Search Engine for UNT Domain

Lakshmi Naga Vardhan Rao Pakalapati

Dept. of Computer Science and Engineering

University of North Texas

LakshmiNagaPakalapati@my.unt.edu

ABSTRACT:

This project mainly involves in implementing a Search Engine based on vector space model. The domain for the project is UNT domain pages. The search engine takes all the links from the domain and crawl the documents. It accepts the query and matches it with the crawled documents. The overview of the project involves that the crawler which follows BFS is used to crawl the data, the vector space is used for inverted indexing of these crawled documents and a User Interface is used for accepting the queries and computes similarity with the documents. Then it outputs the results in the form of top 5 links also gives an option to get the next five in the order.

Keywords: Search Engine, vector space model, crawling, BFS.

1.INTRODUCTION:

Increase in the amount of data on the web creates new challenges in Information Retrieval. A search Engine helps in finding the related documents or text very easily. The total project is explained in various sections. In section 2, it tells about the main components used in building a search engine. In Section 3, we have the implementation part which involves execution of the various modules and the execution of the search engine. Section 4 deals with the Evaluation & Results followed by Challenges in section5 and other strategies in sections 6. Future work in section 7 and conclusion in section 8.

2. COMPONENTS:

Crawler:

The crawler [5] in this project crawls all the web pages of the domain by using the strategy of Breadth First Search (BFS) and keeps track of previously visited links data. The crawler grabs all the links and returns the list of URLs crawled.

Vector Space Model:

Vector space model [1] implemented for the project deals with processing the content of the crawled URLs to build an inverted index for this crawled document collection. In this module, the irrelevant data and tags will be removed followed by the process of tokenization.

Inverted Index:

The inverted index consists of the data regarding all the tokens of the documents and the term frequency, the inverted document frequency are also stored which are required to find the similarity between query and the crawled documents. The tf-idf for all the document collection is calculated by using the formula:

$W_{ij} = tf_{ij} * idf_i = tf_{ij} * \log_2 (N/df_i)$

Similarity Measure:

The similarity measure computed for the implementation in this project is cosine similarity. For each query term and document the cosine similarity is computed by taking values from pre-computed inverted index using the following formula:

$$\frac{\sum x_i y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i y_i}$$

The similarity is measured by calculating the inner product of the document d_j and the query q by using the formula:

$$sim(d_j, q) = d_j q = \sum_{i=1}^{t} w_{ij} w_{iq}$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query.

Interface:

When we run the main.java of the server, it prompts to start the server, after starting the server open any web browser and connect to localhost: 8585, it displays a search tool. When we enter a query it displays the top 5 results' links and also shows an option to get next 5 results and so on.

3. IMPLEMENTATION:

Used Tools:

The tools that are used for the development of the search engine are as follows

Programming language: JAVA.

IDE: NetBeans IDE 8.1.

Software's:

JSOUP [12]: it's a java library which is used to work with real time HTML.

JETTY: web server [10] which is nothing but the server used.

SQLite – JDBC [11]: which is used for the purpose of creating the database file to store the crawled data, precisions and tf-idf of all the tokens.

Modules:

Various modules are used in developing the project. A brief description of the important modules is described below.

Crawling Package:

Crawl.java: This class traverse the web in BFS fashion and stores all the unique links in a set. It is written in such a way that it will stop crawling after it reaches a certain no of links that we specified.

Preprocessor.java: this class will tokenize [2] the given document into various tokens.

Main.java: This class will initiate the crawling and after crawling it will retrieve each document text using JSOUP and passes it to preprocessor. Preprocessor tokenizes and then passes the tokens back to main class. This process repeats for all URLs. After all URLs have been processed, it will calculate the term frequencies and tf-idf and store the data in Database file.

Vectorspace package:

MyTokenizer.java: This class tokenizes by removing the punctuations.

Main.java: This class is used to calculate the term frequency (TF) and inverted document frequency (IDF).

Server package:

All the classes in this package are used to develop a User Interface for giving the query and getting the query results. The server will start by running the Main.java.

