Advances in Internet of Things, 2012, 2, 106-109
http://dx.doi.org/10.4236/ait.2012.24013 Published Online October 2012 (http://www.SciRP.org/journal/ait)

Application of Full Text Search Engine Based on Lucene

Rujia Gao[1], Danying Li[2], Wanlong Li[1], Yaze Dong[3]
[1]Computer Science and Technology, Changchun University of Technology, Changchun, China
[2]Changchun Rural Commercial Bank, Changchun, China
[3]Software Vocational and Technical College, Changchun University of Technology, Changchun, China
Email: gaorujia723@126.com, Shadow_519@126.com, lwl@mail.ccut.edu.cn, DongYaze@mail.edu.cn

Received August 12, 2012; revised September 28, 2012; accepted October 10, 2012

ABSTRACT

This paper introduces us the full-text search engine based on Lucene and full-text retrieval technology, including indexing and system architecture, compares the full-text search of Lucene with the String search retrieval's response time, the experimental results show that the full text search of Lucene has faster retrieval speed.

Keywords: Full Text Search Engine; System Architecture; Lucene

1. Introduction

With the rapid development of Internet and with the explosive growth of Web information, Internet users how to remove the impurities and retained the essence quickly and easily to gain the information they need in the vast ocean of information to become a hot research topic in this field.

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 rather than the external features based on a variety of computer data such text, sound, image as processing object [1]. Create all the possible terms in the index which are searched by network users as well as help people to manage and order extensive information and enable network users to quickly and easily retrieve any information they need. Lucene is a pure Java software project which is mature, free and open source. In recent years, Lucene has become one of the most highly praise and most popular information retrieval library.

2. The Process of Full Text Search

2.1. Build a Text Database

Firstly, we should build a text database which is used to store all information retrieved by the user, then determine text model of retrieval system. The model has an identifiable and low degree of redundancy [2]. Once the model is confirmed, we should not make further more changes any longer.

2.2. Create Indexing

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.

2.3. 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.

2.4. Filter and Sort the Results

After the information retrieval system search the information that the users need and it will filter or sort the information by making a certain rule and then return the user related information [3].

3. Full Text Search Engine of Lucene

Lucene is one of the Jakarta projects of Apache Software Foundation which is an open source full text search engine toolkit, it's not a full text search engine [4], but a full-text search engine 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 API interface so that people can conveniently and quickly develop the search engine.

3.1. Systematic Systematic Structure of Lucene

Lucene is an excellent full text search engine, its structure has a strong object-oriented features. Lucene source

package has seven modules, the five main modules are as follows [5]:

1) *Org.apache.lucene.analysis* Analyzer, Its primary role is to segment the document and remove the stop words which are no help for retrieve but occurrence frequency is very high such as "and" "ah" further separate semantic search words such as Chinese phrase, English words and E-mail address. Lucene can also provide us with two parsers such as SimpleAnalyzer and StandardAnalyzer.

2) *Org.apache.lucene.document* Document Management, Document is similar to a record in relational database, it mainly responsible for the management of fields, and it divided into text field and date field.

3) *Org.apache.lucene.index* Indexing Management, including the establishment of index, inserts records and deletes records. Indexing package is the core of information retrieval system, the purpose of full text search is to adopt the terms which are separated to create index so user can search the information only to those have indexed but not the full text search further greatly improve the efficiency of information retrieval.

4) *Org.apache.lucene.search* Search Management, according to the query to obtain the results of retrieval.

5) *Org.apache.lucene.queryParser* queryParser, parsing the user query and then pass the searcher.

3.2. The Indexing of Lucene

In Lucene, 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. The index process of Lucene is started from the add Document method of IndexWriter, as shown in **Figure 1** [6].

In **Figure 1**, introduce a new class is DocumentWriter. In the API of Lucene, the main role of IndexWriter is to add documents to the indexing which provides us with the main interface for indexing. But writing process of indexing is completed by DocumentWriter. Separating data source and calculating the frequency and location of keywords as well as writing process of indexing is the most complicated thing in Lucene, which are actually occurred in the class of DocumentWriter.

