A Focused Web Search Engine: UR Search Engine!

1. Introduction:

WWW (World Wide Web) is a hub of huge amount of data. Internet connect all to this WWW. To learn from these big data, we need to do searching to get desired data. Only then data becomes asset for individuals. In this project, I develop a focused search engine name "UR Search Engine". This search engine serves like all other search engine. Unfortunately, because of data storage limitation I limit this engine up to 50 maximum data indexing capability. Although, this can work beyond this limit. Design of my web crawler is done by PHP as programming language. Besides, I use HTML, UNIX command, SQL, MySQL server and so on for database and data indexing. Popular BFS (Breadth First Search) algorithm is used for crawling on the WWW to collect data. Full-Text search using Boolean Mode is used for searching and ranking results. TF/IDF (Term Frequency/ Inverse Document Frequency) algorithm is used for ranking the searching results.

2. System Structure:

Following figure 1 illustrate the complete structure of UR search engine. From login to indexing and then from searching to ranking the results.

3. Methods applied:

UI (User Interface) of my search engine is designed by simple HTML language. I develop my crawler using php programming language. The very basic idea to develop this crawler is gained from Dr. Wen-Chen Hu's website[1]. Inside php code I use *system()* function for running Unix command. Lynx text browser is used for dumping source code of a page when crawling URLs [2] [3] [4]. I design several queries for this search engine. For query processing I use SQL query inside php. For database design I use MySQL server. I create the structure of my database using DDL language. BFS (Breadth First Search) algorithm is used for crawling URLs based on seed URL[5] [6].

I use Full – Text search for searching and ranking from database. I found this Full-Text search very efficient because at the same time this is doing multiple operation such as stop words removal, searching against given keywords and ranking the search result based on TF/IDF (Term Frequency/ Inverse Document Frequency) [7] [8] [9] [10] [11] [12] .

Full-Text search support MySQL server and we can design our searching queries against character-based data from database table. Database table must have one or more full-text supported index declaration before implementation Full-Text search. Only VARCHAR, CHAR and Text datatype is supported for Full-Text search [9] [8] [7].