Db package:

SQLlite.java: It consists of the SQL queries. The purpose of this class is to create a database file to store the data obtained from crawling the domain.

Execution:

First web crawling happens in BFS. After getting all URLs, it goes to each URL and gets all the text present in the given URL using JSOUP. After getting text, it tokenizes and calculates frequencies of words. After processing all links it will calculate tf and tf-idf. Then checks for the similarity with the given query. Let's see how to execute the written code.

Crawling:

The process of execution starts by running the Main.java of the crawling package. When we run that, it calls the class Crawl and it starts crawling the links in breadth First Search (BFS) fashion. We have given 3000 as the limit after reaching the limit it stops and now starts processing the obtained URLs. In this process of processing the URLs it obtains data from URLs, tokenizes them and calculates the TF-IDF for every document. All this data will be stored in the Test.db file. After finishing the process of crawling now we have to run the server.

Server:

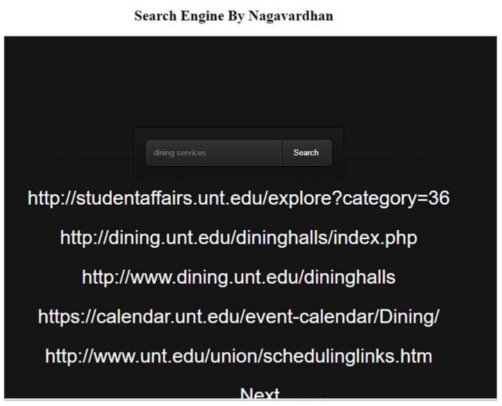
The next part after crawling is starting and running the server. I have used Jetty web server as the server for implementation. To start the server, run the Main.java file in the Server package. It takes few seconds and then it prompts to start the server. When we click the start button in the prompt, the server will start. Now open any web browser and in the URL tab enter as "localhost:8585" and proceed. Now it displays the screen of the search engine developed.

Search:

There is a search [4] tab where we the query is inserted. Insert any query and search it. It takes few seconds because it has to match the similarity measures of the query with the existing ones. After that, it displays the top most 5 relevant links, there is also an option to go to the next page of links which shows the next best 5 and so on. If you click on a particular link it will direct you to its web page. Again enter another query and so on. This is how the developed search engine will be executed.

4. EVALUATION AND RESULTS:

A manual evaluation has been done for 15 queries and I am including the results for 3 of those queries in Figure1, 2, 3.


Search Engine By Nagavardhan

dining services

http://studentaffairs.unt.edu/explore?category=36

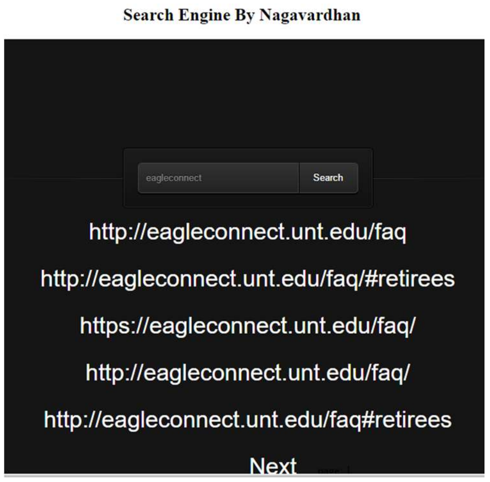
http://dining.unt.edu/dininghalls/index.php

http://www.dining.unt.edu/dininghalls

https://calendar.unt.edu/event-calendar/Dining/

http://www.unt.edu/union/schedulinglinks.htm

Figure1 – Query: dining Services


Search Engine By Nagavardhan

eagleconnect

http://eagleconnect.unt.edu/faq

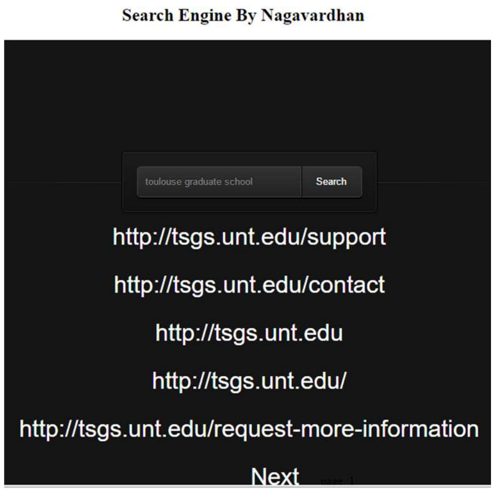
http://eagleconnect.unt.edu/faq/#retirees

