Lucene的全文检索的研究与应用

李永春[1],丁华福[2]
(1.哈尔滨理工大学 计算机学院,黑龙江 哈尔滨 150080;
2.哈尔滨工业大学 计算机学院,黑龙江 哈尔滨 150001)

摘 要:为了改善传统全文检索方法在检索效率上的不足,结合Lucene构建了一个全文检索系统模型。介绍了全文检索的基本过程、Lucene源码结构和逻辑结构,分析了Lucene的索引组成,对比了Lucene全文检索和其它全文检索的区别。该模型可用于中小型的全文检索系统的实现,同时可基于此模型开发定制个性化的搜索引擎。最后通过实验对比了其与传统检索方式的响应时间,利用Lucene的全文检索具有更快的响应速度。

关键词:全文检索;Lucene;索引
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2010)02-0012-04

Research and Application of Full Text Search Based on Lucene

LI Yong-chun[1],DING Hua-fu[2]
(1. Computer Academy of Harbin University of Science and Technology,Harbin 150080,China;
2. Computer Academy of Harbin University of Industry,Harbin 150001,China)

Abstract:In order to improve the efficiency in traditional method of retrieval, propose a system model for full text search based on Lucene. First introduced the general process of full-text search, Lucene code structure and logical structure, compared to the differences between Lucene full-text search and other full-text search. This model can be used for small and medium-sized full-text retrieval system and can be used to develop the personalized search engine. Finally, through experiments with the traditional retrieval methods, based on Lucene full-text search has a faster response speed.

Key words:full text retrieval;Lucene;index

0 引 言

随着网络的发展以及数据库技术的成熟,人们已经可以存储大量的信息,如何在海量的信息中快速、准确地进行检索已成为人们越来越关心的问题。

信息检索的核心技术是全文检索技术,全文检索[1]是以各种计算机数据诸如文字、声音、图像等为处理对象,提供按照数据资料的内容而不是外在特征来实现的信息检索手段。在索引中创建一个包含一系列用户搜索条件的查询,它能帮助人们进行大量文档资料的整理和管理工作,并使人们能够快速方便地查到他们想要的任何信息。Lucene是目前最流行开源全文检索工具包,已经在许多搜索项目中得到了广泛的应用。

收稿日期:2009-05-10;**修回日期**:2009-08-15
基金项目:国家自然科学基金资助项目(60736014)
作者简介:李永春(1985-),男,硕士研究生,研究方向为数据挖掘、信息检索;丁华福,教授,硕士生导师,研究方向为自然语言处理、数据挖掘。

文中将介绍Lucene和全文检索的过程,并利用Lucene实现一个小型的检索系统,然后通过实验对利用Lucene和传统关键字查找的效率进行比较。

1 全文检索的过程

全文检索的过程如下:

(1)首先构建一个文本库,这个文本库用来保存用户可能检索的信息,在这些信息的基础上确定检索系统中的文本模型。文本模型就是被系统认可的一种信息格式,一旦确定之后,不应对其再进行大的变动。

(2)建立索引,索引可以大大提高信息检索的速度。采用何种方式取决于检索系统的规模,大型的检索系统通常采用倒排的方式来建立索引。

(3)索引建立之后,就可以开始进行搜索。通常由用户提交请求,请求分析后,然后利用文本操作进行处理。

(4)对结果进行过滤和排序,再将过滤排序后的结果返回给用户。

2 Lucene 概述

Lucene 是一个用 Java 写的全文检索引擎工具包,可以方便地嵌入到各种应用中实现针对应用的全文索引/检索功能,Lucene 有两个主要的服务,索引和检索,两者任务是相互独立的。这使得开发人员可以根据需要对它们进行扩展。Lucene 提供了丰富的 API,可以与存储在索引中的信息方便的交互。需要说明的是它并不是一个完整的全文检索应用,而是为应用程序提供索引和搜索功能。即若想让 Lucene 真正起作用,还需在其基础上做一些必要的二次开发。

Lucene 源码中共包括 7 个子包,每个包完成特定的功能[2],具体如表 1 所示:

表 1　Lucene 源码包对应功能表

包名	功能
Org. apache. lucene. analysis	语言分析器,主要用于切分词
Org. apache. lucene. document	索引存储时的文档结构管理
Org. apache. lucene. index	索引管理,包括索引建立、删除等
Org. apache. lucene. queryParser	查询分析器,实现查询关键词间的运算
Org. apache. lucene. search	检索管理,根据查询条件,检索得到结果
Org. apache. lucene. store	数据存储管理
Org. apache. lucene. util	公共类

