广西科学院学报
Journal of Guangxi Academy of Sciences

2010,26(4):433～435,442
Vol. 26, No. 4 November 2010

Lucene 全文检索引擎的结构机制与应用方式
Lucene Structural Mechanism for Full-text Retrieval Engine and Application Mode

李明宙,罗 艳,王宗义
LI Ming-zhou, LUO Yan, WANG Zhong-yi

(南宁海蓝数据有限公司,广西南宁 530022)
(Highland Digital Technology INC., Nanning, Guangxi, 530022, China)

摘要:介绍开放源代码的全文检索引擎工具包 Lucene 的系统结构和实现机制,分析 Lucene 的组织结构及全文检索的分词的相关方法后,给出 Lunene 在全文检索领域中的应用方式。

关键词:全文检索 Lucene 索引 搜索
中图法分类号:TP391.3 **文献标识码**:A **文章编号**:1002-7378(2010)04-0433-03

Abstract: The system structure and mechanism of the Lucene, a open source full-text retrieval software library are introduced. Lucene's organizational structure and related methods of full-text retrieval's segmentation are analyzed. Then Lucene's application mode in the field of full-text retrieval is presented.

Key words: full text retrieval, Lucene, indexing, search

全文检索(Full-text Search)是计算机索引程序通过扫描文件中的每一个词,对每一个词建立一个索引,指明该词在文章中出现的次数和位置,当用户查询时,检索程序就根据事先建立的索引进行查找,并将查找的结果反馈给用户的检索方式[1]。全文检索大体分两个过程:索引创建(Indexing)和搜索索引(Search)。索引创建是将现实世界中所有的结构化和非结构化数据提取信息创建索引的过程,搜索索引是得到用户的查询请求后搜索创建的索引再返回结果的过程。全文检索引擎 Lucene 是一个开放源代码的全文检索引擎工具包,使用它可以方便的嵌入到各种应用中实现针对应用的全文索引和检索功能。本文介绍 Lucene 的系统结构和实现机制,分析 Lucene 的组织结构及全文检索的分词的相关方法,给出 Lunene 在全文检索领域中的应用方式。

1 Lucene 的系统结构和实现机制

全文检索引擎 Lucene 的系统结构运用了大量的面向对象的设计思想。首先是定义了一个与平台无关的索引文件格式,其次通过抽象系统的核心组成部分设计为抽象类,具体的平台实现部分设计为抽象类的实现,此外与具体平台相关的部分,比如文件存储也封装为类,经过层层的面向对象式的处理,最终达成了一个低耦合高效率,容易二次开发的检索引擎系统。

从图 1 中可以清楚看到,Lucene 系统由基础结构封装、索引核心、对外接口三大部分组成,其中直接操作索引文件的索引核心又是系统的重点。索引的最后结果就是产生索引文件,这些索引文件构成索引库[2]。

Lucene 在实现中,不是维护一个索引文件,而是在扩展索引的时候不断创建新的索引文件,然后定期的把这些新的小索引文件合并到原先的大索引中(针对不同的更新策略,批次的大小可以调整),这样在不影响检索的效率的前提下,提高了索引的效率。

为了实现快速的搜索,Lucene 会首先将需要处理的数据以一种称为倒排索引的数据结构进行存储。Lucene 会为需要被搜索的数据整理优化出一份索引文件(Index file),而这个过程称之为"索引"。Lucene 面向全文检索的优化在于首次索引检索后,将所有结果中匹配度最高的头 100 条结果

收稿日期:2010-08-15
修回日期:2010-10-12
作者简介:李明宙(1979-),男,助理工程师,主要从事数字档案管理技术研发工作。

(TopDocs)的ID放到结果集缓存中并返回,即使检索匹配总数很多,Lucene的结果集占用的内存空间也不会很多。如果首批缓存结果数用完后还要读取更后面的结果时Searcher会再次检索并生成一个上次的搜索缓存数大1倍的缓存,并再重新向后抓取[3]。

