第 33 卷 第 18 期 计 算 机 工 程 2007 年 9 月
Vol.33 No.18 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)18—0095—02 文献标识码：A 中图分类号：TP391

Lucene 搜索引擎

周登朋，谢康林
(上海交通大学计算机科学与工程系，上海 200240)

摘 要：Lucene 是一个高性能、易扩展的基于 Java 技术的全文信息检索工具包，它能非常方便地为各种应用程序加入全文索引和搜索功能。该文探讨了 Lucene 中使用的向量空间模型，分析了 Lucene 索引文件的结构以及搜索排序算法，讨论了 Lucene 的压缩算法并且通过实验验证了 Lucene 的建立索引的过程。

关键词：Lucene；向量空间模型；排序算法；信息检索

Lucene Search Engine

ZHOU Deng-peng, XIE Kang-lin
(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 As an information retrieval library written in Java, Lucene, with high performance and easy to scale, can easily add searching and indexing capabilities to applications. This paper discusses the vector space model used in Lucene, analyzes the structure of index files and ranking algorithm, and describes the compressing algorithm in Lucene. An experiment is done to test the indexing process of Lucene.

【Key words】 Lucene; vector space model; ranking algorithm; information retrieval

1 向量空间模型

在向量空间模型中，文档被表示成一个 K 维(K-Dimensional)向量。K 的大小是所有被索引的文档中的索引项(Term)的个数，通常情况下，每个索引项就是文档中的一个单词或者短语。这样，K 维向量中的每一项的值，就是该索引项在这个文档中的权重。权重一般情况下是个大于或者等于 0 的值，如果该索引项在该文档中不存在，则权重为 0，否则权重被赋予一个大于 0 的值。按照这个定义，每个文档可以表示为

$$\boldsymbol{d}_j=(w_{1,j},w_{2,j},\cdots,w_{k,j})$$

其中，$w_{k,j}$ 表示第 k 个索引项在文档 j 中的权重。同理，每个查询语句也可以表示成向量 $\boldsymbol{q}=(w_{1q},w_{2q},\cdots,w_{kq})$。将文档和用户的查询语句都表示成向量之后，就可以利用文档向量和查询向量之间的相似性来表示文档和查询之间的相关性。为了衡量文档向量和查询向量之间的相似性，一种比较常用的方法是使用余弦系数(cosine coefficient)表示为

$$Sim(\boldsymbol{d}_j,\boldsymbol{q})=Cosine(\boldsymbol{d}_j,\boldsymbol{q})$$

$$\frac{\boldsymbol{d}_j\bullet\boldsymbol{q}}{|\boldsymbol{d}_j|\times|\boldsymbol{q}|}=\frac{\sum_{i=1}^{t}w_{i,j}\times w_{i,q}}{\sqrt{\sum_{i=1}^{t}w_{i,j}^2}\times\sqrt{\sum_{j=1}^{t}w_{i,q}^2}}$$

该算法用两个向量的夹角的余弦值来表示两个文档之间或者文档与查询向量之间的相似程度。利用该算法，向量空间模型就能够计算出系统中的每个文档向量和查询向量的相似程度。计算结果的数值在 0 到 1 之间，值越大说明相似程度越高，也说明这个文档和查询之间的相关性越高，这样就可以对查询结果进行排序，把相关性高的搜索结果排在前面。

2 Lucene 的索引机制

2.1 反向索引

为了实现快速检索，Lucene 采用的是反向索引(inverted index)机制，反向索引是相对前向索引(forward index)而言的。在前向索引中，每个文档对应着一个该文档所包含的词或者短语的列表；在反向索引中，文档中的每个词或者短语对应着一个文档列表，该列表中的文档是所有包含这个词或者短语的文档。该列表还会包含一些辅助信息，比如该词或者短语在文档中出现的次数以及出现的位置等，这些信息会被用来对搜索结果进行排序。这种结构对于“哪些文档中包含单词 X”这样的问题能够快速得到搜索结果。例如，典型的反向列表可表示为 $t_i\rightarrow\langle d_1,\ldots\rangle,\langle d_2,\ldots\rangle,\ldots,\langle d_n,\ldots\rangle$。

2.2 索引文件结构

为了实现高效的索引和检索，就必须具有良好的索引文件结构。Lucene 的索引文件包括逻辑结构和物理结构。Lucene 的每个索引文件都由一个或者多个片段(segment)组成；每个片段都是一个可以被独立检索的模块，包含一定数量的文档(document)，这里的文档可以是一个 HTML 页面，一个 XML 文档，或一个 Word 文档。Lucene 的索引文件的逻辑结构如图 1 所示。

