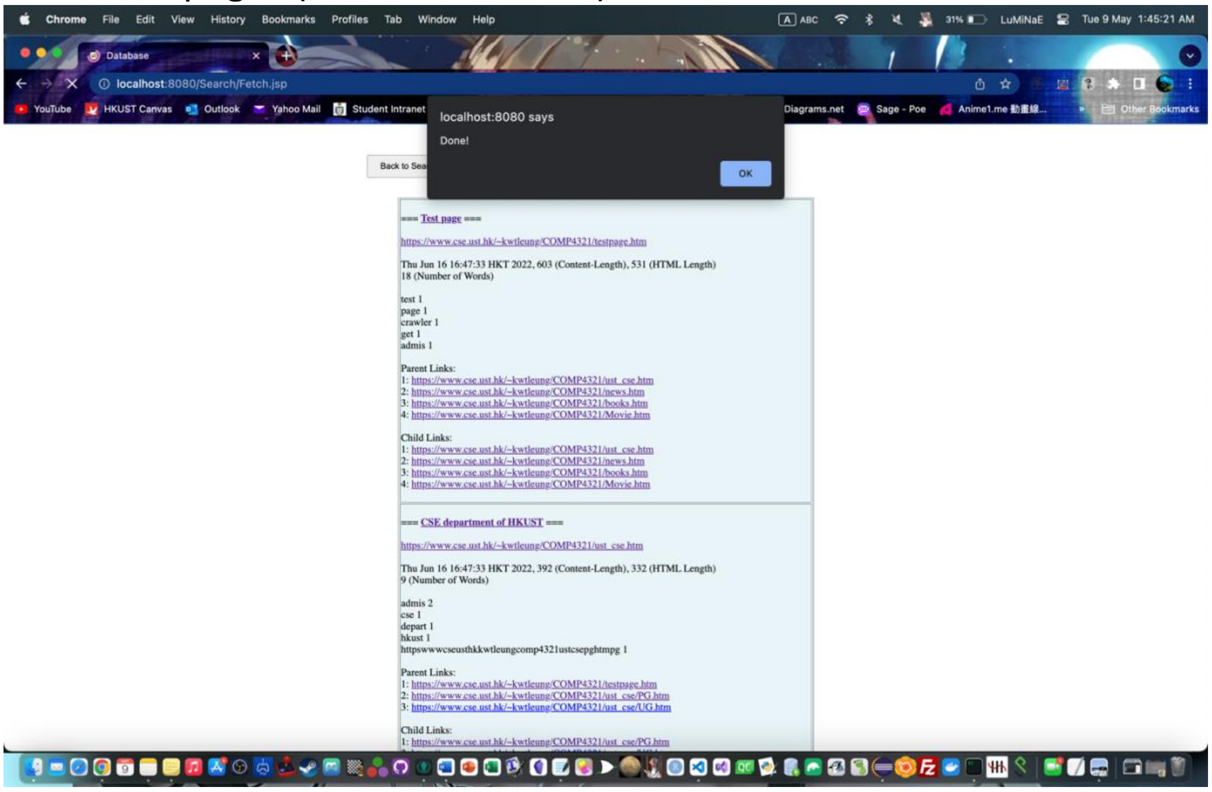
Thu Jun 16 16:47:33 HKT 2022, 392 (Content-Length), 332 (HTML Length)
9 (Number of Words)

admis 2
cse 1
depart 1
hkust 1
httpswwwcseusthkkwtleungcomp4321ustcsepghtmpg 1

Parent Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Fetch 500 pages (Around 2-3 mins)



localhost:8080 says

Done!

=== Test page ===

https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Thu Jun 16 16:47:33 HKT 2022, 603 (Content-Length), 531 (HTML Length)
18 (Number of Words)