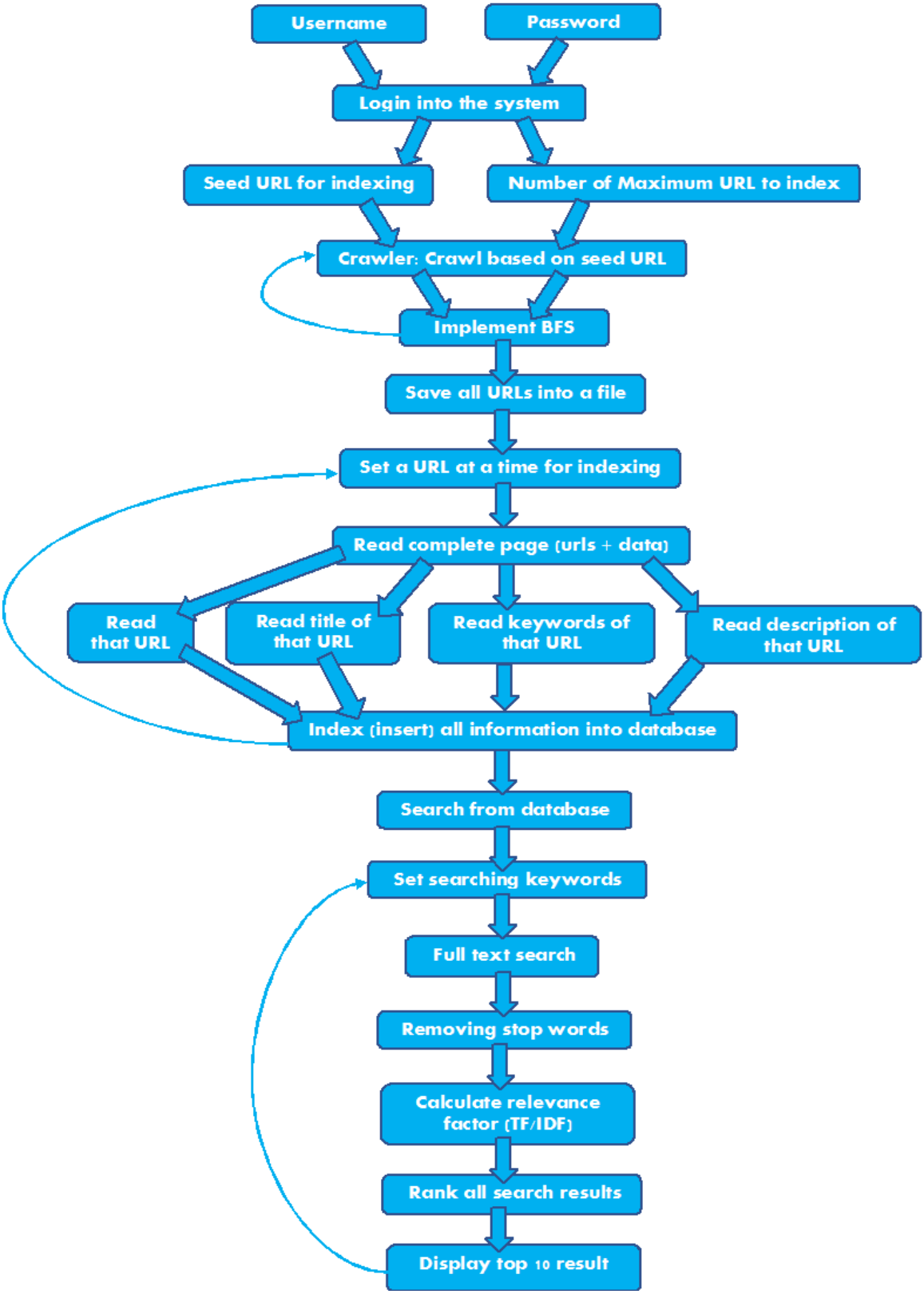

Fig. 1: System structure of UR search engine

I am using Boolean mode for this Full-Text search. Besides searching this mode returns ranking of the search result [10]. This mode support +/- operator to include/remove a word from searching keywords. This also support >,<,~,* and so on Boolean operators in searching.

3.1 Relevancy Rankings (Boolean Mode Search):

Design of Full-text search is based on the Sphinx full-text search engine[10]. Algorithms of this search use BM25 for searching and TF-IDF for ranking [10] [9]. "*The TF-IDF weighting is based on how frequently a word appears in a document, offset by how frequently the word appears in all documents in the collection. In other words, the more frequently a word appears in a document, and the less frequently the word appears in the document collection, the higher the document is ranked"*[10].

3.2 Calculation of Relevancy Ranking:

The term frequency (TF) value is the number of times that a word appears in a document. The inverse document frequency (IDF) value of a word is calculated using the following formula, where total_records is the number of records in the collection, and matching_records is the number of records that the search term appears in.

${IDF} = log10(${total_records} / ${matching_records})

When a document contains a word multiple times, the IDF value is multiplied by the TF value:

*${TF} * ${IDF}*

Using the TF and IDF values, the relevancy ranking for a document is calculated using this formula:

*${rank} = ${TF} * ${IDF} * ${IDF}*

This complete paragraph is copied from [10]. For multiple words search the ranking equation should be like the following.

${rank} = ${TF} * ${IDF} * ${IDF} + ${TF} * ${IDF} * ${IDF}

I use this Full-Text search on two tables and three columns. Those columns contain VARCHAR data on keyword, url_title and url_description from each webpage. So, my search is based on three columns of data from my MySQL database.

4. Experiment result:

Data indexing works without any error in this project. I can insert desired amount of data based on seed URL and based on regular expression for keyword, title and description my crawler can separate and insert those data into my database. An example scenario of the expected result is given in the following figure 2. From this figure we can see that my crawler can get 1465 URLs based on given seed URL. And then insert keyword, url_title and url_description into the tables of MySQL database. This case, the crawler indexed 50 records because I limit maximum number as 50.