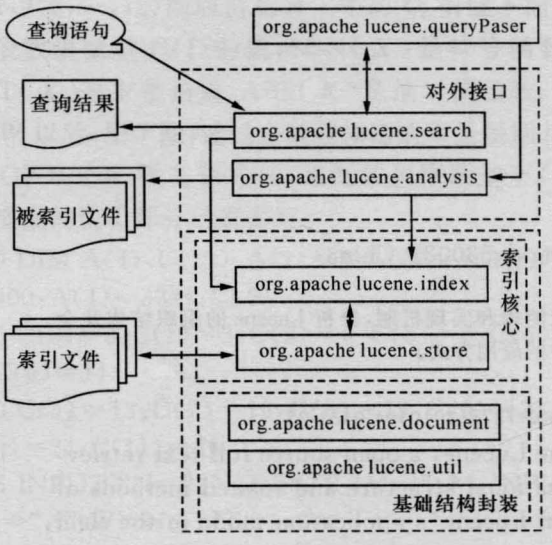

图1 Lucene系统结构

2 Lucene的应用方式

Lucene是一个高效的,可扩展的全文检索库,全部用Java实现,无须配置。Lucene仅支持纯文本文件的索引和搜索,不负责由其他格式的文件抽取纯文本文件,或从网络中抓取文件的过程。Lucene的索引是应用反向索引,被索引的文档用Document对象表示,当用户有请求时,Query代表用户的查询语句,IndexWriter通过函数addDocument将文档添加到索引中,实现创建索引的过程(图2)。Lucene的IndexSearcher通过函数search搜索Lucene Index,IndexSearcher计算term weight和score并且将结果返回给用户,返回给用户的文档集合用TopDocsCollector表示(图2)。

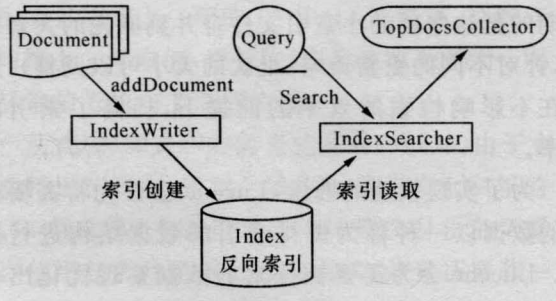

图2 Lucene组件

2.1 调用Lucene API实现索引和搜索过程[4]

Lucene API函数的简单调用进入Lucene的源代码后,Lucene会有很多包,关系错综复杂。然而通过图3不难发现,Lucene的各源码模块,都是对普通索引和搜索过程的一种实现。

索引过程:(1)创建一个IndexWriter用来写索引文件,它有几个参数,INDEX_DIR就是索引文件所存放的位置,Analyzer便是用来对文档进行词法分析和语言处理的。(2)创建一个Document代表要索引的文档。(3)将不同的Field加入到文档中。一篇文档中,题目、作者、修改时间、内容等不同类型的信息用不同的Field来表示,在本例子中,一共有两类信息进行了索引,一个是文件路径,一个是文件内容。其中FileReader的SRC_FILE就表示要索引的源文件。(4)IndexWriter调用函数addDocument将索引写到索引文件夹中。

搜索过程:(1)IndexReader将磁盘上的索引信息读入到内存,INDEX_DIR就是索引文件存放的位置。(2)创建IndexSearcher准备进行搜索。(3)创建Analyer用来对查询语句进行词法分析和语言处理。(4)创建QueryParser用来对查询语句进行语法分析。(5)QueryParser调用parser进行语法分析,形成查询语法树,放到Query中。(6)IndexSearcher调用search对查询语法树Query进行搜索,得到结果TopScoreDocCollector。

2.2 Lucene的索引和搜索过程

Lucene索引过程:(1)创建IndexWriter对象。创建IndexWriter对象的代码为:IndexWriter writer=new IndexWriter(FSDirectory.open(INDEX_DIR), new StandardAnalyzer(Version.LUCENE_CURRENT), true, IndexWriter.MaxFieldLength.LIMITED)。IndexWriter对象主要包含:用于索引文档,合并段并在合并段的文章中详细描述,保持索引完整性、一致性和事务性,以及一些配置方面的信息。(2)创建文档Document对象,并加入域(Field)。创建文档Document对象的代码:Document doc = new Document();doc.add(new Field("path", f.getPath(), Field.Store.YES, Field.Index.NOT_ANALYZED));doc.add(new Field("modified", DateTools.timeToString(f.lastModified(), DateTools.Resolution.MINUTE), Field.Store.YES, Field.Index.NOT_ANALYZED));doc.add(new Field("contents", new FileReader(f)))。Document对象的主要部分包括:文档的boost默认为1,大于1说明比一般的文档更加重要,小于1说明更不重要;1个