Except for adding documents to indexing, Lucene will go further judge some cases about indexing and then merge indexing [7].

4. Examples of Lucene Retrieval Application

Lucene full text search 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].

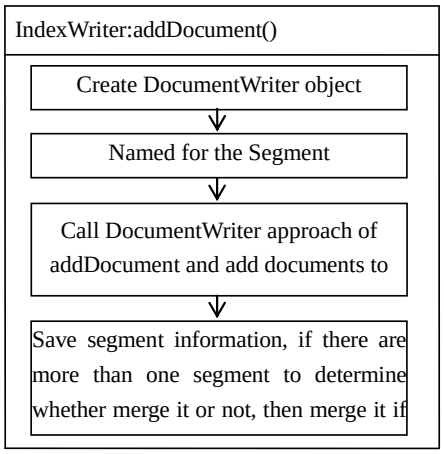

Figure 1. Indexing process of Lucene.

4.1. System Implementation

This paper employed the toolkits of Lucene to simulate two documents retrieval in the Eclipse development environment [9]. Lucene Development Kit version is LUCENE-CORE-2.0.0. JAR and its word tool is JEANALYSIS-1.4.0.JAR, it can also require java runtime environment above JDK1.6 version and need to import JAR package into Eclipse [10].

1) *Preprocessing Module*: *Before* using Lucene 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 Lucene, this paper will divide the large documents into small documents and assign a unique ID number for each document. Main codes are as follows:

```
public class FilePreprocess {
public static void preprocess (File file1, String outputDir)
splitToSmall (characterProcess (file,outputDir + "output.all"), outputDir)
public static File character Process (File file1, String destFile)
private static String replace (String line)
public static void splitToSmall (File file1, String outputpath)
}
```

The replace method is used for storing full-width characters and half-width characters by creating a HashMap and then traverse HashMap, if full-width characters are found we can replace it and finally return replaced characters. The characterProcess method is to convert full-width characters into half-width characters and return new files. The splitToSmall method is to call characterProcess to complete the replacement of full-width characters and half-width characters and finally return new file as splitToSmall method's first parameter, new files named for "output.all" and stored it into outputDir.

Then the splitToSmall method is to divide new files into several small files and stored it into the directory of outputDir.

2) *Indexing Module*: After processing the document, you can use Lucene to process relevant information. Firstly, create indexing for processing documents. Secondly, build query object; Lastly, search in index. At first create a new IndexProcessor class for the document, the main code are as follows [11]:

```
public class IndexProcesser{
private String INDEX_STORE_PATH = "e:\\index1"
public void createIndex (String inputDir)
Directory dir = new SimpleFSDirectory (new File (INDEX_ STORE_PATH))
IndexWriter writer = new IndexWriter (dir, new Standard-Analyzer(),
true,IndexWriter.MaxFieldLength.UNLIMITED)
Document doc = new Document ()
doc.add (new Field ("filename", files [i]. getCanonicalPath(),
Field.Store.YES,Field.Index.NO_NORMS,Field.TermVector.YES))
doc.add(new Field ("content", new FileReader (files [i])))
writer.addDocument (doc)
writer. close()
}
```

First of all, creating IndexWriter object which used StandardAnalyzer as an analysis tool, in order to generate indexing and store it into directory. IndexWriter. MaxField-Length. UNLIMITED shows that IndexWriter creates indexing for fields in the Document and the length of field has no limitations; secondly, creating Documents as well as Fields and adding fields like file name, file contents to Documents; Field.Store.YES indicates that it will store the field of file name; Field. Index. NO_ NORMS shows that it can indexing but not analyze file name; Field.TermVector.YES will store terms of field; FileReader (files [i]) stands for adding values to Fields by using the approach of FileReader; Finally join the document into indexing and use close method to close indexing and write all the data in the cache memory to the disk, close all the data flow. If not closed, the experimental results show that only one segment file in indexing directory.

