IJCSN International Journal of Computer Science and Network, Volume 6, Issue 2, April 2017
ISSN (Online) : 2277-5420
www.IJCSN.org
Impact Factor: 1.5

Efficient Information Retrieval Using Indexing

[1] Shweta J. Patil; [2] Dilip K. Budhwant

[1][2] Department of Computer Science and Engineering,
Jawaharlal Nehru Engineering College,
Aurangabad, M.S, India.

Abstract - This paper introduces information retrieval using Hoot, including indexing and system architecture, compares information retrieval of Hoot and information retrieval of Lucene, the experimental results show that Hoot gives efficient information retrieval and has faster retrieval speed than Lucene.

Keywords - *Information Retrieval (IR); System Architecture; Indexing.*

1. Introduction

Information retrieval is mainly considered as a component of computer science that deals with the representation, storage, and access of information[1]. Information retrieval is pertained with the organization and retrieval of information from immensely colossal database Sources [2]. Information Retrieval (IR) is the technique by which a sizably voluminous accumulation of data is represented, stored, and fetched for the purport of cognizance revelation as a result to a utilizer's request or query[3].

The main aim of information retrieval model is to "finding relevant knowledge-base information or a document that fulfill user needs". To achieve this aim, IR usually employ following processes:

In indexing process, the documents are represented in restate content form.

Searching is the main process of IR[1].

The core of information search is the full-text retrieval technology. Full-text search technology provided us with the information retrieval tool according to the content of data[4]. The amount of digitally available information is growing at an exponential rate. A large part of this data consists of text. Each scenario where text is used to express information requires a different form of retrieving information from the text. There is a basic search task, however, that underlies all those applications. String matching is the process of finding the occurrences of a short string (called the pattern) inside a (usually much longer) string called the text.

String matching can be carried out in two forms. Sequential string matching requires no preprocessing of the text, but rather traverses it sequentially to point out every occurrence of the pattern. Indexed string matching builds a data structure (index) on the text beforehand, which permits finding all the occurrences of any pattern without traversing the whole text. Indexing is the choice when-

(i) the text is so large that a sequential scanning is prohibitively costly,
(ii) the text does not change so frequently that the cost of building and maintaining the index outweighs the savings on searches, and
(iii) there is sufficient storage space to maintain the index and provide efficient access to it[5].

A. Information Retrieval Process

Information Retrieval (IR) is the process by which a collection of data is represented, stored, and searched for the purpose of knowledge discovery as a response to a user request (query)[5]. The goal of any information retrieval system is to satisfy user's information need. Unfortunately, characterization of user information need is not simple. User's often do not know clearly about the information need. Query is only a vague and incomplete description of the information need[6].

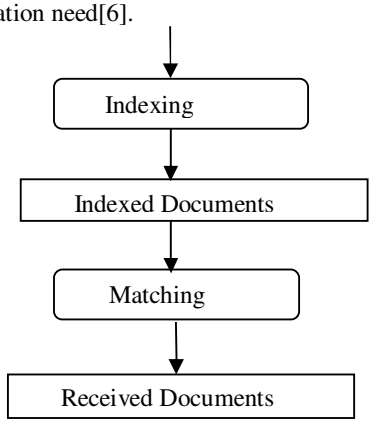

Fig 1 – Information Retrieval Process

Information retrieval process includes-

1. Create index with the model according to the text of database. Indexing can greatly improve the speed of information retrieval. Which way do you use depends on the scale of information retrieval system. Large-scale information retrieval systems such as Google, Baidu take advantage of the approach of inverted index.

2. Search-After indexing the documents, you can start to search information you need. Search requests are submitted by the users and information retrieval systems to preprocess and search the information eventually return user the information[4].

2. Related Work

2.1 Different Information Retrieval software

1. Hoot

Hoot is a framework that can provide users complete query engine, text indexing engine and part of the analysis engine, it can also provide a simple but powerful interface so that people can conveniently and quickly develop the search engine.

Hoot is a smallest application of information retrieval using inverted WAH bitmap index, highly compact storage, operating in database and document modes. Information retrieval is the process of searching for words in a block of text.

