Customized Search Application

Thingujam Joseph
Computer Science
Engineering & IT
Assam Don Bosco University
Guwahati, India

Bethsheba J. Rapthap
Computer Science
Engineering & IT
Assam Don Bosco University,
Guwahati, India

Nupur Choudhury
Computer Science
Engineering & IT
Assam Don Bosco University
Guwahati, India

ABSTRACT
The main purpose of the search application is to help users find appropriate in-formation that they are looking for. The user needs to login first in order to per-form his search. The Admin will upload the document and the search application will store all the words contain in the document in the database table with some details of the word like how many times the word is present, which document it is present, etc. The system will then search the results of a specific targeted query entered by the user by calculating the tf-idf value and based on the tf-idf value, the result will be displayed. This result is nothing but lists of documents which is most appropriate or relevant to the query that the user has entered. This project also aims at implementing some of the features like customizing by displaying previously viewed sites of each user, recently uploaded documents, etc. It is implemented in PHP, HTML and with MySQL as the database.

Keywords
tf-idf, tokenization, parsing, indexing, parser etc.

1. INTRODUCTION
Search engines or search applications are programs that searches the results of a particular targeted query entered by a user and returns a list of documents or search results which is most appropriate to the keyword. There are many popular search engines like Google, Yahoo, Bing, etc.

The Search application includes:

A. An algorithm to upload a document and breakdown the contents of document into words.

B. An algorithm to store all the words to each respective database table i.e., based on the alphabets the words begin with.

E.g. Dog will be stored in the table name 'D'.

C. An algorithm which receives user query, search the word or words in the table, calculate the tf-idf value of the documents (documents containing the words) and return with the result to the user based on the tf-idf value.

2. FEATURES OF THE APPLICATION
2.1 System Architecture Body Text

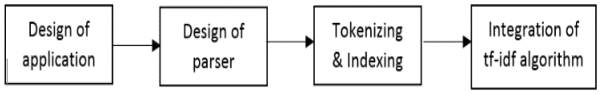

Figure.2.1.2: Architecture of the Customized Search

2.2 Flowchart

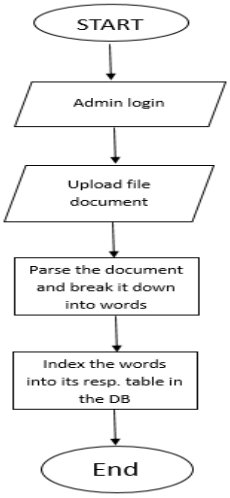

Figure 2.2.1: Flowchart of Customized Search Application (Parsing And Indexing)

As shown in Figure 2.2.1, the Admin will upload documents. When Admin up-loads a document, document parsing will take place. The application will then store the words in its respective table depending on the letter each word begins with and some details are calculated like how many times a particular word is pre-sent in the document, how many words are present in the document, etc. and this information is stored with the words. This information is calculated and stored in the table beforehand thus when a user enters a query, calculating the tf-idf value for the document with these words will be easier and faster.

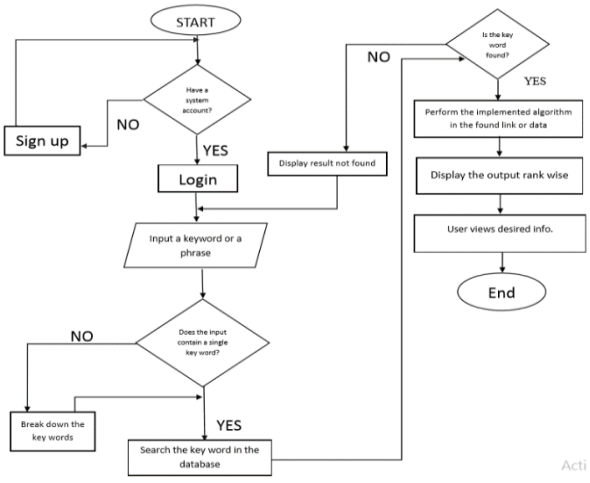

Figure 2.2.2.: Flowchart of Customized Search Application

As shown in figure 2.2.2, the User will enter the query and the application will search the query word by word. If the word is contain in the table (depending on the letter it begins with), it will calculate the tf-idf of the document for each word. If two or more different words is contain in the same document, the tf-idf value for those words will be added thereby increasing in the tf-idf value of that document. After all the words in the query is calculated, it will display the result to the user. The result is nothing but a list of documents arranged in a descending order respective of the tf-idf value calculated.