test 1
page 1
crawler 1
get 1
admis 1

Parent Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

=== CSE department of HKUST ===

https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm

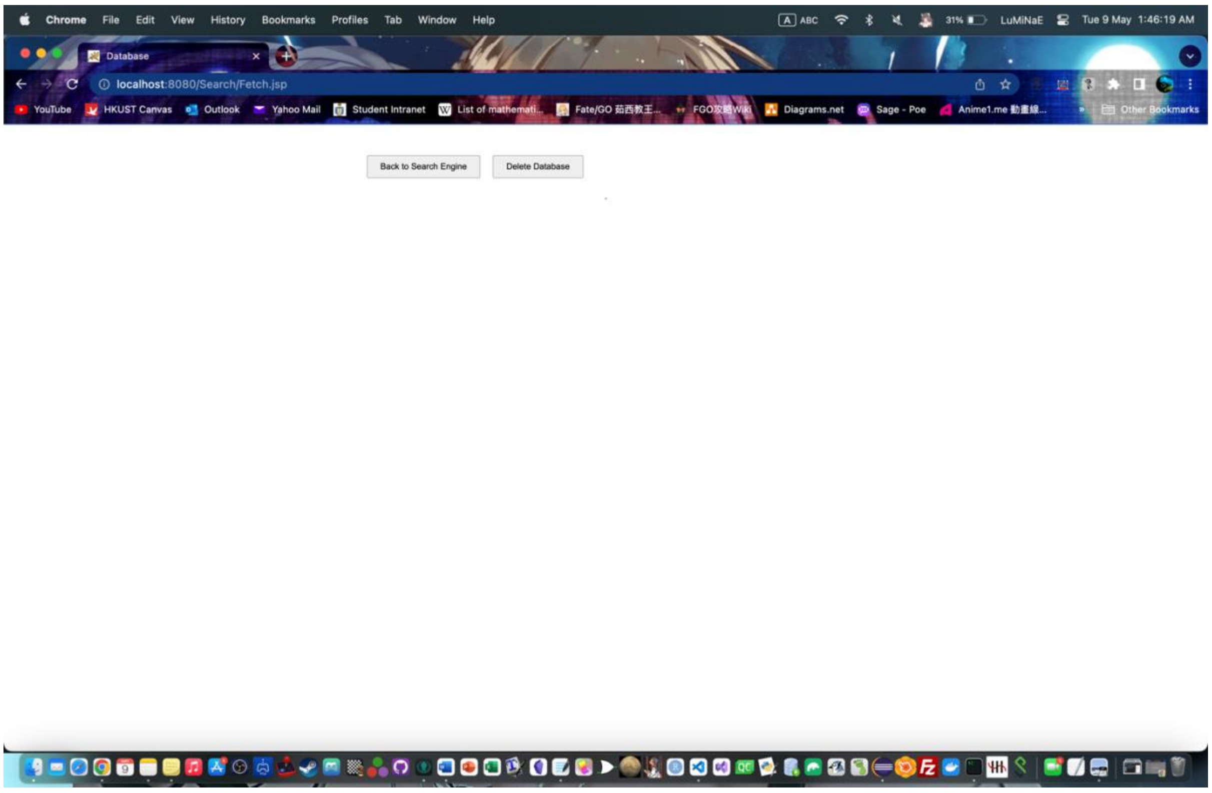
Thu Jun 16 16:47:33 HKT 2022, 392 (Content-Length), 332 (HTML Length)
9 (Number of Words)

admis 2
cse 1
depart 1
hkust 1
httpswwwcseusthkkwtleungcomp4321ustcsepghtmpg 1

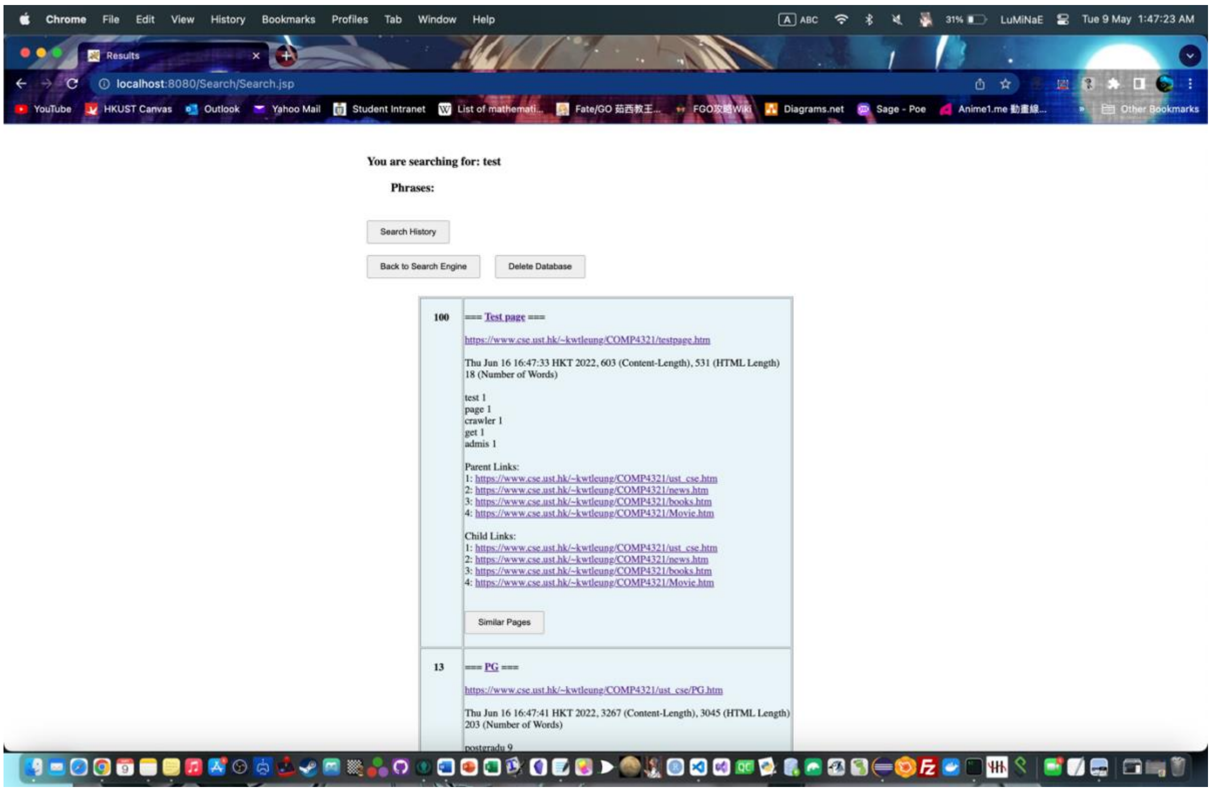
Parent Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/UG.htm