Lucene 功能非常强大,但从根本上说,主要包括两块:一是文本内容经切分词后索引入库;二是根据查询条件返回结果,即索引部分和查询部分两部分。下面结合上述的源码包给出 Lucene 的逻辑结构图[3]。如图 1 所示,在图中可以看源码包所对应完成的功能属于索引部分还是查询部分。

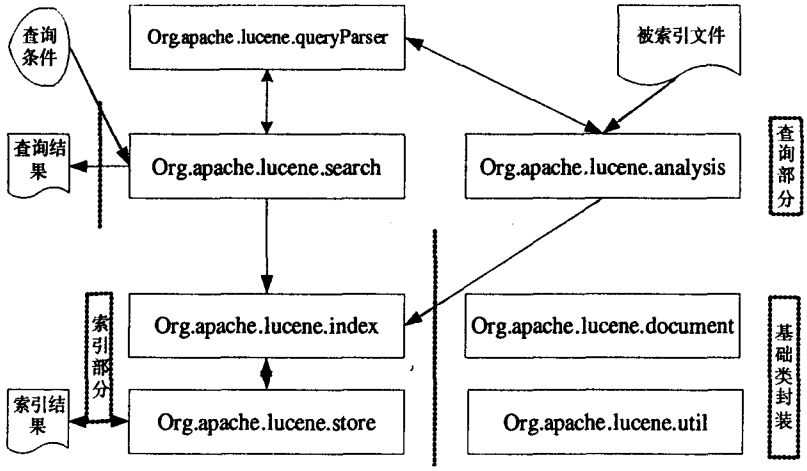

图 1　Lucene 逻辑结构图

2.1 Lucene 的索引

Lucene 索引信息存储有三种可选方式:内存(RAM),文件系统(FS)和数据库(DB)。RAM 存储适用于较小的检索系统,文件系统存储可用于中型检索,数据库存储则适用于对检索性能要求更高的系统。下面对其索引存储逻辑进行分析。

在 Lucene 中,索引(index)由段(segment)组成,段(segment)由记录(document)组成,记录(document)由域(field)组成,域(field)由字符串(term)组成[4]。例如,一个 document 可以与一个物理文件对应,将其中文件名、文件内容、文件创建时间等信息提取为数据源放入 document 中,也可将多个物理文件的数据源同时放入一个 document。数据源是由一个被称为 field 的类表示。这个 field 类主要用来标识当前数据源各种属性[5,6],主要是以下三种标识:

可分词性:该数据源是否需要经过分词。可分词时,内容被分成多个可被索引的词;不可分词时,整个字段内容作为一个词。

可存储性:字段内容直接按照词存放,而不是以倒排形式存放。

可索引性:该数据源的数据是否要在用户检索时被检索,需要检索时,字段内容以倒排形式存放,即记录字段每个词在某一文档中出现的频率。

从代码方面上看,在 Lucene 的索引部分,invertDocument()方法是最重要的,由它调用分析器的接口,分析统计词条的位置与频率信息。作用就是建立倒排索引,直观的讲,就像一本书在最后给出在书中出现的名词列表,同时对应给出该名词出现在第几章和第几页。在查找的时候,可以直接定位到具体页码,而不用从目录开始逐个查找。

Lucene 在维护和扩展索引的时候不断创建新的索引文件,最终将这些新的小索引文件并入大索引中。使程序员可以根据不同的要求自行调整批次大小、周期长短。这点对于 Lucene 的检索效率也相当的重要。建立索引,是需要占用内存资源的,当有新的记录加入索引时,并不直接写入硬盘而是先放在内存中,所以最直接提高检索速度的方法就是提高内存存放索引的缓冲区的大小。具体缓冲区的大小设置需要根据实际情况而定。

2.2 Lucene 和其它全文检索的区别

传统的查找技术是通过逐次匹配内存中的文本实现的,即顺序查找。对文档集合中的信息进行少量预处理或不做处理,这种方法只适合文档较少的情况,虽然结构简单,易实现,但检索速度比较慢,尤其在处理海量数据和模糊查询时有着明显不足。当信息量在 TB 级别时,查找的速度是无法忍受的。Lucene 通过特殊的索