3. METHODOLOGY

3.1.Module 1 Requirement Analysis

3.1 Minimum Hardware Requirement

Processor : Intel Core Duo 2.0 GHz or more

RAM : 256 MB

Disk space : 2 GB

3.2 Minimum Software Requirement

Language : PHP, HTML

Database : MySQL

Operating System : Window (32 or 64 bit)

3.2.Module 2 Techniques

a. Document parsing

Document parsing breaks the words of a document for insertion into the forward and inverted indices. The words found are called tokens. Hence, parsing is more commonly referred to as tokenization in the context of search engine indexing and natural language processing.

b. Tokenization

Computers do not understand the structure of a document and cannot automatically recognize words and sentences. Since this system is concentrated only on text documents, hence during tokenization, the parser in this system identifies sequences of characters which represent words.

c. Indexing

The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. After uploading and parsing the docu-ment the system stores all the words to each respective database table i.e., based on the alphabets the words begin with. E.g. Dog will be stored in the table name ‘D’. Hence when the user search for the word dog, the system will search only in the table name ‘D’. Without an index, the search application would scan every document in the database, which would require considerable time and computing power.

3.3. Module 3 Algorithms

Term Frequency-Inverse document Frequency (tf-idf)

Tf-idf means term frequency-inverse document frequency. To evaluate how im-portant a word is to a document in a database, the tf-idf weight is used. The im-portance increases to the number of times a word appears in the document. The tf-idf weight is used by search engine in scoring and ranking a document's rele-vance to the user query. The ranking functions is computed by summing the tf-idf for each query term.

How to Compute:

3.1. The tf-idf weight first computes the normalized Term Frequency (TF), i.e., the number of times a word appears in a document, divided by the total number of words in that document.

3.2. The second is the Inverse Document Frequency (IDF), which measures how important a term is. It is computed as the number of the documents in the database divided by the number of documents where the term appears.

TF: Term Frequency measures how frequently a term occurs in a document. It is possible that a term would appear much more times in long documents than shorter ones since every document is different in length. Thus, the term frequency is often divided by the document length (the total number of terms in the document).

TF (t) = (Number of times term t appears in a document) / (Total number of terms in the document).

A document receives a higher value if it contains a query term more often. The sum is to be computed over the query terms, of the match values between each query term and the document.

IDF: Inverse Document Frequency reflects how important a term is in a document. All terms are considered equally important while computing the term frequency. However certain terms such as ‘is’, ‘of’, and ‘the’, may appear a lot of times but these words have little importance. Thus we need to put in less value down the terms which occur most frequently by computing the following:

IDF (t) = log_e (Total number of documents / Number of documents with term t in it).

Certain terms have little importance in determining relevance. Thus we need to scale down the terms with high collection frequency or the total number of occurrences of a term in the database.

Example:

If we have a document containing 500 words in which the word boy appear 10 times. Then the time frequency (i.e., tf) for the word boy is then (10/500) = 0.02. Now, assume we have 10,000,000 documents and the word boy appears in 1000 of these. Then the inverse document frequency (i.e., idf) is calculated as log (10,000,000 / 1,000) = 4. Thus, the tf-idf value is obtained from the product of tf and idf: 0.02*4=0.08.

4. RESULTS

Some of results that were obtained in the form of snapshots are shown below

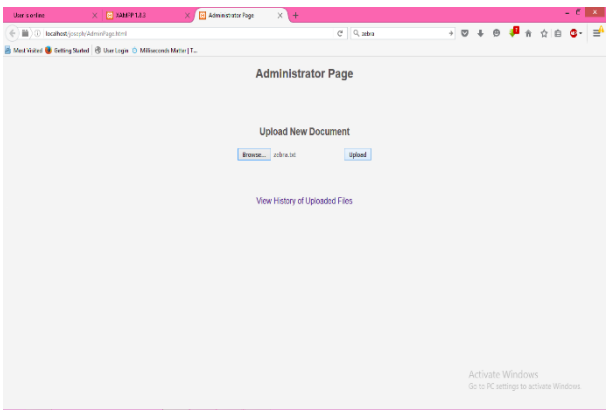

Figure 1: Word Table before file is uploaded

As shown in Figure 1, Admin chose the document name ‘zebra.txt’ to upload.