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm

Delete Database



Search Test



You are searching for: test

Phrases:

100

=== Test page ===

https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm

Thu Jun 16 16:47:33 HKT 2022, 603 (Content-Length), 531 (HTML Length)
18 (Number of Words)

test 1
page 1
crawler 1
get 1
admis 1

Parent Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

Child Links:
1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm
2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm
3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm
4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

13

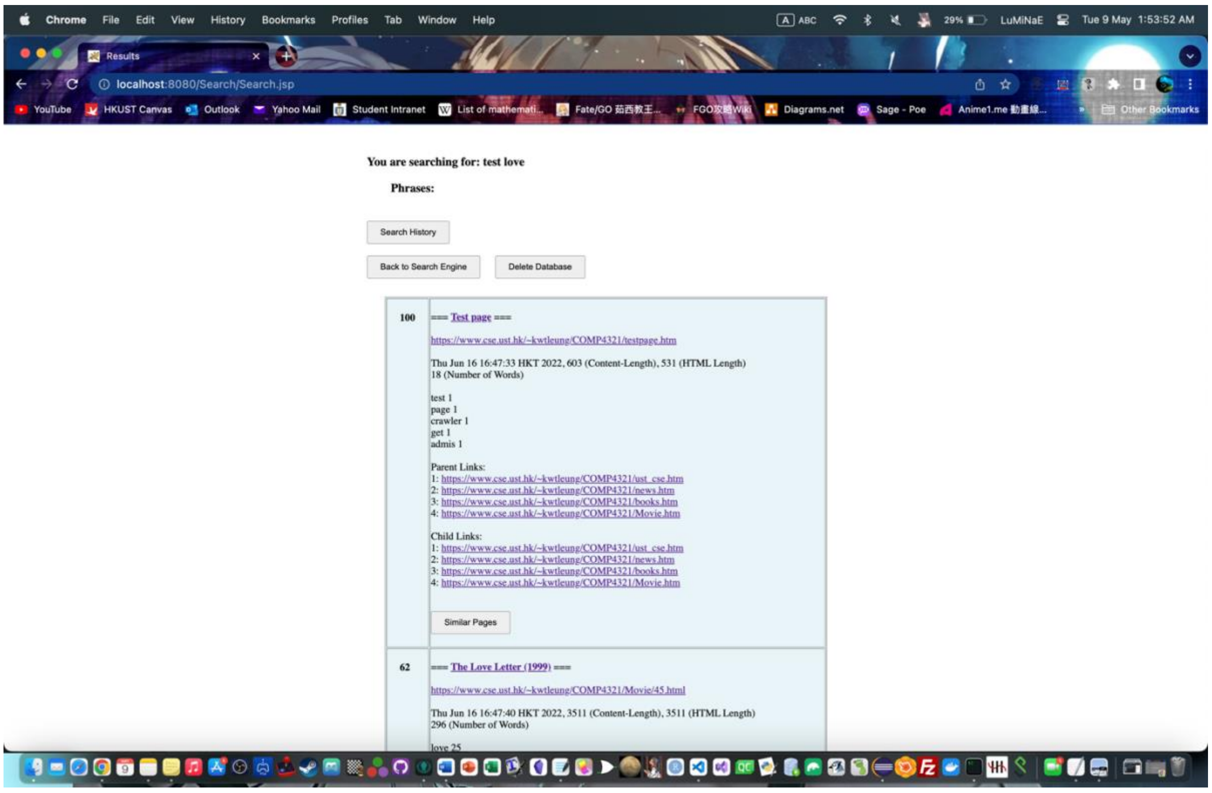
=== PG ===

https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse/PG.htm

Thu Jun 16 16:47:41 HKT 2022, 3267 (Content-Length), 3045 (HTML Length)
203 (Number of Words)

