COMP4321 Spring 2017

Project Final Submission

Search Engine Design Document

littleGoogle

Author:

Mohammad Arslan Arshad (20216202, maarshad)
Mang Tik Chiu (20276758, mtchiu)
Yao-Chieh Hu (20216239, yhuag)
Yu-Ning Huang (20213421, yhuangbl)

Overview

Our search engine consists of 4 main components, namely Crawler / Indexer, Database Manager, Querier and the Web Interface. The 4 components introduce modularity yet efficient search experience with detailed information.

I. Crawler / Indexer

Our **Crawler** roots at the starting test page and crawls all pages that are reachable from it. Using a BFS searching technique, it recursively visits each child link and index all the page information. From each web document, the crawler first checks if it has been indexed before or modified by comparing with existing entries. If the web page has not been visited or the last modification date is renewed, its meta-data (Title, Size, Last modification date, etc), contained words and parent/children links are processed and saved to our database. The condition checking ensures that no same pages are visited twice and cyclic links are well handled. After crawling for a specified number of pages or no more pages are to be crawled, it terminates and precomputes **tf-idf** score and the **PageRank** for each document. The lists of scores are then saved in our database for maximum query speed in the future since no further document score computation is required.

II. Database Manager

On top of the raw record manager provided by JDBM, we built our own Database classes that handles all the database management methods. Specifically, we have implemented **Index** and **MapTable** classes. Index class is built to handle entries with multiple attributes, for instance a document with multiple words to be recorded; MapTable class is built to handle id-to-value mapping, insertion and extraction. For runtime efficiency, MapTable consists of both forward and backward HashMap which allows bidirectional extraction in **O(1)** time. At last, a **Database** class is built as a wrapper for all used tables, which includes Inverted Index, Forward Index, id-word MapTable, etc.

III. Querier

To facilitate frontend to backend cooperation, a **Querier** is built in Java to handle user queries and other information retrieval requests. Querier is the class where most runtime similarity and ranking comparisons take place. Specifically, when a user inputs a query phrase, the query string is sent to Querier for processing. In here, multiple operations will be carried out, which are quoted string extraction, query tf-idf weighting, query-title/document **cosine similarity computation** and score ranking. Bonus features such as user **query history** and **query suggestion** are also handled in this class.

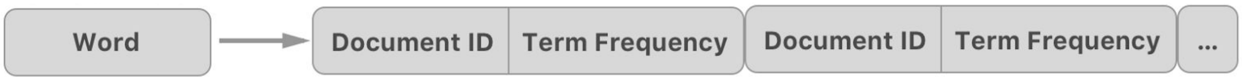
IV. Web interface

Our web interface provides front end graphical user interface for users to search for documents. In addition to a basic search box, we improve user experience by providing **past query phrase suggestion**, **stemmed word database interaction** and **similar page recommendation**. We also allow users to specify **number of results to return** and **CosSim/PageRank weighting**.

Database Design

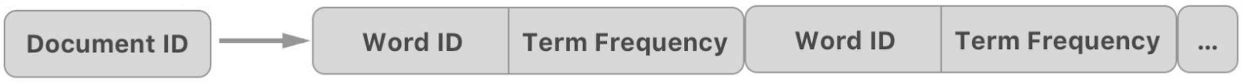
V. Inverted Index Table (General & Title)

Given a specific term/word, this table is to store a list of documents and the corresponding term frequency. It speeds up the relevant documents searching when providing a related keyword.



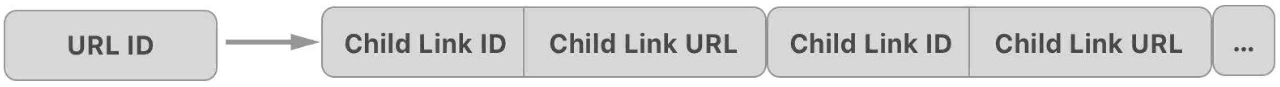
VI. Forward Index Table (General & Title)

Given a document ID, this table is to store a list of word IDs and its corresponding term frequency. It provides some vital information such as the length of a document and the total size of document stored to be searched.



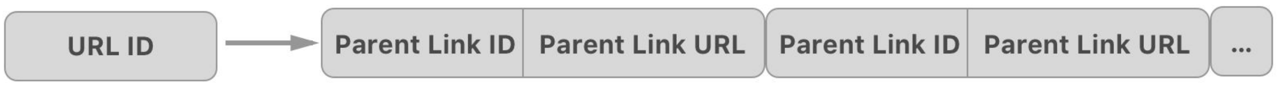
VII. Link Index Table

Given a URL ID, this table is to store a list of child link IDs and its corresponding child link URL. It provides the relationships between documents by examining the links in between, which could be used in page ranking.