引结构实现了传统检索不擅长的全文索引机制，这也是 Lucene 快速发展的原因之一。表 2 对比了 Lucene 和其它全文检索的区别[7]。

表 2 Lucene 和其它全文检索的区别

	Lucene 全文检索	其它全文检索系统
索引方式	可以进行增量索引，也可以进行批量索引	只支持批量索引，哪怕数据量只有一点变化
索引内容	Lucene 的文档是由多个字段组成，可以控制检索哪个字段，甚至不检索哪个字段。进一步需要索引的字段可以确定哪些需要分词。需要分词的字段比如标题、内容，不需要分词如作者	往往检索了整个文档
语言分析	支持非英文语言，能够过滤不必要的词，比如“的”字，能够进行语法分析，比如会将 jumpedjumping 都当作 jump 处理	没有通用接口
查询分析	可自定制查询语法规则，自定义关键字的 +，-，与，或关系	

3 Lucene 检索应用实例

对于中文全文检索应用来说，分词模块（analysis）、索引模块（index）和检索模块（search）是主要的三个模块。分词模块负责对信息源的预处理工作，这个过程对于用户来说是透明的。索引模块是为了提高检索的响应速度。检索模块是和用户交互的主要模块。下面的实例将分别实现这三个模块功能，以完成一个检索应用的实例。

3.1 系统实现

文中利用 Lucene 工具包，在 Eclipse 开发环境下模拟一个全文检索环境。Lucene 开发包版本为 lucene-core-2.4，分词工具包采用 je-analysis-1.5.jar，同时需要 JDK1.5 版本以上 java 运行环境。开发时 Eclipse 中导入上述两个 jar 包。

首先对文档进行预处理，将文档处理成一种相对原始的状态，以方便分词工具的处理。这种处理是有必要的，尤其是在大型的搜索项目中，如果缺乏必要的处理，可能导致分词不准确，从而导致影响用户搜索的结果。同时也有必要对一些无意义但出现频率很高的字词如“了”、“啊”、“的”等此类信息进行处理。在这里为了模拟全文检索，将大文件通过程序分割成若干的小文件，简化为对若干文本文档进行处理。在复杂的检索中，文档是不同类型和不同格式的，前期的预处理也比较复杂。在这里简化并不影响 Lucene 是如何建立索引的理解。

预处理过程代码结构：

```
public class FilePreprocess {
  public static void preprocess(File file, String outputDir)
  public static File charactorProcess(File file, String destFile)//字符处理
  public static void splitToSmallFiles(File file, String outputpath)//文档切分
  private static String replace(String line)}
```

程序执行过程中依次调用上述三个函数，首先进行入 preprocess()函数，进行入后调用字符处理 charactorProcess()函数，最后执行切分程序，完成对文档的预处理工作。

预处理完成后，进行分词并创建文档索引，在分词过程中，尤其是对中文进行分词，Lucene 主要使用的是二元切分法[8]。因为英文可以根据单词间的自然空格进行切分，而中文不行。举例说明二元分词，如“浙江杭州西湖”，切分之后就是“浙江、江杭、杭州、州西、西湖”。采用二元分词是因为二元是最小的查询单位。如果采用单个字符，那么“上海”被分成“上、海”，结果很可能将“海上”一起查了出来。多元分词可能会降低分词精度。二元切分是中文分词的普遍方法。Lucene-core.jar 对中文分词的支持并不理想，所以加入 je-analysis.jar 处理中文。

索引的建立在 Lucene 中比较简单，在函数中 createIndex()传入被建立索引的文件的位置即可，关键代码如下：

```
public class IndexProcess {
  public void createIndex(String inputDir) throw Exceptions {
    IndexWriter iw = new IndexWriter(INDEX-STORE, new MMAnalyzer(),true){ Document doc = new Document(...);
Field field = new Field(...);
doc.add(field);
iw.addDocument(doc);.....}}
```

首先创建一个 IndexWriter 对象，用来生成索引，然后创建 Document 记录，Field 域，将 field 添加至 doc 记录中，最后把 doc 记录加入至 IndexWriter 对象里面。上面的程序中省略了一些设定是否分词或存储的参数在前文中均有提及。

索引建立之后，再提供一个搜索功能的类。在这个类中，提供两个方法：一个是利用 Lucene 建立索引进行检索，另一个利用传统的字符匹配进行检索。并计算两种方法各自的耗时，以比较两种方式的检索效率。

```
public class Search {
private String INDEX_STORE = "E:\\index";
public void indexSearch(String searchType,String searchKey){
IndexSearcher is = new IndexSearcher(INDEX-STORE);.....}
```

```
public void StringSearch(String keyword, String searchDir)
{.....}
    Date endTime = new Date();
    long timeOfSearch = endTime.getTime() - beginTime.
getTime();}
```