postgradu 9

Search Test Love



localhost:8080/Search/Search.jsp

You are searching for: test love

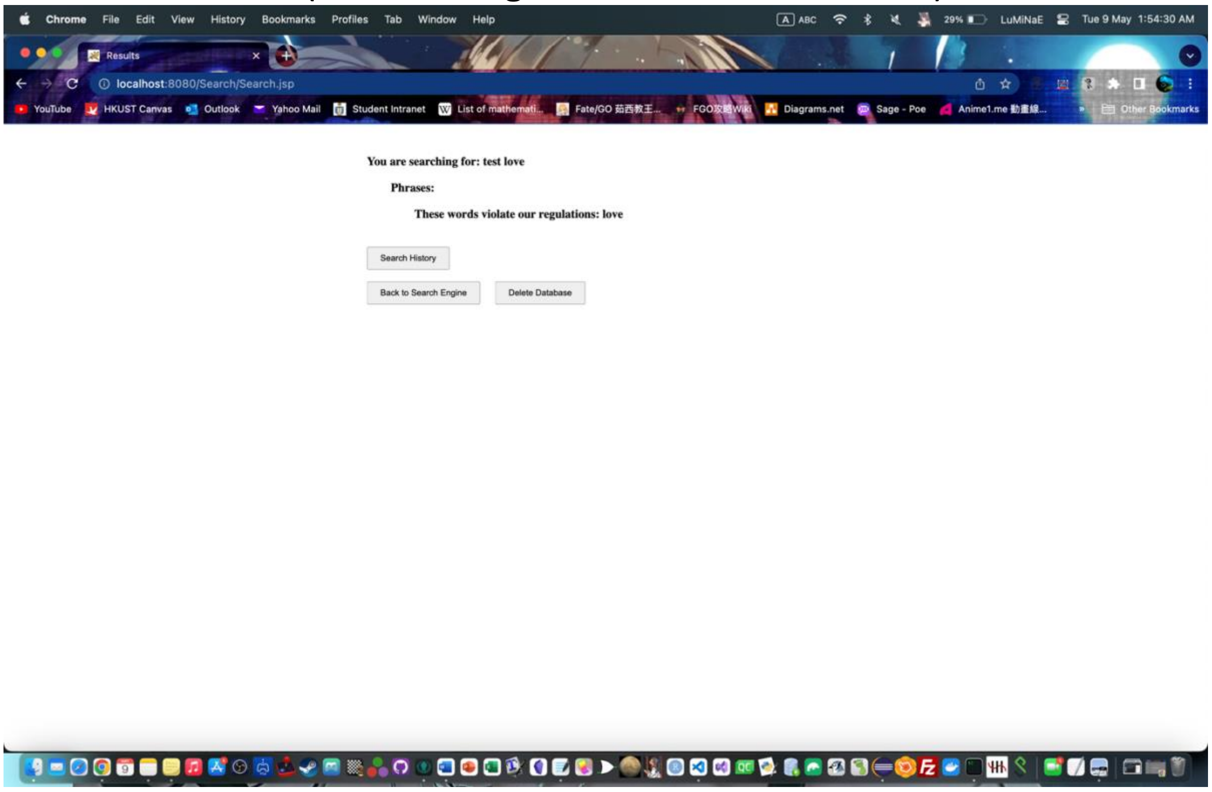
Phrases:

Search History

Back to Search Engine   Delete Database

100	=== Test page === https://www.cse.ust.hk/~kwtleung/COMP4321/testpage.htm Thu Jun 16 16:47:33 HKT 2022, 603 (Content-Length), 531 (HTML Length) 18 (Number of Words) test 1 page 1 crawler 1 get 1 admis 1 Parent Links: 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm 2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm 3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm 4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm Child Links: 1: https://www.cse.ust.hk/~kwtleung/COMP4321/ust_cse.htm 2: https://www.cse.ust.hk/~kwtleung/COMP4321/news.htm 3: https://www.cse.ust.hk/~kwtleung/COMP4321/books.htm 4: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm Similar Pages
62	=== The Love Letter (1999) === https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/45.html Thu Jun 16 16:47:40 HKT 2022, 3511 (Content-Length), 3511 (HTML Length) 296 (Number of Words) love 25

Search Test Love (after having Love as Sensitive Word)



localhost:8080/Search/Search.jsp

You are searching for: test love

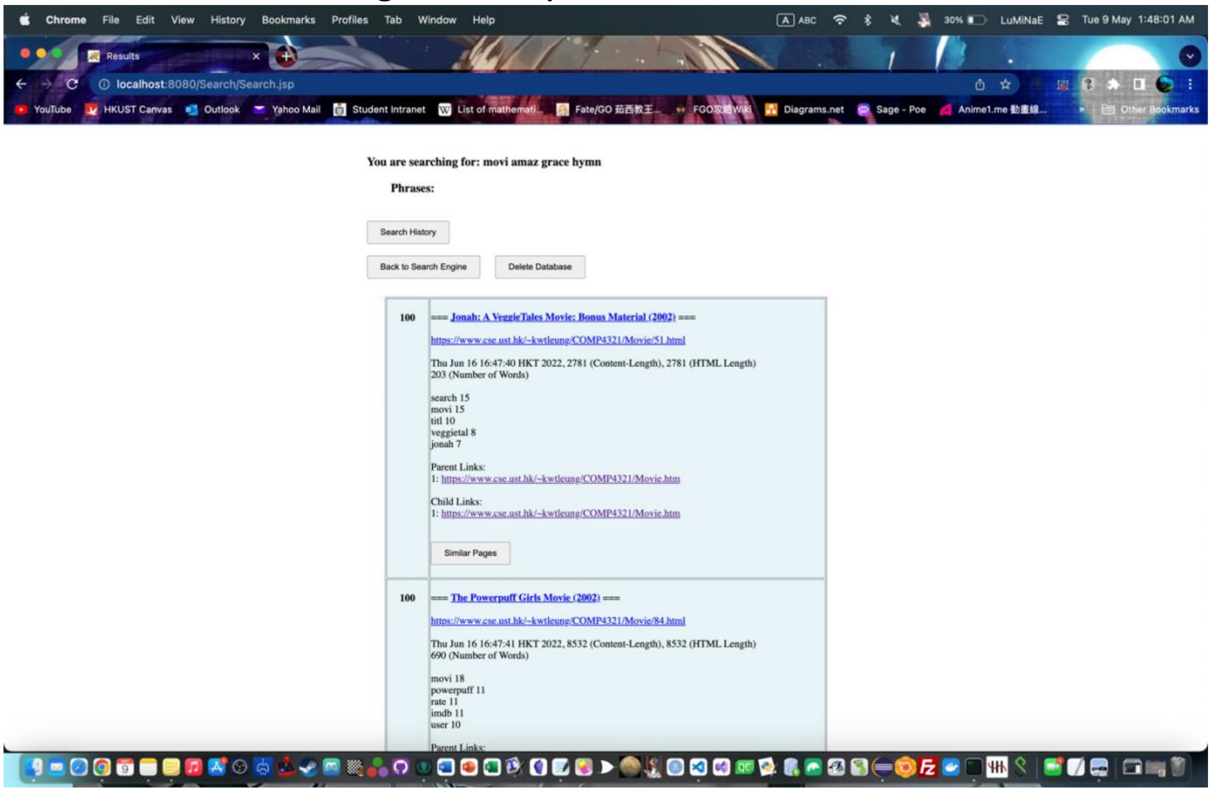
Phrases:

These words violate our regulations: love

Search History

Back to Search Engine   Delete Database

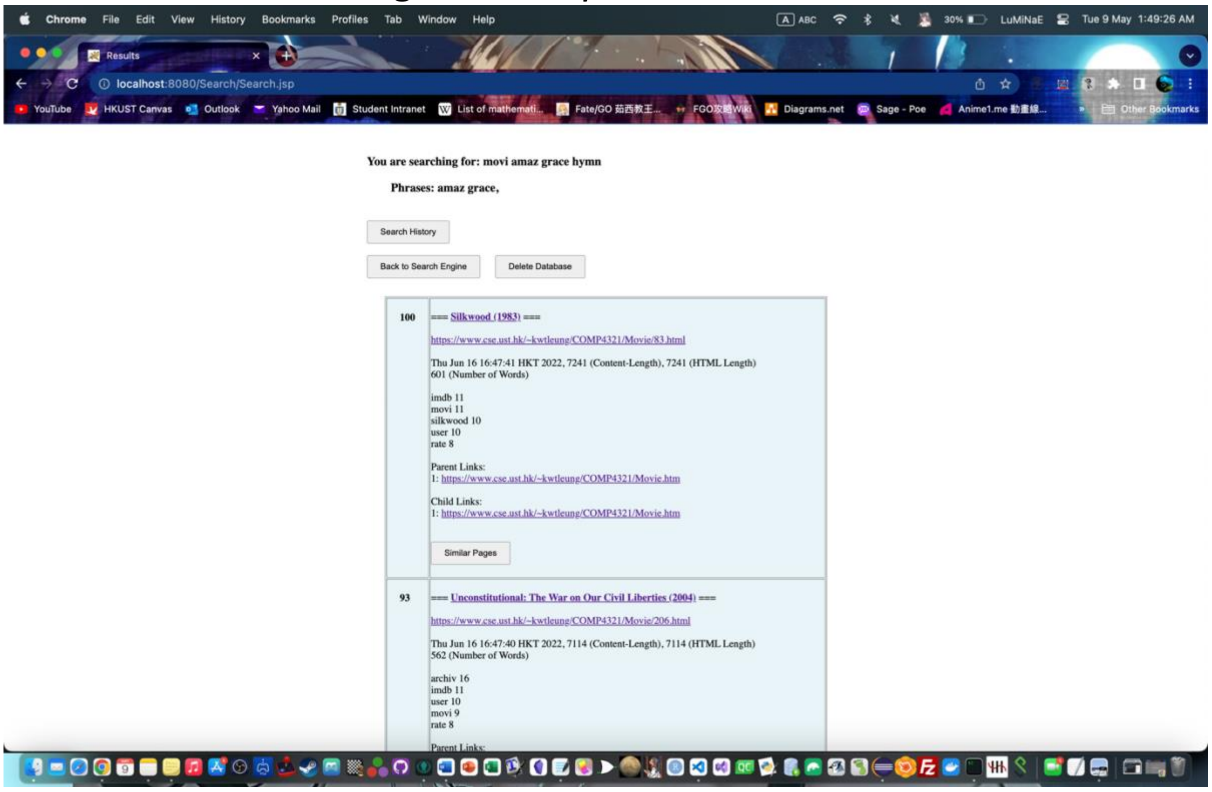
Search movie Amazing Grace Hymn


You are searching for: movi amaz grace hymn

Phrases:

```
100   === Jonah: A VeggieTales Movie: Bonus Material (2002) ===

      https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/51.html

      Thu Jun 16 16:47:40 HKT 2022, 2781 (Content-Length), 2781 (HTML Length)
      203 (Number of Words)

      search 15
      movi 15
      titl 10
      veggietal 8
      jonah 7

      Parent Links:
      1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

      Child Links:
      1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

      Similar Pages

100   === The Powerpuff Girls Movie (2002) ===

      https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/84.html

      Thu Jun 16 16:47:41 HKT 2022, 8532 (Content-Length), 8532 (HTML Length)
      690 (Number of Words)

      movi 18
      powerpuff 11
      rate 11
      imdb 11
      user 10

      Parent Links:
```