3) *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 Lucene. However, string-Search approach used java long String to search information. You can also use the delete approach to delete operation on the preprocess text document. Main codes are as follows:

```
public class Search {
//search information with Lucene
public void indexSearch(String searchType,String searchKey)
Directory dir = new SimpleFSDirectory( new File(INDEX _STORE_PATH))
IndexSearcher is = new IndexSearcher(dir)
QueryParser parser = new QueryParser(Version.LUCENE_ CURRENT, searchType, new StandardAnalyzer (Version. LUCENE_CURRENT))
Query query = parser.parse(searchKey)
Date beginTime = new Date ()
is.search(query, collector)
Date endTime = new Date ()
long timeOfSearch = endTime.getTime()-beginTime.getTime()
//search information with String
public void stringSearch(String keyword, String searchDir){
Date beginTime=new Date ()
……
Date endTime=new Date ()
Long timeOffSearch=beginTime.getTime()-endTime.getTime()
public void delete()
IndexReader reader = IndexReader.open (new SimpleFSDirectory ( new File (INDEX_ STORE_PATH )), false)
String a1= "E:\\testfolder\\output3.txt"
Term term=new Term("title",a1)
int a2= reader. deleteDocuments (term)
System. out.println(a1+a2)
reader. deleteDocument(2150)
reader. close()
}
```

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.

4.2. Experimental Results

Based on searching for two documents' keywords to obtain retrieval results by comparing Lucene retrieval with String retrieval [12]. The number of the first document is 250,000 words, the former time is 75 ms, and the latter time is 1988ms; when the number of the document increased to 40 million words, the former time is 108ms, and the latter time is 5688ms. So we can see Lucene's retrieval time-consuming is superior to String retrieval time-consuming. If string retrieval was applied to a large-scale information retrieval system, the search speed will be intolerable when the information storage capacity reaches TB level.

The instance support document indexing and retrieval for the form of txt. If there is a need of practical applications,

you can use PDFBox to process PDF documents and use POI to process Excel and Word and use Jacob to deal with word documents, as long as put data source into a Document object, you can search the information you need.

5. Conclusion

Lucene is a full text indexing engine toolkit written in Java, multi-user support access, quickly visit indexing time and can cross-platform use [13]. This paper in detail analyze the analysis of Lucene, indexing and searching three main modules from system architecture and compare the Lucene full text search with the String retrieval's response time, the experimental results show that the Lucene full text search has faster retrieval speed.

REFERENCES

[1] T. Hong, J. H. Li, P. Zhou and C. S. Gao, "Lucene IN ACTION," Electronics Industry Press, Beijing, 2007.

[2] L. Zhong, H. Wang, R. T. Li and H. Z. Song, "Research and Development of Full Text Search Engine Based on Lucene," *Computer Engineering*, Vol. 33, No. 8, 2007, pp. 282-284.

[3] X. Zhang and Y. Zhou, "Improvement of an Algorithm for Ranking Pages Based on Lucene," *Computer System Application*, Vol. 18, No. 2, 2009, pp. 155-158.

[4] 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.

[5] W. He, S. J. Xue, M. R. Kong, *et al*., "Design and Implementation of Full-Text Search Engine Based on Lucene," *Information Science*, Vol. 3, No. 9, 2006, pp. 88-89.

[6] 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.

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

[8] Q. Zhao, "Information Retrieval," China Machine Press, Beijing, 2008.

[9] C. Dong, "In the Application to Add Full-Text Search Function-Lucene Full-Text Indexing Engine Based on Java, [EB/OL]," 2012. http://www.chedong.com/tech/html

[10] X. W. Lang and S. K. Wang, "Research and Development of Full Text Search Engine Based on Lucene," *Computer Engineering*, Vol. 32, No. 4, 2006, pp. 95-99.

[11] J. C. Cai, Y. P. Guo and L. Wang, "Design and Implementation of School Search Engine Based on Lucene Net," *Computer Technology and Development*, Vol. 16, No. 11, 2006, pp. 73-75.

[12] Z. Qiu, T. T. Fu, "Develop Its Own Search Engine Lucene2.0 + Heritrix," People Post Press, Beijing, 2007

[13] J. Luan and J. F. Li, "Research and Application of Full Text Search Engine Based on Lucene," *Computer & Digital Engineering*, Vol. 38, No. 2, 2010, pp. 184-195.