ArrayList 保存文档所有的域;每 1 个域包括域名、域值和一些标志位,和 fnm、fdx、fdt 中的描述相对应。(3)将文档加入 IndexWriter。代码:writer. addDocument(doc); ——> IndexWriter. addDocument(Document doc, Analyzer analyzer) ;——> doFlush = docWriter. addDocument (doc, analyzer); ——> DocumentsWriter. updateDocument (Document, Analyzer, Term) 。代码中的"——>"代表一级函数调用 。IndexWriter 继而调用 DocumentsWriter. addDocument,其又调用 DocumentsWriter. updateDocument。(4)将文档加入 DocumentsWriter。代码:DocumentsWriter. updateDocument (Document doc, Analyzer analyzer, Term delTerm); ——> ① DocumentsWriterThreadState state = getThreadState (doc, delTerm); ——>② DocWriter perDoc = state. consumer. processDocument(); ——>③ finishDocument(state, perDoc)。DocumentsWriter 对象主要用于写索引文件、删除文档,和缓存管理。(5)关闭 IndexWriter 对象。代码:writer. close(); ——> IndexWriter. closeInternal(boolean); ——>①将索引信息由内存写入磁盘: flush(waitForMerges, true, true); ——>② 进行段合并: mergeScheduler. merge(this)。

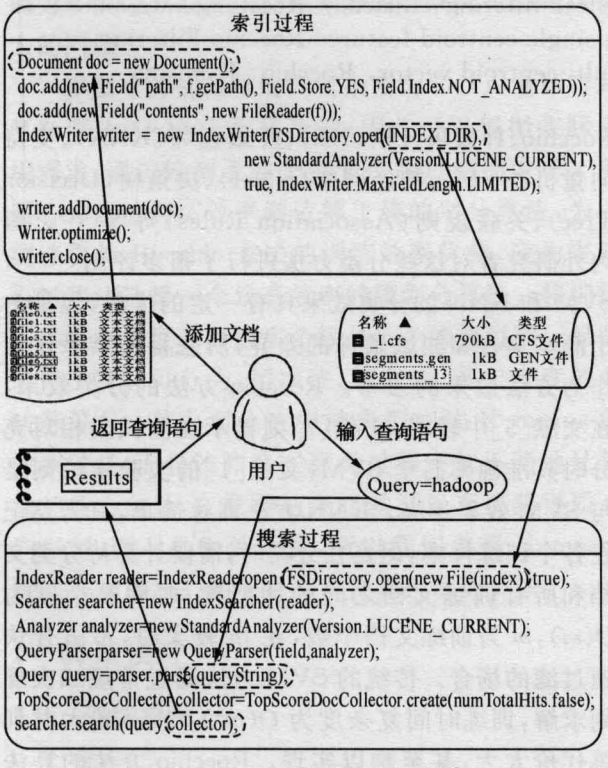

图 3 Lucene 实现的包结构[5]

Lucene 搜索过程:(1)打 IndexReader 指向索引文件夹。代码为:IndexReader reader = IndexReader. open (FSDirectory. open (indexDir))。其实是调用 DirectoryReader. open (Directory, IndexDeletionPolicy, IndexCommit, boolean, int) 函数,其主要作用是生成一个 SegmentInfos. FindSegmentsFile 对象,并用它来找到此索引文件中所有的段,并打开这些段。(2)打开 IndexSearcher。代码为:IndexSearcher searcher = new IndexSearcher(reader)。IndexSearcher 表面上看起来好像仅仅是 reader 的一个封装,它的很多函数都是直接调用 reader 的相应函数,如:int docFreq(Term term), Document doc(int i), int maxDoc();然而它提供了两个非常重要的函数。(3) QueryParser 解析查询语句生成查询对象。代码为:QueryParser parser = new QueryParser(Version. LUCENE _ CURRENT, "contents", new tandardAnalyzer(Version.