IV. Parent Index Table

Given a URL ID, this table is to store a list of parent link IDs and its corresponding parent link URL. It provides the relationships between documents by examining the links in between, which could be used in page ranking.



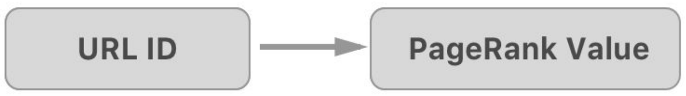
V. MapTable (Word & URL)

This MapTable is to build up the mapping relationship between word and its corresponding word ID, as well as URL and its corresponding URL ID. In so doing, it provides bidirectional data/ID retrieving for both word-domain and URL-domain searching.

Word ID
Word
URL ID
URL
Word
Word ID
URL
URL ID

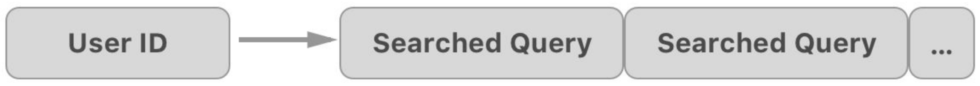
VI. Rank Table

As Rank table does not involve bidirectional relationship, this table is implemented using HTree instead of MapTable.
Given every document, the program calculates the PageRank of the document based on parents of the document which can be derived from Parent Index Table.



VII. History Table

As History table uses tilde ("~") instead of white space as the delimiter, this table is implemented using HTree instead of Index Table.
Given a User ID, this table is to store a list of queries that were searched by the specific user previously. It could be used in generating personalized search history.



Featured Highlights

I. Personalized searched query history

1. Introduction

Personalized search query history provides each user to have a look on the queries he has searched before.

2. Design

When adding a query searched into the history table, a tilde (“~”) is added in front of the value as a delimiter. The keys of the table are the user IDs got in cookies.

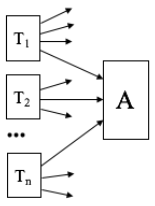
II. Google PageRank

1. Introduction

Google PageRank is a way to measure the importance of the documents. The importance of each document is directly related to the parent documents which are pointing to it. If the parent document is of great importance (i.e. large PageRank value) and has only a few links pointing out, it will contribute more PageRank value to the document.

2. Design

The PageRank algorithm for every document A is extracted from the slides. In the project, we imitate the example given in class to make the damping factor equals to 0.85. And after experimenting with different numbers of iterations, we set the number of iteration to 50 for PageRank values to converges.



$$PR(A) = (1-d) + d\left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + ... + \frac{PR(T_n)}{C(T_n)}\right)$$

III. Query Suggestion

1. Introduction

Query suggestion provides users better chance to optimize their search. It suggests a modified query upon the query given by user as assists.

2. Design

The user query is broken down into word segments, which are filtered by stop-words set. The segments can not be recognized will be extracted for nearest word comparison to propose a most possible alternative. The Levenshtein distance algorithm allows the segment to be compared with all the words in the wordMapTable and it dynamically finds out the best fit for the segment’s replacement. The Levenshtein distance between two word strings a and b is shown below:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

IV. Title Favor Search

1. Introduction

Title favoring search improves the importance of a match between the query and the title. It enhances the score of the document containing query words in its title by tfidf scheme.

2. Design

The matches in the title is treated independently for further score adding, multiplied by the title favor weights specified, default to be 2.0. Three databases are especially constructed for this feature, including titleInvertedDB, titleForwardDB and titleVsmDB.

V. Toggle function on number of results and score weights

1. Introduction

To allow users to customize their user experience, we provide the functionality for users to change the number of results returned and the weights for similarity score and PageRank.

2. Design

There are two bars in the home pages for users to set their desired number of results and weights for similarity score and PageRank. The final score used to rank the result equals to weight input * similarity score + (1-weight input) * PageRank value.

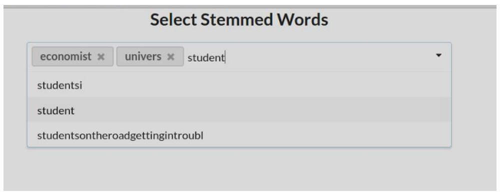
VI. Stemmed words

1. Introduction

To allow users to choose the combination of keywords they want and submit it as a query.