https://eagleconnect.unt.edu/faq/

http://eagleconnect.unt.edu/faq/

http://eagleconnect.unt.edu/faq#retirees

Figure2 – Query: eagleconnect


Search Engine By Nagavardhan

toulouse graduate school

http://tsgs.unt.edu/support

http://tsgs.unt.edu/contact

http://tsgs.unt.edu

http://tsgs.unt.edu/

http://tsgs.unt.edu/request-more-information

Figure3- Query: toulouse graduate school

The results are almost relevant to the given queries.

5. CHALLENGES:

The main challenges that are encountered during the building of the search engine are:

Time Complexity:

The time taken to crawl the links takes more time, it took almost 92 mins to crawl all the 3000 URL links. Also the time that takes for showing the results after entering the query is a bit long of on an average it takes 30-40 secs for the first query. Apart from 1st query if we enter another query it doesn't take any time. It shows results instantly for the rest of the queries.

Storing the data:

The storing of the data that is obtained during crawling of the domain I have used a database file. It occupies more space as it stores huge amount of data for 3000 links.

6. COMPARISON WITH ALTERNATIVES:

Similarity:

The similarity can be calculated by using various methods. The method used in this project to compute the similarity between the query and the document of the document collection is the Cosine Similarity [3]. The various alternative methods may be Dice Coefficient or Jaccard Coefficient also can be used. But the previous works on the Similarity Measures [7] concludes that Cosine Similarity is the better one.

Crawler:

The crawler can be implemented by both Depth-First Search (DFS) and Breadth-First Search (BFS). I have used BFS because the previous works [9] shows that BFS can yield better results than DFS.

7. FUTURE WORK:

The efficiency can be improved using query expansion which can be obtained by considering the synonyms for the query words. By removing the limit of 3000 links, we can get efficient results but it will be a difficult task with this one. Need more optimized code to reduce the time complexity. A better User Interface can also reduce some of the time complexity.

8. CONCLUSION:

This project is a simple search engine based on vector space model that searches queries and gives outputs only related to the UNT domain. The implementation of this gives a clear idea of an IR system. A search engine play a crucial role in the web. As the data is increasing day by day, many new challenges arise in Information Retrieval [6] Systems.

REFERENCES:

[1] http://www.cse.unt.edu/~ccaragea/cse5200/lectures/lecture5-invidx.pdf, Vector space model,

[2], http://www.cse.unt.edu/~ccaragea/cse5200/lectures/lecture3-text.pdf, Text Processing.

[3] http://www.cse.unt.edu/~ccaragea/cse5200/lectures/lecture6-eval.pdf, Evaluation

[4] http://www.cse.unt.edu/~ccaragea/cse5200/lectures/lecture8-hits.pdf, Web search

[5]http://www.cse.unt.edu/~ccaragea/cse5200/lectures/lecture10-crawling.pdf, Web crawling,

[6] Introduction to Information Retrieval, http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[7] A. Huang. Similarity measures for text document clustering. In Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, pages 49{56, 2008.

[8] E. Loper and S. Bird. Nltk: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume1, pages 63{70. Association for Computational Linguistics, 2002.

[9] M. Najork and J. L. Wiener. Breadth-first crawling yields high-quality pages. In Proceedings of the 10th international conference on World Wide Web, pages 114 -118. ACM, 2001.

[10] Jetty web server, http://download.eclipse.org/jetty/9.2.14.v20151106/dist/jetty-distribution-9.2.14.v20151106.zip

[11] SQLite, https://bitbucket.org/xerial/sqlite-jdbc/downloads

[12] JSOUP, https://jsoup.org/download