LUCENE _ CURRENT)); Query query = parser. parse("+(+apple * -boy) (cat * dog) -(eat~ foods)")。根据查询语句生成的是 1 个 Query 树,这个 Query 树很重要,并且会生成其他的树,一直贯穿整个索引过程。(4)搜索查询对象。代码: TopDocs docs = searcher. search(query, 50);其最终调用 search(createWeight(query), filter, n)。

3 结束语

相对于其它已经存在的商业全文检索引擎,Lucene 具有相当的优势。首先,它的开发源代码发行方式遵守 Apace Software License,不仅可以充分利用 Lucene 所提供的强大功能,而且可以深入细致地学习到全文检索引擎制作技术和面相对象编程的实践,进而在此基础上根据应用的实际情况编写出更好的更适合当前应用的全文检索引擎。在这一点上,商业软件的灵活性远远不及 Lucene。其次,Lucene 秉承了开放源代码一贯的架构优良的优势,设计了一个合理而极具扩充能力的面向对象架构,可以相对容易地在 Lucene 的基础上扩充各种功能,譬如中文全文检索功能。

Lucene 合适中小型项目构建全文检索系统,目前我们已经将 Lucene 应用在数字档案馆检索系统中。Lucene 全部用 Java 实现,无须配置,既高效又可扩展。相信随着研究的不断深入,Lucene 在中文全文检索领域应用的前景也将会越来越广阔。

(下转第 442 页)

著，但是越译汉的效果还需要进一步提高。下一步我们将通过对高质量的平行语料库中抽取人名对训练、学习来扩大汉越音译知识，从而提高音译的整体效果。另外，本文的音译方法不仅可以用于汉越人名的翻译，还可以应用于汉越机器翻译和汉越双语命名实体对的抽取。

参考文献：

[1] 王蕾. 基于字形的英汉机器音译改进研究[D]. 哈尔滨工业大学，2007.

[2] 周美玲. 英汉人名音译方法的研究与实现[D]. 苏州大学，2009.

[3] 庞薇，徐波. 基于多模式融合的人名翻译系统[J]. 中文信息学报，2009，23(1)：44-49.

[4] 邹波，赵军. 英汉人名音译方法研究[C]. 第四届全国学生计算语言学研讨会会议论文集，2008：232-238.

[5] 艾山·吾买尔，吐尔根·伊布拉音. 英文维文人名机器翻译算法的研究与实现[J]. 新疆大学学报：自然科学版，2007，24(1)：97-101.

[6] 吴妙玲. 越语人名与汉语人名的对比与翻译问题[D]. 桂林：广西师范大学，2008.

[7] 雷航. 现代越汉词典[M]. 北京：外语教学与研究出版社，1998.

[8] Phạm Văn Hải，Lê Văn Dặng. Chữ Hán và Tiếng Hán-Việt[EB/OL]. http://www.viethoc.org/eholdings/PhamVanHai/ChuHanvaTiengHanViet.pdf. 2005.

[9] 梁远. 实用越汉分类词典[M]. 北京：民族出版社，2007.

（责任编辑：邓大玉）

（上接第435页）

参考文献：

[1] Ricardo Baeza-Yates，Berthier Ribeiro-Neto，王知津，等. 现代信息检索[M]. 北京：机械工业出版社，2005.

[2] 何伟，薛素静，孔梦荣，等. 基于Lucene的全文搜索引擎的设计与实现[J]. 情报杂志，2006(9)：88-89.

[3] 管建和，甘剑峰. 基于I.，ueene全文检索引擎的应用研究与实现[J]. 计算机工程与设计，2007(2)：490-491.

[4] 赵汀，孟祥武. 基于Lucene API的中文全文数据库的设计与实现[J]. 计算机工程与应用，2003，20：179-183.

[5] 葛帅. 开方源代码的全文检索引擎[EB/OL]. http://www.lucene.com.cn/about.htm 2006.09.

（责任编辑：邓大玉）