2. Design

The keywords are extracted from word HashMap.



Testing Demonstration

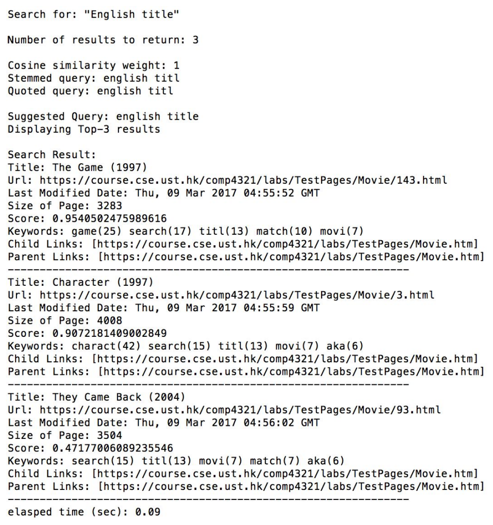
I. Normal query vs Quoted query

1. Aim

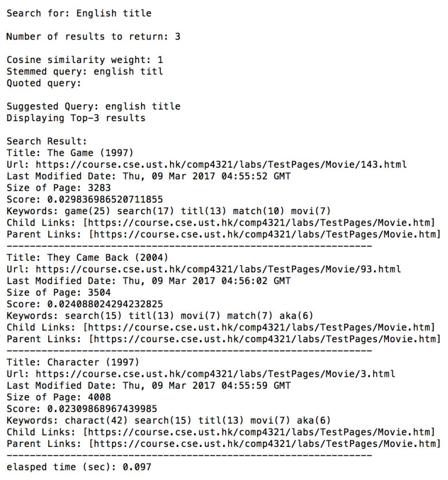
Compare with quoted search to observe result differences.

2. Demonstration

The query **“English title”** returns the result on the left; while the query **English title** (without quotation marks) returns the result on the right.



```
Search for: "English title"

Number of results to return: 3

Cosine similarity weight: 1
Stemmed query: english titl
Quoted query: english titl

Suggested Query: english title
Displaying Top-3 results

Search Result:
Title: The Game (1997)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/143.html
Last Modified Date: Thu, 09 Mar 2017 04:55:52 GMT
Size of Page: 3283
Score: 0.9540502475989616
Keywords: game(25) search(17) titl(13) match(10) movi(7)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
Title: Character (1997)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/3.html
Last Modified Date: Thu, 09 Mar 2017 04:55:59 GMT
Size of Page: 4008
Score: 0.9072181409002849
Keywords: charact(42) search(15) titl(13) movi(7) aka(6)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
Title: They Came Back (2004)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/93.html
Last Modified Date: Thu, 09 Mar 2017 04:56:02 GMT
Size of Page: 3504
Score: 0.47177006089235546
Keywords: search(15) titl(13) movi(7) match(7) aka(6)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
elasped time (sec): 0.09
```



```
Search for: English title

Number of results to return: 3

Cosine similarity weight: 1
Stemmed query: english titl
Quoted query:

Suggested Query: english title
Displaying Top-3 results

Search Result:
Title: The Game (1997)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/143.html
Last Modified Date: Thu, 09 Mar 2017 04:55:52 GMT
Size of Page: 3283
Score: 0.029836986520711855
Keywords: game(25) search(17) titl(13) match(10) movi(7)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
Title: They Came Back (2004)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/93.html
Last Modified Date: Thu, 09 Mar 2017 04:56:02 GMT
Size of Page: 3504
Score: 0.024088024294232825
Keywords: search(15) titl(13) movi(7) match(7) aka(6)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
Title: Character (1997)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/3.html
Last Modified Date: Thu, 09 Mar 2017 04:55:59 GMT
Size of Page: 4008
Score: 0.02309868967439985
Keywords: charact(42) search(15) titl(13) movi(7) aka(6)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
--------------------------------------------------------------
elasped time (sec): 0.097
```

3. Result

“Character” is rank higher than “They Came Back” in quoted search. This is because in “They Came Back”, “title” appears 16 times but only 1 of them is in the phrase of “English title”. On the other hand, in “Character”, “title” also appears 16 times but “English title” appears twice.

II. Customize number of returning results

1. Aim

To test if users can specify number of results to be returned.

2. Demonstration

Setting the variable topK as 2 and 5 yield different results.