Testing Observation:

- Blazing fast operating speed.
- Incredibly small code size.
- Uses WAH compressed BitArrays to store information.
- Multi-threaded implementation meaning you can query while indexing.
- Tiny size only 38kb DLL.
- Highly optimized storage.

2. Lucene

Lucene is one of the Jakarta projects of Apache Software Foundation which is an open source application of information retrieval toolkit, it's not a full text search engine [9].

Testing Observation:

- Average operating speed
- Large code size.
- size is 300kb DLL.
- Less optimized storage.

3. Proposed Information Retrieval Software -HOOT

3.1 Information Retrieval in HOOT

Hoot is mainly composed of analysis, indexing and searching three modules.

Analysis module is responsible for preprocessing document information; the principal role of indexing module is to enhance the speed of retrieval; searching module is mainly used for interacting to users [8].
In Hoot, an index is composed of segments, a segment is made up of documents, a document is composed of fields, and many terms consist of a field.[7].

Once documents are built and analyzed, next step is to index them so that this document can be retrieved based on certain keys instead of whole contents of the document. Indexing process is similar to indexes in the end of a book where common words are shown with their page numbers so that these words can be tracked quickly instead of searching the complete book.

Working of HOOT :

- Set the index storage, it will store all the index information.
- Choose a folder where you want to crawl for content.
- Load Hoot: It will load Hoot so you can query an existing index.
- Start Indexing : It will load Hoot and start indexing the directory you specified.
- Stop Indexing : It will come active after you have started indexing so you can stop the process.
- Count Words : It will show the number of words in Hoots dictionary
- Save Index : It will save anything in memory to disk.
- Free memory : It will call the internal free memory method on Hoot (It will only free the bitmap storage and not release the cache).
- You can search for content, it will show the count and time taken.
- To open the file just double click on the file path in the list box.

3.2 Indexing Techniques

There are several popular information retrieval (IR) indexing techniques; the technique which Hoot used is called Inverted Index.

Inverted index

An inverted index is a special index which stores the words to bitmap conversion data. In a normal index you would store what words are in a certain document,

an inverted index is the opposite given a word 'x' what documents have this word is stored. Bitmap index is the index based record numbers to the document.

Bitmap indexes were formerly also used in Information Retrieval (IR), but are today mainly replaced by inverted indexes. WAH [11] is one of several introduced compression schemes. Although there are schemes with more compact indexes, WAH supports efficient query processing. This combined with the fact that FastBit is openly available motivates the use of WAH-compressed bitmap indexes.

In IR, inverted indexes consist of a search structure for all searchable words called a dictionary, and lists of references to documents containing each searchable word, called inverted lists.

An inverted index for an attribute in a DSS consists of a dictionary of the distinct values in the attribute, with pointers to inverted lists that reference tuples with the given value through tuple identifiers (TIDs).

To reduce both space usage and the I/O requirements in query processing, the inverted lists are often compressed [10].

Steps for Indexing-

- Collect all tagged PDF documents to be indexed into one or more folders.
- You can build an index file from all the tagged PDF files in a set of folders you define.
- you choose a folder where the index will be stored. Indexing proceeds in the background. A small index definition file is created.
- Carry out text searches in the currently open PDF, all PDFs in a given folder or on a prebuilt index file.
- Searches can be done using a single string, multiple search strings, patterns or masks. Use the Organizer to specify the search criteria. A PDF Index file is a searchable archive of PDF documents.

3.3. System Implementation

This paper employed the toolkits of Hoot to simulate information retrieval.

Preprocessing Module: Before using Hoot we need to preprocess the prepared text documents. The mainly role of preprocessing is to convert full-width characters into half-width characters. In order to better display the use of Hoot, this paper will divide the large documents into small documents and assign a unique ID number for each document.

Indexing Module: After processing the document, you can use Hoot to process relevant information. Firstly, create indexing for processing documents. Secondly, build query object; lastly, search in index.

Searching Module: After indexing, system will establish a search class, the class will provide us with two approaches, index search approach is used to search indexing which are built by Hoot. However, string search approach used to search information. First give the search path then parse the string and generate query object to search information. The three main modules are the general process of all information retrieval.

