Design and Implementation of Full Text Search Engine Based on Lucene

Na-na ZHANG1,a*, Yi-song WANG1 and Kun ZHU1
1College of Computer Science and Technology, Guizhou University, China
anana_zhang92@163. com
*Corresponding author

Keywords: Lucene, Inverted Index, Index retrieval, Similarity, Logical expression.

Abstract. With the development of information technology, the types and quantities of documents are increasing rapidly. How can people extract information that they need from these documents quickly and accurately, which have already become a very important problem? For this problem, this paper proposed the full text search engine based on Lucene. At first, it achieved the extraction of text content based on the various types of documents, and established indexes for the various types of documents under the folder, then denoted the keyword by the logical expressions. Finally, we used the GfUI (Graphical User Interface) to classify and display the result of retrieval, and verified the processes of establishing index and retrieving through experiments. Experimental results show that Lucene can retrieve the documents quickly and accurately, so meet the needs of users.

Introduction

Nowadays, with the rapid development of information technology, the types and quantities of electronic documents are increasing gradually in study and work. The motivation for a full-text search engine based on Lucene is proposed in this paper: At present, the retrieval of local disk information is mainly using the retrieval method of the operating system itself, and its retrieval efficiency is low, this kind of retrieval method cannot satisfy the retrieval of a large amount of information resources [1]. So how to efficiently retrieve the text content that is a question worthy of further study [2]. The main technology of information retrieval is the full text retrieval technology. Full text retrieval refers to the information retrieval technology based on the contents of various documents [3] (such as plain text, audio, pictures, etc.). Lucene source code is open [4], therefore, it can implement secondary development for different application backgrounds.

Lucene Overview

Lucene was originally a subproject of the Apache Software Foundation Jakarta project team [5], its main function is to easily achieve the retrieval of document content and be able to implement secondary development in the application system. Initially, Lucene was written in the Java language to support multiple operating systems. With the development of computer technology, gradually introduced the Lucene with C, Delphi and other languages written. So far, there are a lot of application projects using Lucene as a full text search engine, the famous are: (1) Eclipse: powerful IDE tool[6] (2) Jive: Web forum system[7](3) Conoon: Web publishing framework based on XML[8] (4) image video retrieval[9].

Most of the search engines use the B-tree structure to maintain the index, and the
update operation of index will lead to a large number of I/O operations, Lucene improved this structure. Lucene continuously build new index files to extend the indexes, and then regularly make the new small index files into the original large indexes, improve the efficiency of the index.

Establish Indexes

Inverted Index Technique

Inverted index is most common storage method for search engine, and is the core content of implementing the search engine. Inverted index determines the visiting record based on the value of the property [10]. Each row in the index table contains an value of attribute and the address of each record that contains the value of attribute. Since the value of attribute determines the position of the record, not the record determines the value of attribute, so called the kind of index method for the inverted index. All in all, each item in the inverted index structure consists of a keyword and a document number, and use the keyword as its primary key. The following example illustrates the inverted index.

For example, there are both A and B documents, their contents are as follows:
A: Its color is red. B: Red is a kind of color.

Table 1 show the results of the general index and Table 2 show the results of the inverted index:

Table 1. General Index.

<table>
<thead>
<tr>
<th>Appeared word</th>
<th>Appeared document</th>
<th>Frequency of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>red</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>red</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>color</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>kind</td>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Inverted Index.

<table>
<thead>
<tr>
<th>Appeared word</th>
<th>Appeared document</th>
<th>Frequency of occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>A,B</td>
<td>1,1</td>
</tr>
<tr>
<td>color</td>
<td>A,B</td>
<td>1,1</td>
</tr>
<tr>
<td>kind</td>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>

As can be seen from Table 1, the quantities of keywords are increasing with the expansion of the text content. The inverted index in Table 2, if it appears the same keyword as above, only modifies the corresponding record information, the quantities of keywords does not increase with the expansion of the text content, and reduces the space occupied by the index file, so improves retrieval efficiency.

Index Retrieval

Index Retrieval Model

Lucene's retrieval model is based on the vector space model [11]. In the vector space model, the document is represented as a K-dimensional vector. It can be expressed as

\[ d_j = (W_{1,j}, W_{2,j}, W_{3,j}, \ldots, W_{k,j}) \]

where \( W_{k,j} \) is the weight of the \( k \)th index item in document
j. Similarly, the query statement is expressed as a vector, \( q = (W_1,q, W_2,q, W_3,q, \ldots, W_k,q) \).