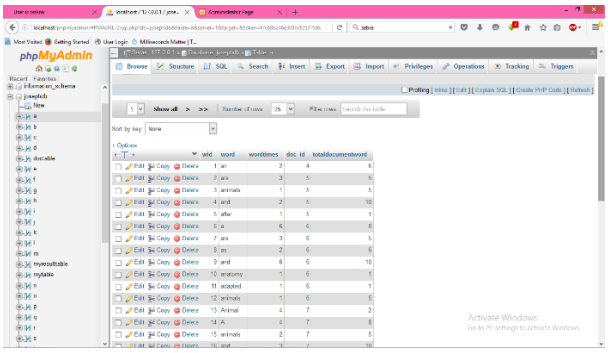

Figure 2: Word Table before file is uploaded

Figure 2 shows the Word Table before the file ‘zebra.txt’ is uploaded. Word table contains the document Id (document Id for each document is obtain from the document table), word (the words in a document), word times (how many times the word appears in that particular document), total document words (in how many documents the word is present).

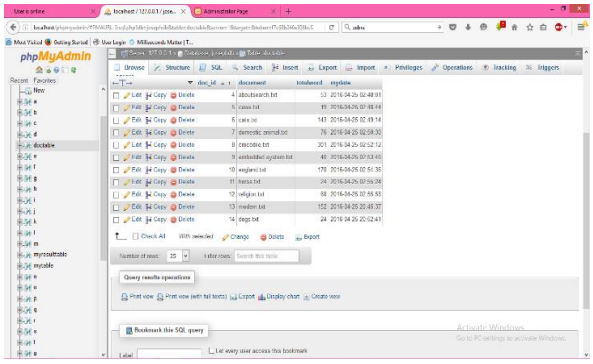

Figure 3: Document table before file is uploaded

Figure 3 shows the Document Table before the file ‘zebra.txt’ is uploaded.

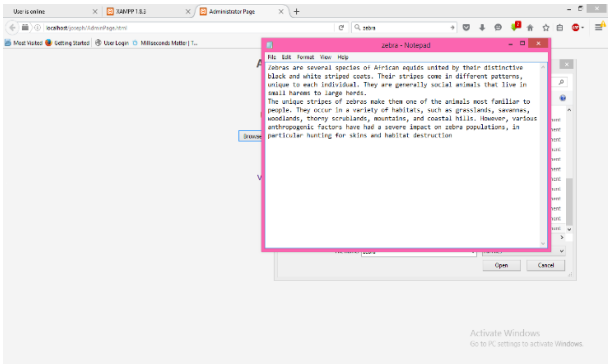

Figure 4: Contents of the Document Zebra

The contents of the document ‘zebra.txt’ are shown in the Figure 4.

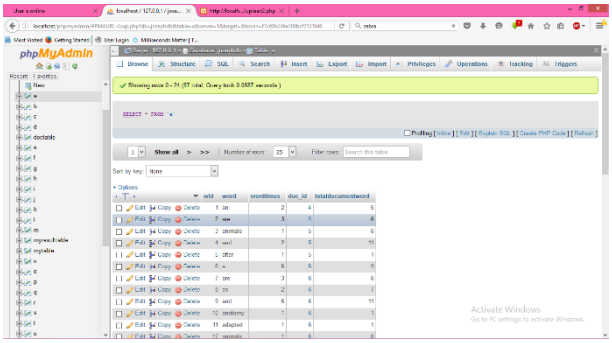

Figure 5: Word table after uploading file

Figure 5 shows the Word Table after the file ‘zebra.txt’ is uploaded.