3. Result

Correct number of results are returned.

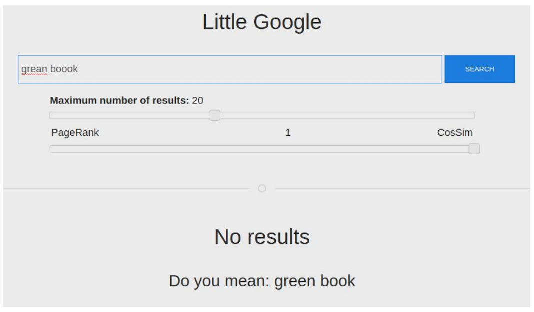
III. Search for query containing typos

1. Aim

To test if the system can suggest a query with typos corrected.

2. Demonstration

For this case the user search for “grean boook” as query.



Little Google

grean boook

Maximum number of results: 20

No results

Do you mean: green book

3. Result

The suggested query turns out to be “green book”.

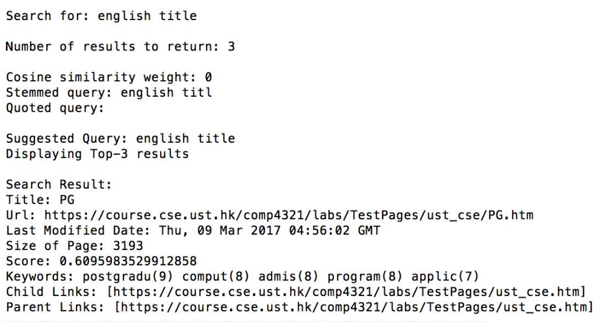
IV. Customize the weight for cosine similarity score

1. Aim

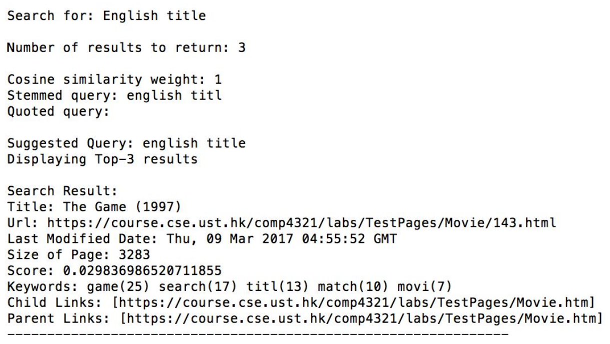
To test if users can specify the weight for cosine similarity score.

2. Demonstration

The query on the left specifies the weight for cosine similarity score = 0; while the query on the right specifies the weight for cosine similarity score = 0.



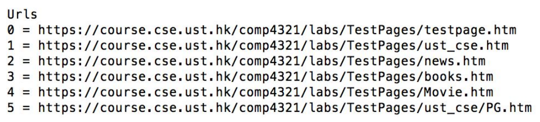
```
Search for: english title

Number of results to return: 3

Cosine similarity weight: 0
Stemmed query: english titl
Quoted query:

Suggested Query: english title
Displaying Top-3 results

Search Result:
Title: PG
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/ust_cse/PG.htm
Last Modified Date: Thu, 09 Mar 2017 04:56:02 GMT
Size of Page: 3193
Score: 0.6095983529912858
Keywords: postgradu(9) comput(8) admis(8) program(8) applic(7)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/ust_cse.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/ust_cse.htm]
```



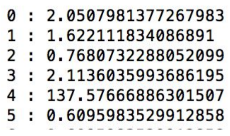
```
Search for: English title

Number of results to return: 3

Cosine similarity weight: 1
Stemmed query: english titl
Quoted query:

Suggested Query: english title
Displaying Top-3 results

Search Result:
Title: The Game (1997)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/143.html
Last Modified Date: Thu, 09 Mar 2017 04:55:52 GMT
Size of Page: 3283
Score: 0.029836986520711855
Keywords: game(25) search(17) titl(13) match(10) movi(7)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
---------------------------------------------------------------
```