The similarity between the keyword and the document is measured by the cosine of the angle between the vector \( q \) of the keyword and the vector \( d \) of the document. 

\[
\text{Sim}(d_j,q) = \text{Cosine}(d_j,q) = \frac{\sum_{i} W_{q,i} \cdot W_{d,i}}{\sqrt{\sum_{i} (W_{q,i})^2 \cdot \sum_{i} (W_{d,i})^2}} \quad (1)
\]

Combine this formula; we can get the similarity of between the query words and the documents in index library. The result of the formula is greater than 0 and less than 1. The larger the result, the higher the similarity between the document and the keyword, so the query results can be sorted and the documents with higher relevance are ranked in front of the retrieval results.

**Design and Implementation of Lucene Retrieval**

**Application Environment**

The experiment used Lucene toolkit to achieve the full text search under the diskin the MyEclipse development environment. Lucene adopted development kit version of Lucene_30, can be downloaded from Lucene's official website http://lucene.apache.org/, and also required xmlbeans-2.3.0.jar, lucene-core-2.0.0.jar, PDFBox-0.7.3.jar and other required jar packages.

**Establish Indexes**

(1) Create an Index Flow Chart

Flow chart of establishing indexes is shown in Figure 1.

![Flow chart of establishing indexes](image)

Figure1. Flow chart of establishing indexes.

(2) Interface Implementation

The interface of index and retrieval is shown in Figure 2.
Click on "Select the location of index library" and "Select the location of index file", for all documents of the types of ".doc", ".xls", ".pdf", ".ppt", ".html" under D:\Test\files, click on "Establish indexes", the GUI of indexing is shown in Figure 3.

The result shows that all documents of the types of ".doc", ".xls", ".pdf", ".ppt", ".html" under D:\Test\files that have been placed in the index library, and calculates the time for indexing, and gives the documents that cannot be placed in the index library.

The Extraction of Text Content of Multiple Formats. In the disk, storing documents in multiple formats, such as Word, Excel, Powerpoint, PDF and other formats[12], but Lucene itself can only handle plain text format data. In the process of the secondary development of Lucene, it must have function that deal with a variety of common documents. Therefore, the content of the unstructured suffixes ".doc", ".pdf", ".xls", ".ppt", ".html" will be processed. Flow chart of document parsing is shown in Figure 4.
Implementation of Lucene Retrieval

Keyword - Query Logical Expression. There are four main types of query logical expressions:

(1) Wildcards
Keywords contain the symbol "?", "?" represents a single and arbitrary character. For example, if the keyword is "Lucen?", then can search the document that contains "Lucene".

Keywords contain the symbol "*", "*" represents a number of characters. For example, if the keyword is "luce*", then can search the document that contains "Lucene".
Note: You cannot use the "*" or "?" symbol at the beginning of searching.

(2) Fuzzy Query
Add the symbol "~" to the end of the keyword. For example, fuzzy query "lucen ~", you can search the document that contains "Lucene".

(3) Boolean Operator
The expression may contain AND, OR, NOT, "+", and "-" operators.
AND: The AND (&&) operator retrieves a document with coexisting two keywords.
OR: The OR (||) operator is the join operator between the default keywords in Lucene. Retrieve a document that contains any of keywords.
NOT: The NOT (!) operator retrieves a document that does not contain the keyword that after NOT. For example, retrieve documents that contain "open source", but that do not contain "toolkit". The keyword is "open source NOT toolkit".
(Note: The NOT operator must have keywords on both left and right sides. For example, if the keyword is "NOT Toolkit", then it will not retrieve any documents, the keyword "NOT Toolkit" code runs incorrectly)

(4) Combined Query
The keyword contains two or more Boolean operators. For example, Search the document that contains "learn" OR "Lucene", and the document that contains "study", the keyword is "(learn OR Lucene) AND study".

**Implementation of Retrieval.** Interface Implementation

The GUI of retrieval is shown in Figure 5.

![Figure 5. The result of retrieval.](image)

The input keyword is (Lucene AND learn) OR (HTML page), retrieved two documents, D:\Test\files\wendang.doc contains (Lucene AND learn) keyword, D:\Test\files\htmlwangye.html contains (HTML page) keyword.

Copy the document address to the text box, click on the "Open" button, you can open the local file.

**Conclusions**

This paper introduces full text search technology based on Lucene, which extends the types of data sources when establishing indexes, so can establish indexes for multiple format documents. In retrieval, the keyword uses query logical expressions, which makes the retrieval results more accurately, and introduces the analysis of the logical expressions when retrieving the keyword. The next step, we will combine some crawler development search and index optimization work to improve the efficiency of establishing indexes and retrieving.

**Acknowledgement**

This work was supported by the General Program of National Natural Science Foundation of China (NSFC) under Grant No. 60963009.

**References**