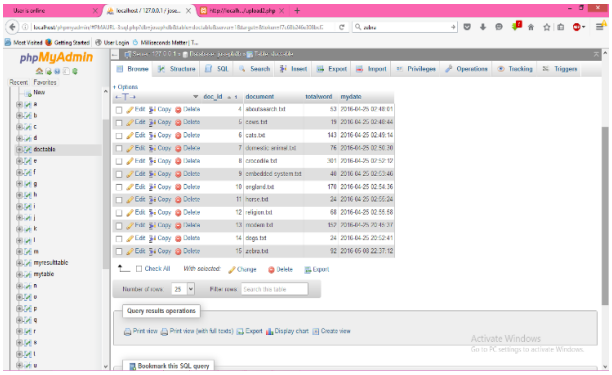

Figure 6: Document table after uploading file

Figure 6 shows the Document Table after the file ‘zebra.txt’ is uploaded.

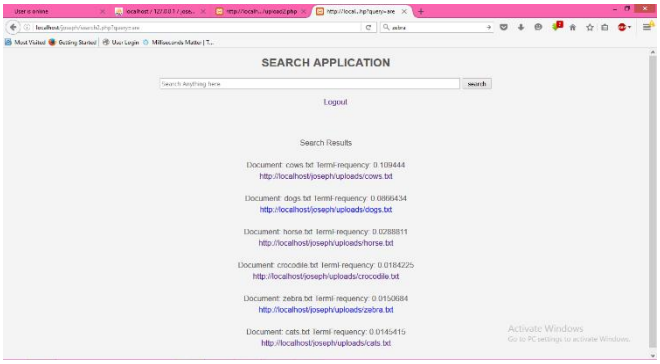

Figure 7: Search results of the word ‘animal’ (displaying result with tf-idf value)

In Figure 7, the user enters the query ‘animal’ and the search results of the word ‘animal’ is shown with the tf-idf value.

Note: If the query entered by the user contains more than one word, the system calculate the tf-idf value of documents for each word. If the words are contained in the same document, the tf-idf value is added.

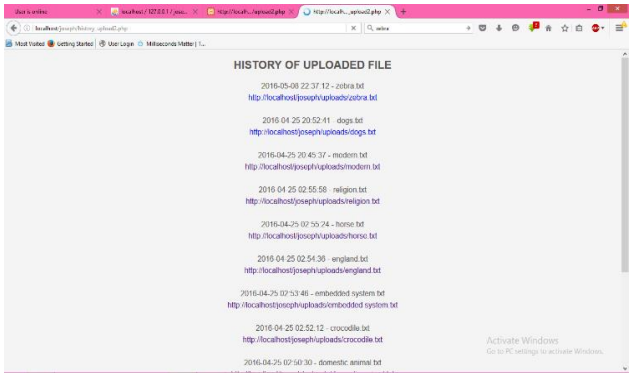

Figure 8: History of uploaded file

Figure 8 shows the history of the files which were uploaded by the Admin.

5. CONCLUSION

Search engine or Search application offers users with information that is more relevant to the query entered by the user.

5.1.Limitation

Since it is a text based Search Application its primary input and output are based on text rather than alphanumeric, graphics or sound. This project is concentrated only on text documents and not on documents or file which contain images or videos.

5.2.Future Work

For the future work, we will try to implement stemmer. A stemmer is computer program or an algorithm that helps us in reducing words to their stem, base or root form. For example, a stemming algorithm reduces the words washing, washed and wash to the root word wash. Stemming is mostly used to increase the efficiency of a search engine. We will also try to implement crawler.

6. REFERENCES

[1] K. Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval". Journal of Documentation, 28 (1). 1972.

[2] G. Salton and Edward Fox and Wu Harry Wu. "Extended Boolean information retrieval". Communications of the ACM, 26 (11). 1983.

[3] G. Salton and M. J. McGill. "Introduction to modern information retrieval". 1983

[4] G. Salton and C. Buckley. "Term-weighting approaches in automatic text retrieval". Information Processing & Management, 24 (5). 1988.

[5] H. Wu and R. Luk and K. Wong and K. Kwok. "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, 26 (3). 2008.

[6] Jeffrey 3A. Hoffer, Joey F. George, Joseph S. Valacich, “Modern Systems Analysis and Design”, Dorling Kindersley(India) Pvt. Ltd, 2009.

[7] http://rankwatch.com/blog/a-brief-introduction-to-search-engines/

[8] http://www.brighthubpm.com/project-planning

[9] http://infolab.stanford.edu/pub/papers/google.pdf

[10] https://en.wikipedia.org/wiki/search_engine