先给出索引的路径,然后再提供要搜索的 Field 和搜索的关键字作为参数,最后生成一个查询对象即可开始查询。

通过这个实例,可以了解检索的一般流程,其它功能的复杂的检索只是对具体方法的细化和完善,但都会有上面的三个部分。

下面对实验结果做简要说明。

3.2 实验结果说明

通过对两个文档的检索,来对比应用 Lucene 进行检索和传统检索方式的效率。第一篇文档字数为 50 万左右,检索目标词汇,前者用时 120ms,后者用时 389ms,大约是前者的三倍。当文档字数增加到 2500 万字左右时,即用第二篇测试文档进行,采用 Lucene 的系统用时大约 300ms,而传统检索方法用时接近 4000ms。采用 Lucene 的检索明显优于字符串匹配方式。由此可以想象当数据量达到 TB 级别时,传统检索方式的检索速度将是无法忍受的。

以上的测试文档均为 txt 格式,其实 Lucene 并没有规定数据源的格式,它提供了一个通用的结构来接受索引的输入,因此输入的数据源可以是数据库、Word 文档、PDF 文档、HTML 文档,只要能够设计相应的解析转换器将数据源构造成 Document 对象即可进行索引。

该实例默认支持 txt 形式文档的索引和检索。如因实际应用需要,还可以利用第三方工具如 POI,XPDF 和 Jacob 包对 Excel,Word 和 PDF 进行操作,只要将数据源构造成一个 Document 对象,就可以实现对其的检索支持。

4 结束语

文中讲述了全文检索技术,描述了 Lucene 的结构和特点,提出了一种解决中文全文检索的方法,可以应用在中小企业网站站内检索,个人用户桌面搜索引擎建立,特定文档检索数据库建立等方面。Lucene 所独有的增量索引技术使其能够被大部分的项目成功应用,相比传统的检索方式在效率上有很大的改善,同时用户可以根据自己的需要构建个性化的搜索,从而摆脱对 Google,baidu 站内搜索的依赖。实现对目标文档检索和管理,降低中小网站的运营成本。个人如果有兴趣,还可以结合一些开源的爬虫开发自己的搜索引擎,这也是下一步将开展的工作。

参考文献:

[1] 孙西全,马瑞芳,李燕灵.基于 Lucene 的信息检索的研究与应用[J].情报理论与实践,2006,29(1):521-528.

[2] Gospodnetic O,Hatcher E. Lucene in action[M].[s.l.]:Manning Publications Co,2005.

[3] 林碧英,赵　锐,陈良臣.基于 Lucene 的全文检索引擎研究与应用[J].计算机技术与发展,2007,17(5):186-190.

[4] 索红光,孙　鑫.基于 Lucene 的中文全文检索系统的研究与设计[J].计算机工程与设计,2008,29(19):5083-5086.

[5] 邱　哲,符滔滔.开发自己的搜索引擎 Lucene2.0+Hertrix[M].北京:人民邮电出版社,2007.

[6] 朱学昊,王儒敬,余锋林,等.基于 Lucene 的站内搜索设计与实现[J].计算机应用与软件,2008,25(10):6-8.

[7] 郎小伟,王申康.基于 Lucene 的全文检索系统研究与开发[J].计算机工程,2006,32(4):95-99.

[8] Zhang Yuletide,Zhang Tao,Chen Shijie. Research on Lucene-based English-Chinese cross-language information retrieval[J]. Journal of Chinese Language and Computing,2005,15(1):25-32.

(上接第 11 页)

brid Kernel and Minimal Vapnik-Chervonenkis Dimension [J]. IEEE Transactions on Knowledge and Data Engineering, 2004,16(4):385-395.

[6] Czajkowska J,Rudzki M,Czajkowski Z. A New Fuzzy Support Vectors Machine for Biomedical Data Classification[C]//30th Annual International IEEE EMBS Conference. Vancouver, British Columbia, Canada:[s.n.],2008:4476-4479.

[7] TSANG E C C,YEUNG D S,CHAN P P K. Fuzzy Support Vector Machines for Solving Two-class problems[C]//Proceedings of the Second International Conference on Machine Learning and Cybernetics. Xi'an:[s.n.],2003:1080-1083.

[8] Li Xuehua,Shu Lan. Fuzzy Theory Based on Support Vector Machine Classifier[C]//Fifth International Conference on Fuzzy Systems and Knowledge Discovery. Jinan, China:[s.n.],2008:600-604.

[9] Lin C F,Wang S D. Fuzzy Support Vector Machines[J]. IEEE Transactions on Neural Networks,2002,13(12):466-471.

[10] Soria E, Martin J, Camps G, et al. A low complexity fuzzy activation function for artificial neural networks[J]. IEEE Trans Neural Networks,2003,14(6):1576-1579.