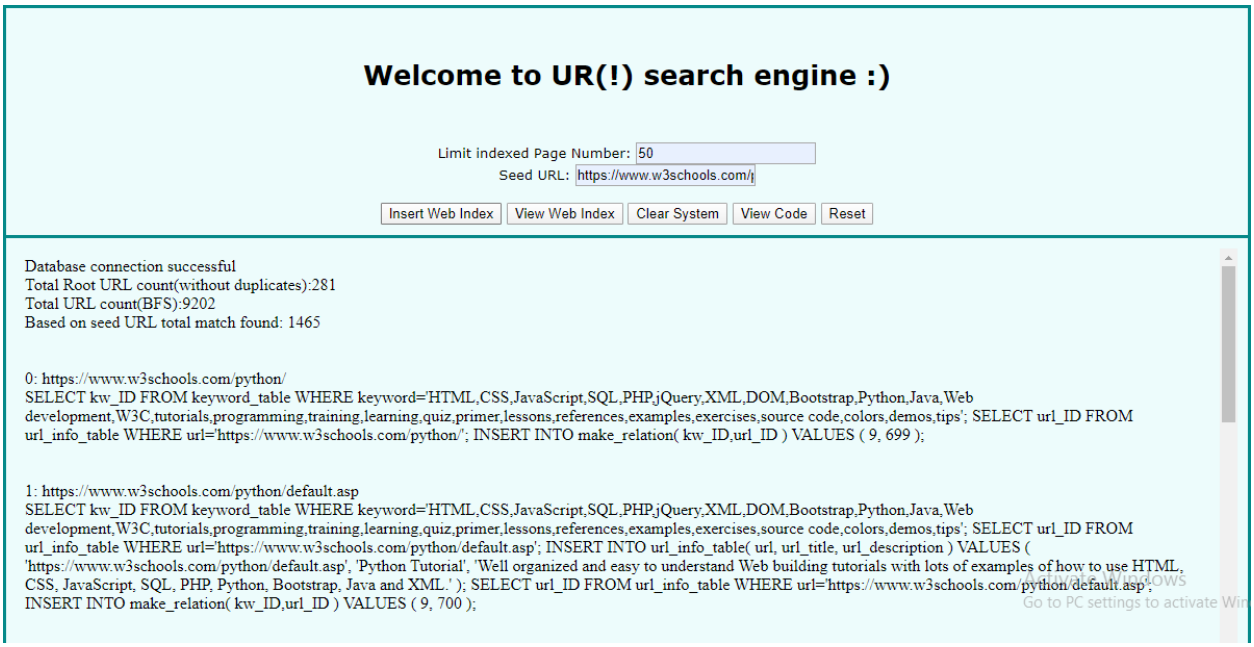

Fig. 2: Example scenario for data indexing

Searching and Ranking algorithm gives expected result for this project. Implementation of Full-Text search successfully works and return perfect result for my search engine. An example scenario is shown in the following figure 3. In this example, I use four keywords for searching from database. This engine searches these keywords and then rank them based on TF/IDF algoritm.