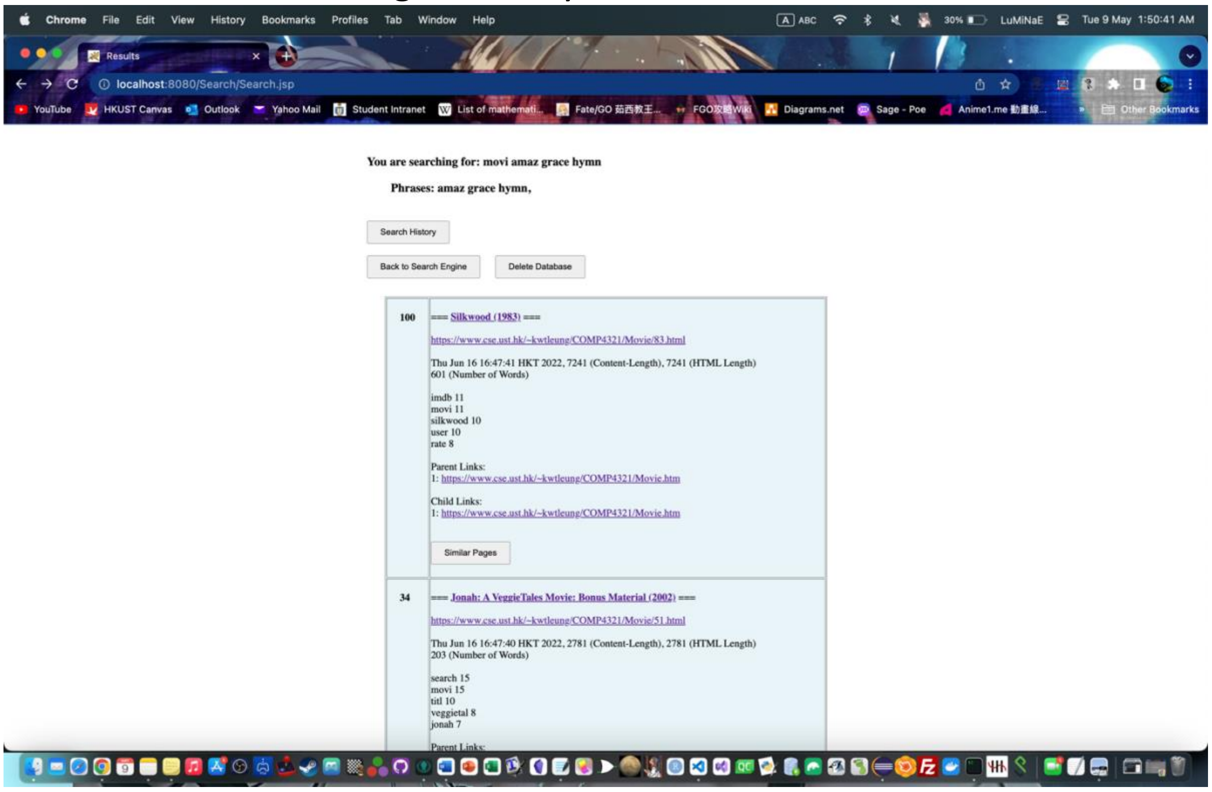
Search movie “Amazing Grace” Hymn


You are searching for: movi amaz grace hymn

Phrases: amaz grace,

```
100   === Silkwood (1983) ===

      https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/83.html

      Thu Jun 16 16:47:41 HKT 2022, 7241 (Content-Length), 7241 (HTML Length)
      601 (Number of Words)

      imdb 11
      movi 11
      silkwood 10
      user 10
      rate 8

      Parent Links:
      1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

      Child Links:
      1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm

      Similar Pages

93    === Unconstitutional: The War on Our Civil Liberties (2004) ===

      https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/206.html

      Thu Jun 16 16:47:40 HKT 2022, 7114 (Content-Length), 7114 (HTML Length)
      562 (Number of Words)

      archiv 16
      imdb 11
      user 10
      movi 9
      rate 8

      Parent Links:
```

Search movie “Amazing Grace Hymn”


