Implementing of Content-Based Desktop Search System

Sanjeev Kumar, Rohit Kumar, Umang Akash, NIT Kurukshetra, Feb 2011
Project Guide : Prof Sanjay Kumar Jain, NIT Kurukshetra

Abstract-***Currently there are many open-source content and name based search engine tools are available which have different functions and features. Just as these open-source full-text search engine tools, there is an excellent one that is discussed here. This paper designed and implemented a desktop content based search system for text document files.***

I. INTRODUCTION

The name based search engines are available from the earlier times. The search engine search for the keyword in the name of the file and produces output if file found with that name.

But this criteria for searching becomes inefficient if the content stored in file doesn't match with file name, so the text/content based search introduces .The full-text search engine is the one that can search each word in documents. The full-text search engine first indexes for each document, then the search system will search the index database according to the keywords which are inputted by users. The search results will return to users according to a certain sorting algorithm. The characteristic of the full-text search engine is huge amount of information. It segments the whole content of the document and adds to the index database so that it has high recall ratio. The full-text search engine also has the characteristic of short cycle, rapid development and cheap cost.

Today there are often massive of documents stored in our PCs, we often spend much time on finding documents which are we needed. The desktop search systems index the large number of documents, so that they can locate the needed documents immediately. The desktop search systems solve the problems of the difficulty of finding the right document. With a number of open-source search engine tools, we can design our own personal desktop search system conveniently and efficiently.

II. IMPLEMENTATION

This paper developed a simple desktop system for searching the files either by the name of the file or by their content present in them. This application is developed in the perl programming language.

The function of desktop search system is very simple. They can search for individual file name in case of the name based search. If the search is based on the content, it create index for all documents in your computer system and have function of full text searching.

The architecture is shown in figure 1.The application domain shows that users should handle original documents first, usually the documents are at the same directory. Then the system will do a series of treatments for all of the documents. Finally, the index of each document will be added to the index library. If users want to search these documents, they should provide the query keywords, the system will automatically find the corresponding index data from the index library according to the query keywords. Finally result will returned to user according to a certain order.

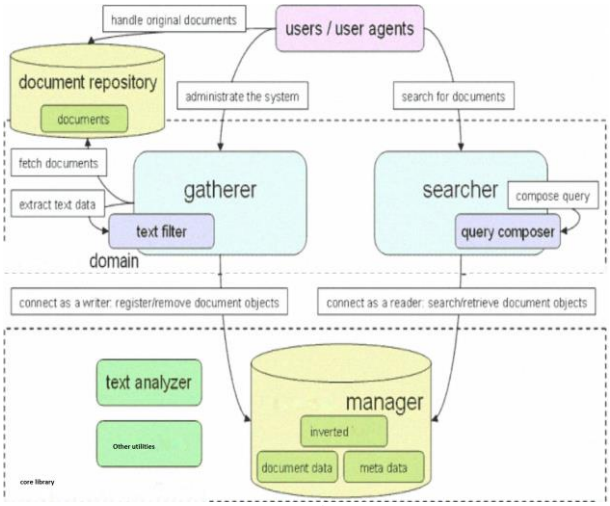


Fig 1. Architecture of the system

The cat command is used to read a regular text file, catdoc command is used to read the .doc extension files and to read pdf files pdftotext command is used. The berkeley database is used for the text indexing purpose using the tie command for creation and making connection with it. The package used for accessing the hierarchy of the files in the system id Find::File in the perl.

The figure below shows the semantic view of the search engine

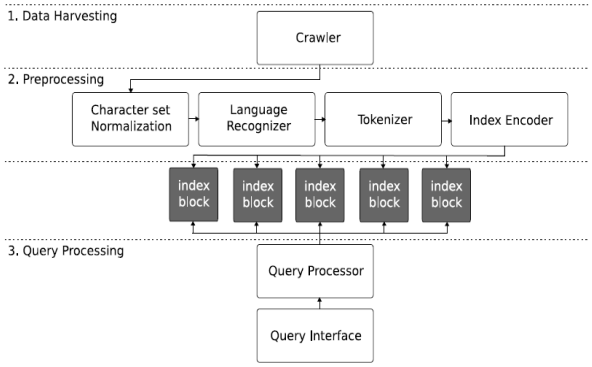


Fig 2. Semantic view of the content based search engine.