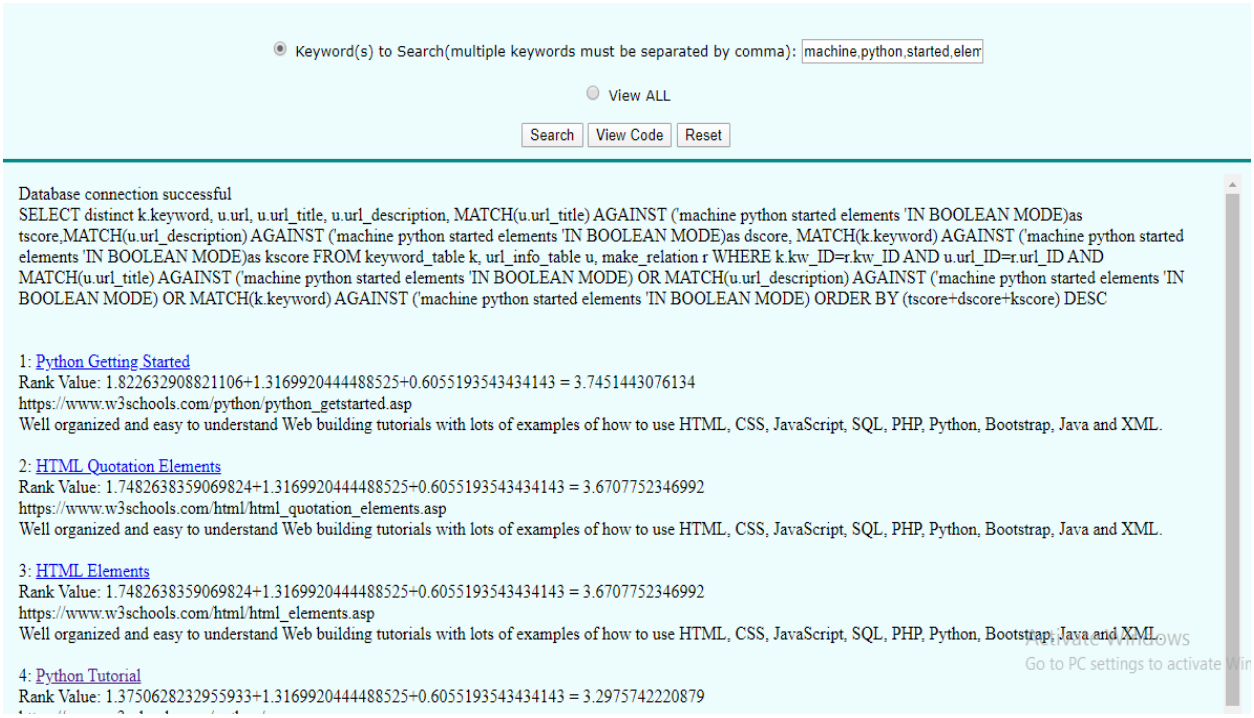

Fig. 3: Example scenario for searching and ranking

5. Conclusion:

UR Search Engine works like a regular search engine within a limited circumstance. Successful implementation of BFS and TF/IDF (Term Frequency/ Inverse Document Frequency) algorithm is ensured in this project. Although, the crawler doesn't give perfect result for a few URls during data indexing. Otherwise this search engine is fully functional. I want to continue this work to improve the performance of this engine and fix those issues.

References:

[1] W.-C. HU, "The Code Sample of Indexing," *undcemcs01.und.edu/~wen.chen.hu/course/515*. [Online]. Available: http://undcemcs01.und.edu/~wen.chen.hu/course/515/2/8.html. [Accessed: 02-Dec-2020].

[2] Lynx.browser.org/, "Lynx." [Online]. Available: http://lynx.browser.org/. [Accessed: 02-Dec-2020].

[3] Lynx.invisible-island.net, "Lynx_users_guide." [Online]. Available: https://lynx.invisible-island.net/lynx_help/Lynx_users_guide.html.

[4] Stackoverflow.com, "Easiest way to extract the urls from an html page." [Online]. Available: https://stackoverflow.com/questions/1881237/easiest-way-to-extract-the-urls-from-an-html-page-using-sed-or-awk-only.

[5] Wikipedia.org, "Breadth-first search." [Online]. Available: https://en.wikipedia.org/wiki/Breadth-first_search.

[6] Stackoverflow.com, "DFS vs BFS in web crawler design." [Online]. Available: https://stackoverflow.com/questions/20579169/dfs-vs-bfs-in-web-crawler-design.

[7] M. Lord, "Rankings with InnoDB Full-Text Search," *mysqlserverteam.com*, 2014. [Online]. Available: https://mysqlserverteam.com/rankings-with-innodb-full-text-search/. [Accessed: 04-Dec-2020].

[8] Mysqltutorial.org, "Creating FULLTEXT Indexes for Full-Text Search," *mysqltutorial.org*. [Online]. Available: https://www.mysqltutorial.org/activating-full-text-searching.aspx.

[9] W3resource.com, "MySQL Full text search," *w3resource.com*, 2020. [Online]. Available: https://www.w3resource.com/mysql/mysql-full-text-search-functions.php.

[10] Mysql.com, "Boolean Full-Text Searches," *dev.mysql.com*. [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/fulltext-boolean.html.

[11] Mysql.com, "Natural Language Full-Text Searches," *dev.mysql.com*. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/fulltext-natural-language.html.

[12] Stackoverflow.com, "Keyword search using PHP MySql?," *stackoverflow.com*. [Online]. Available: https://stackoverflow.com/questions/980376/keyword-search-using-php-mysql.