localhost:8080/Search/Search.jsp

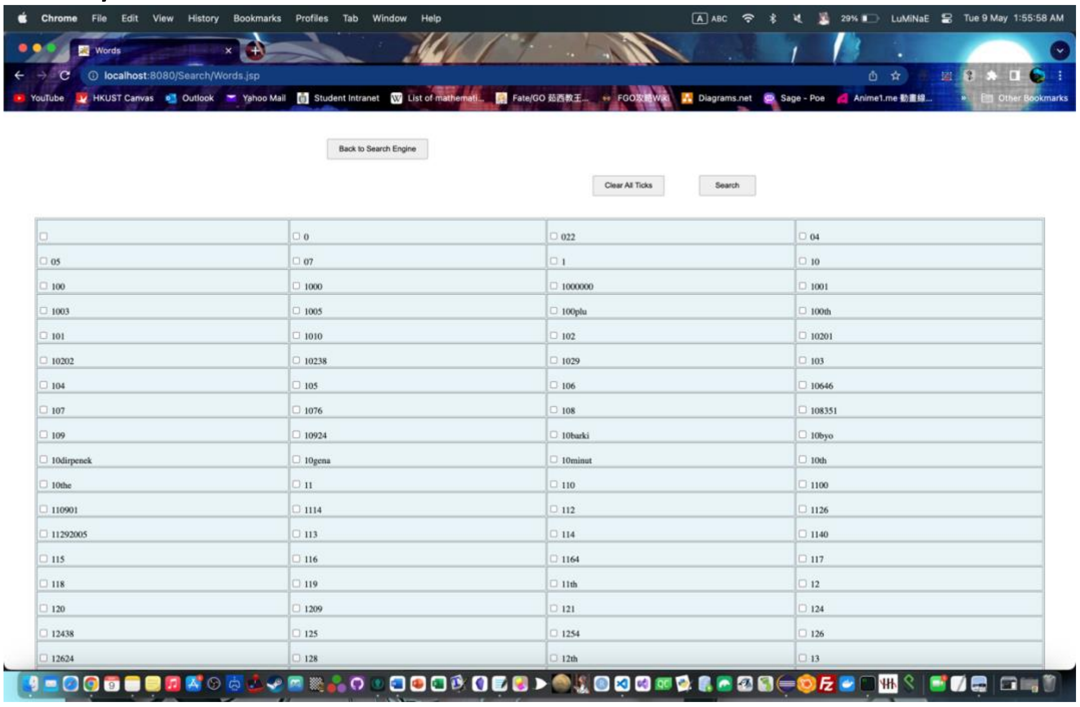
You are searching for: movi amaz grace hymn

Phrases: amaz grace hymn,

100	=== Silkwood (1983) === https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/83.html Thu Jun 16 16:47:41 HKT 2022, 7241 (Content-Length), 7241 (HTML Length) 601 (Number of Words) imdb 11 movi 11 silkwood 10 user 10 rate 8 Parent Links: 1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm Child Links: 1: https://www.cse.ust.hk/~kwtleung/COMP4321/Movie.htm Similar Pages
34	=== Jonah: A VeggieTales Movie: Bonus Material (2002) === https://www.cse.ust.hk/~kwtleung/COMP4321/Movie/51.html Thu Jun 16 16:47:40 HKT 2022, 2781 (Content-Length), 2781 (HTML Length) 203 (Number of Words) search 15 movi 15 titl 10 veggietal 8 jonah 7 Parent Links:

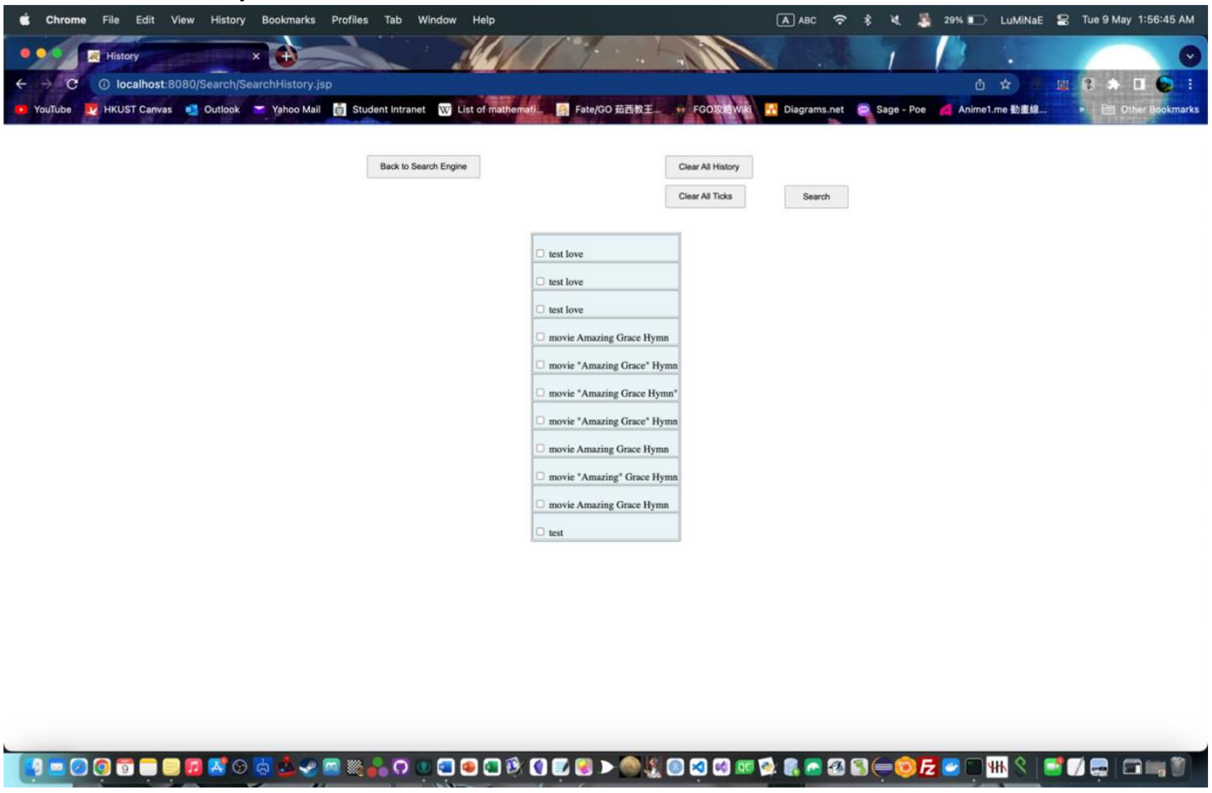
Unconstitutional…(2004) contains the phrase “Amazing Grace” in page while Silkwood(1983) contains the phrase “Amazing Grace Hymn” in page