A. Indexing Process

The search engine uses a indexer.pl perl script file to create the index. The pseudo code is given below:

Algorithm 1: Create Index

1. Create the database object.
2. Open the database
3. If can't open output the error.

4. for each file
5. Create a document object
6. Add attribute to the document object
7. Add body text to document object
8. Register the document object to database
9. If can't register output the error
10. Close the database
11. If can't close output the error

B. Searching Process.

It uses the search.pl script to search for the files on the required criteria. The pseudo code for search.pl is given below:

Algorithm 2: Search the index

1. Create the database object
2. Open the database
3. If can't open output the error
4. Create search condition object
5. Set the search phrase to search condition object
6. Get the result of the search
7. For each document in the result
8. Retrieve the document object
9. Display the attributes
10. Close the database
11. If can't close output the error

Though this example, the query process is actually divided into two steps: indexing and searching.

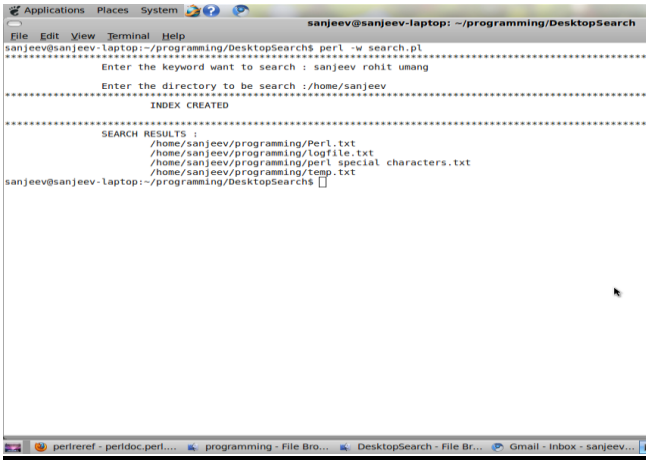
III. PERFORMANCE

The hardware environment of the testing: Intel(R) Core(TM) Core 2 duo T6500 @ 2.00 GHz, 2.00 GB RAM.

The operating system of testing: Linux(Fedora 14), 32-bit.

The documents for testing: *.txt, *.html, *.doc, *.pdf, *.c/cxx at a single directory.
The total size of files in that directory was 210.1 MB.
Firstly the performance of indexing was tested.
The CPU utilization rate was around 40%-55%. The size of the index file index.db created was 47.2MB.
The screenshots of this test are given below:

Figure 3. Indexing and searching the pattern in the files displayed.



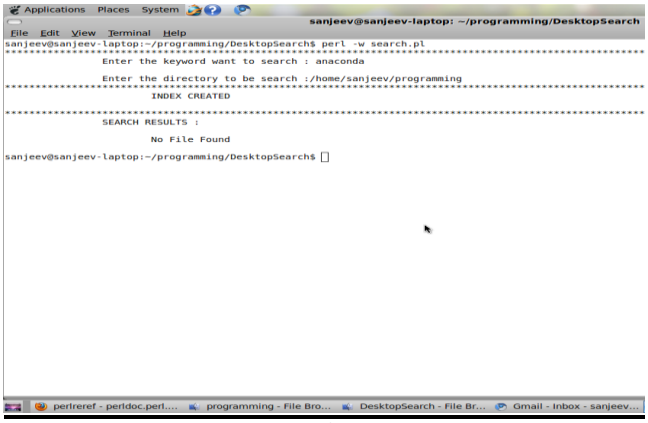


Figure 4. Indexing and searching for pattern which is not in database, gives no file found results.

Through analyzing the experiment results, the system can meet the velocity requirement of indexing and searching. The system has almost perfect recall ratio and high precision ratio. The future work we are faced is under the condition of increasing the amount of data, how we can ensure the efficiency of the system.

IV. CONCLUSION

The paper designed a desktop content and name based based search system and implemented in perl language. After implementing of the system, we also tested the indexing and searching capabilities of the system. The test result shown that the system has two basic characteristics of search engines: good recall ratio and high precision ratio.

REFERENCES

[1] Autofocus: Semantic Search for Desktop by Christiaan Fluit Aduna BV, The Netherland.

[2] Design and Implementation of a Content-Based Search Engine by Villi H. Tuulos, Master's Thesis, University Of Helsinki, Department of Computer Science.

[3] The Beagle++ ToolBox: Towards an Extendable Desktop Search Architecture by Ingo Brunkhorst, Paul – Alexandru Chirita, Stefania Costache, Julien Gaugaz, Ekaterini Loannou, Tereza Iofciu, Enrico Minack, Wolfgang Nejdl and Raluca Paiu.

[4] Leveraging Personal metadata for Desktop Search: The Beagle++ System by Gianluca Demartini, Paul – Alexandru Chirita, Stefania Costache, Julien Gaugaz, Ekaterini Loannou, Tereza Iofciu, Enrico Minack, Wolfgang Nejdl and Raluca Paiu.

[5] Your Unix The Ultimate Guide by Sumitabha Das.

[6] www.perldoc.perl.org/