4. Experimental Results

There is training data of organization containing Approximately 15 Lakhs pages which need to be retrieved efficiently and correctly with less amount of time.

The dataset we considered pages for indexing considering some fixed keyword. Here we consider 20 keywords in a single page for searching after indexing in Hoot and Lucene. Result is presented for four cases having 20 keywords per page and obtain following result.

Table 1-Result Analysis of Hoot and Lucene

Software Used	Hoot	Lucene	Retrieval Accuracy of Hoot (%)	Retrieval Accuracy of Lucene (%)
Keyword Searching Rate For Page 1	19 Keywords	17 Keywords	95%	85%
Keyword Searching Rate For Page 2	19 Keywords	16 Keywords	95%	80%
Keyword Searching Rate For Page 3	17 Keywords	16 Keywords	85%	80%
Keyword Searching Rate For Page 4	18 Keywords	17 Keywords	90%	85%
Average Retrieval Accuracy (%)			91.25%	82.50%

Based on searching for fixed keywords we obtain retrieval results by comparing Hoot retrieval with Lucene retrieval. So we can see Hoot's retrieval time-consuming is superior to Lucene retrieval time-consuming. Total indexing time is also less in Hoot as compare to Lucene.

5. Conclusion

Hoot is a full text indexing engine toolkit written in VB.net, multi-user support access, quickly visit indexing time. This paper provides detail analysis of Hoot, indexing and searching and compare the Hoot retrieval with the Lucene retrieval. Accuracy of correctly retrieved documents using Hoot is 91.25%. The experimental results show that Hoot is efficient and it has faster information retrieval speed than Lucene.

6. Future Scope

In future works, we aim to do indexing and searching for handwritten documents in order to speed up the information retrieval within less time.

References

[1] Balwinder Saini*,Vikram Singh,Satish Kumar "Information Retrieval Models and Searching Methodologies: Survey" Balwinder Saini,Vikram Singh,Satish KumarInternational Journal of Advance Foundation and Research in Science & Engineering (IJAFRSE) Volume 1, Issue 2, July 2014.

[2] R. Baeza-Yates and B. Ribeiro –Neto, "Modern Information Retrieval", Harlow: ACM Press 1999.

[3] H. Dong, F.K. Husain and E. Chang, "A Survey in Traditional Information Retrieval Models", IEEE International Conference on Digital Ecosystems and technologies, pp. 397-402,2008.

[4] Rujia Gao, Danying Li, Wanlong Li, Yaze Dong, Application of Full Text Search Engine Based on Lucene, Advances in Internet of Things, October 2012, 2, 106-109.

[5] GONZALO NAVARRO, "Compressed Full-Text Indexes", ACM Computing Surveys, Vol. 39, No. 1, Article 2, Publication date: April 2007.

[6] Niranjan Lal, Samimul Qamar, Savita Shiwani, Information Retrieval System and challenges with Dataspace, International Journal of Computer Applications (0975 – 8887) Volume 147 – No. 8, August 2016.

[7] Y. C. Li and H. F. Ding, "Research and Application of Full-Text Search Engine Based on Lucene," Computer Technology and Development, Vol. 20, No. 2, 2010, pp.4-56.

[8] T. Liu, B. Qin, Y. Zhang and W. X. Che, "Introduction to Information Retrieval System," China Machine Press, Beijing, 2008.

[9] B. Y. Lin, R. Zhao and L. C. Chen, "Research and Application of Full-Text Search Engine Based on Lucene," Computer Technology and Development, Vol. 17, No. 5, 2007, pp. 184-187.

[10] Truls A. Bjorklund and Nils Grimsmo, Johannes Gehrke , Oystein Torbjornsen, Inverted Indexes vs. Bitmap Indexes in Decision Support Systems, ACM, CIKM'09, November 2–6, 2009, Hong Kong, China.

[11] K. Wu, E. J. Otoo, and A. Shoshani. Optimizing bitmap indices with effcient compression, ACM Trans. Database Syst., 2006.