Dictionary


localhost:8080/Search/Words.jsp

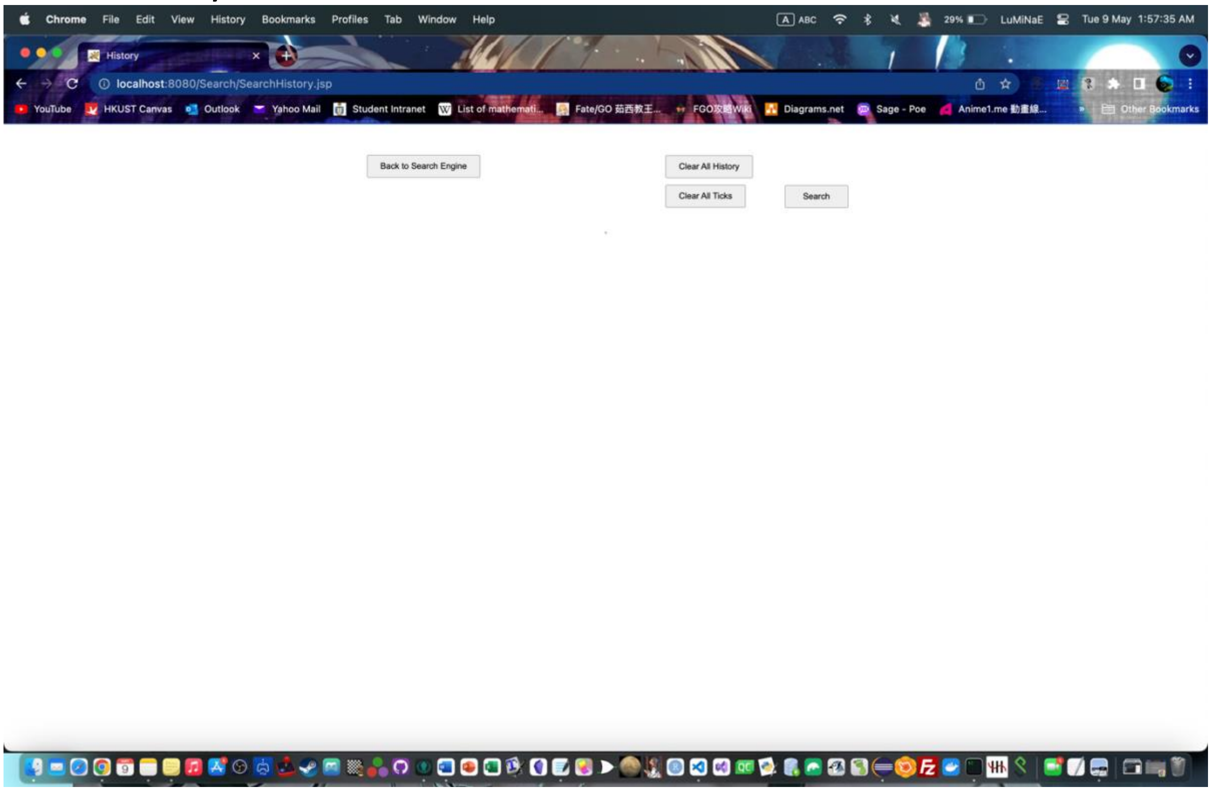
☐	☐ 0	☐ 022	☐ 04
☐ 05	☐ 07	☐ 1	☐ 10
☐ 100	☐ 1000	☐ 1000000	☐ 1001
☐ 1003	☐ 1005	☐ 100plu	☐ 100th
☐ 101	☐ 1010	☐ 102	☐ 10201
☐ 10202	☐ 10238	☐ 1029	☐ 103
☐ 104	☐ 105	☐ 106	☐ 10646
☐ 107	☐ 1076	☐ 108	☐ 108351
☐ 109	☐ 10924	☐ 10barki	☐ 10byo
☐ 10dirpenek	☐ 10gena	☐ 10minut	☐ 10th
☐ 10the	☐ 11	☐ 110	☐ 1100
☐ 110901	☐ 1114	☐ 112	☐ 1126
☐ 11292005	☐ 113	☐ 114	☐ 1140
☐ 115	☐ 116	☐ 1164	☐ 117
☐ 118	☐ 119	☐ 11th	☐ 12
☐ 120	☐ 1209	☐ 121	☐ 124
☐ 12438	☐ 125	☐ 1254	☐ 126
☐ 12624	☐ 128	☐ 12th	☐ 13

Search History



☐ test love
☐ test love
☐ test love
☐ movie Amazing Grace Hymn
☐ movie "Amazing Grace" Hymn
☐ movie "Amazing Grace Hymn"
☐ movie "Amazing Grace" Hymn
☐ movie Amazing Grace Hymn
☐ movie "Amazing" Grace Hymn
☐ movie Amazing Grace Hymn
☐ test

Clear History and Reload


localhost:8080/Search/SearchHistory.jsp
Back to Search Engine
Clear All History
Clear All Ticks
Search