3. Result

Cosine similarity score = 0 means the ranking is completely based on PageRank. On the other hand, cosine similarity score = 1 means the ranking is completely based on cosine similarity score.
To further check the result, from the Database we can know that the ID of PG page is 5. And this page has a PageRank value of 0.6095983529912858 which is consistent with the score we got in the left hand side.



```
Urls
0 = https://course.cse.ust.hk/comp4321/labs/TestPages/testpage.htm
1 = https://course.cse.ust.hk/comp4321/labs/TestPages/ust_cse.htm
2 = https://course.cse.ust.hk/comp4321/labs/TestPages/news.htm
3 = https://course.cse.ust.hk/comp4321/labs/TestPages/books.htm
4 = https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm
5 = https://course.cse.ust.hk/comp4321/labs/TestPages/ust_cse/PG.htm
```



```
0 : 2.0507981377267983
1 : 1.622111834086891
2 : 0.7680732288052099
3 : 2.1136035993686195
4 : 137.57666886301507
5 : 0.6095983529912858
```

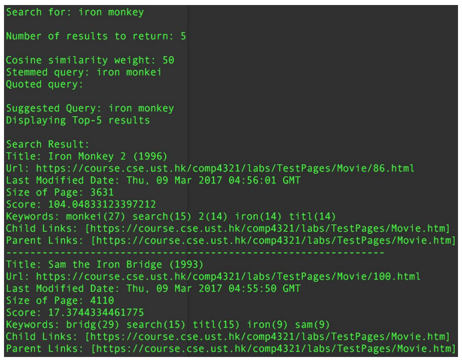
V. Search for query containing exact title specified

1. Aim

To test if the user can search for a document with exact title specified.

2. Demonstration

For this case the user search for “Iron Monkey” as query.



```
Search for: iron monkey

Number of results to return: 5

Cosine similarity weight: 50
Stemmed query: iron monkei
Quoted query:

Suggested Query: iron monkey
Displaying Top-5 results

Search Result:
Title: Iron Monkey 2 (1996)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/86.html
Last Modified Date: Thu, 09 Mar 2017 04:56:01 GMT
Size of Page: 3631
Score: 104.04833123397212
Keywords: monkei(27) search(15) 2(14) iron(14) titl(14)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
---------------------------------------------------------------
Title: Sam the Iron Bridge (1993)
Url: https://course.cse.ust.hk/comp4321/labs/TestPages/Movie/100.html
Last Modified Date: Thu, 09 Mar 2017 04:55:50 GMT
Size of Page: 4110
Score: 17.3744334461775
Keywords: bridg(29) search(15) titl(15) iron(9) sam(9)
Child Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
Parent Links: [https://course.cse.ust.hk/comp4321/labs/TestPages/Movie.htm]
```

3. Result

The first returned result should be “Iron Monkey 2 (1996)”, with the title highly matched. And the second returned result should be “Sam the Iron Bridge (1993)”, with the title partially matched. It proves the functionality of the title-favor-search.

Conclusion

I. System Analysis

1. Strength

a. **Highly personalized and customizable**
Users can specify weighting between Cosine Similarity and PageRank score. In addition, each user with a unique user id has his/her query history stored. This assists the user when he/she queries for similar results in the future.

b. **Higher level of page ranking**
On top of cosine similarity ranking on user's queries, we extract any quoted phrase and favors documents containing corresponding phrases. Furthermore, the PageRank algorithm is used to further refine the relevance of the top search results. We favor pages containing query words in the title which ensures another level of relevance.

2. Weakness

a. **Highlighting**
We haven't implemented the highlighting effects for past visited pages and matched queries. Users might forget which pages he/she has visited or queried which might consume more time when browsing through various sites.

b. **Code redundancy**
Due to collaboration and separate implementation, some codes might be redundant in terms of functionality. For example, Database management sessions can be further merged into 1 big class instead of several subclasses.

II. Re-implement Design Difference

1. System modularity

A better team communication can result in a more concise and neat code modularity. To reduce code redundancy and maximize neatness of codes, functionalities making use of the same classes can be implemented and wrapped together instead of separately, which could improve readability of code and efficiency in implementation.

2. System flow integrity

Instead of putting all functionalities into 1 class, such as putting both crawling and indexing functionalities in **Crawler**, a separate class **Indexer** could have been implemented and linked to **Crawler**. Instead of having index functions in crawling actions, a final toolbox which handles the entire workflow could have been built. This would have improved the overall flow of the system.

III. Prospective Features and Improvement

1. Personalized past visited webpages

The feature is useful for users to revisit the sites they have been to and to retrieve the useful information they previously found. To implement this function, we could consider generating another table. When users click on the results from the search engine, the result links are put into the table.

2. Matched phrase highlighting

Matched phrase highlighting can help users to better understand the contents of the documents given as the search result by the search engine. To implement this function, we could consider using pattern matching algorithm such as Knuth-Morris-Pratt (KMP) method to do full text scanning.