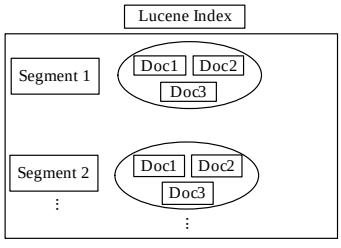

图 1 Lucene 索引文件的逻辑结构

Lucene 索引文件中的核心文件包括：

(1)索引项信息文件(term information file)。该文件存储了

作者简介：周登朋(1983－)，男，硕士研究生，主研方向：数据挖掘，信息检索；谢康林，博士生导师

收稿日期：2006-09-29 **E-mail**：zhoudengpeng@yahoo.com.cn

Lucene 中的所有的索引项以及相关信息，主要由以下字段组成：

1)字段 Term Count 记录了该索引文件中的索引项的个数；

2)字段 Document Frequency 记录了有多少文档包含该索引项；

3)字段 Term 记录了每个索引项的内容；

4)字段 Frequency Data 指向索引项频率文件(Term Frequency File)；

5)字段 Position Data 指向索引项位置文件(Term Position File)；

(2)索引项频率文件。该文件记录了每个索引项(term)在各个文档(document)中的出现频率。这些信息为对搜索结构进行排序提供了重要信息。

(3)索引项位置文件。该文件记录了每个索引项在各个文档中出现的位置信息，这些信息对实现精确查找(exact phrase search)是必需的。

2.3 索引的建立

Lucene 为文档建立索引的过程，对文档格式没有要求，只要能从这些文件中提取出文本信息即可。以 HTML 文件为例，其索引建立过程是：(1)通过一个 HTML Parser 对 HTML 文档进行解析，过滤掉其中的 HTML Tag，提取出重要信息，比如 Title 等；(2)将这些信息传送给 Lucene Analyzer，Lucene Analyzer 把这些内容转换成单独的索引项并提取出相关信息，存储到索引文件中。该过程如图 2 所示。

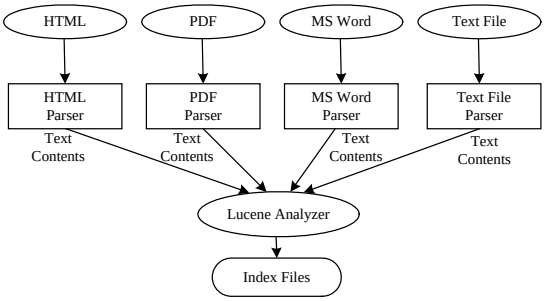

图 2 Lucene 的索引过程

3 Lucene 的评分函数

评分函数是搜索引擎的核心部分，其好坏将直接影响搜索结果。Lucene 使用的评分函数计算出的是对于某个查询 q 的文档 d 的得分

$$Score = \sum_{t\_in\_q} tf(t\_in\_d) \bullet idf(t) \bullet boost(t.field\_in\_d) \bullet len(t.field\_in\_d)$$

对评分函数中每个因子的分析如下：

(1)tf 是某个索引项在一个文档中出现的频率。

(2)idf 反映有多少个文档包含了该索引项，文档数越多，该因子的值越小，反之越大。计算 idf 的常用方法是

$$Idf = \log_2(n_d / df_i) + 1$$

(3)boost(t.field in d)可以用来控制文档中的某个域(Field)对于该文档的重要性，以及某个文档在所有文档中的重要性。

(4)lengthNorm 考虑到了文档的大小，如果文档越长那么这个因子的值就会越低，反之越高。

4 高级查询

本文中 Lucene 支持的操作有：逻辑与(AND)，逻辑或(OR)，逻辑非(NOT)，星号(*)，问号(？)及模糊查询(~)。这些符号的应用如下：

(1)逻辑查询语句

1)Lucene AND Java 返回所有既含有 Lucene 又含有 Java 的文档；

2)Lucene OR Java 返回含有 Lucene 或者 Java 的文档；

3)Lucen NOT Java 返回所有含有 Lucene 但不含有 Java 的文档。

(2)通配符查询语句

1)Luce*返回所有包含以 Luce 为前缀的词的文档，星号(*)代表零个或者多个字符；

2)Luc?ne 包含了通配符问号(？)，其中，问号(？)代表了 1 个字符。它返回所有包含这种模式的文档。

(3)模糊查询语句

1)Lucene~代表了所有和 Lucene 相似的词，并返回相应文档。

2)Lucene~0.8 代表了所有和 Lucene 相似，并且相似度大于 0.8 的词。相似度的大小在 0 到 1 之间。

5 压缩算法

在 Lucene 中，主要对 2 种数据进行压缩，即整型数据和索引项(term)。索引项的数据类型是字符串(string)。

5.1 整型数据的压缩

Lucene 采用的是一种可变长度编码的方法对 Lucene 索引文件的整型数据进行压缩。在该编码中，每个整型数据不是用固定的字节数来表示的，而是随整型数据值的不同而变化，值越大，占用的字节数越多，反之越少。每个字节的最高位表示是否还有后续字节，如果最高位是 0，则无后续字节；如果是 1，则有后续字节。这样，每个字节都会用一位来做标志位。虽然每个字节只能利用 7 位来表示整数的值，但相比用固定长度的字节来表示一个整型数据，该编码可以节约很多存储空间，而且容易解码(decoding)。在解码过程中，越是后面的字节越是处于高位。整型数据的值和编码之间的对应关系如表 1 所示。

表 1 整型数据的压缩

整型值	第 1 个字节	第 2 个字节	第 3 个字节
0	00000000		
1	00000001		
2	00000010		
3	00000011		
…	…		
127	01111111		
128	10000000	00000001	
129	10000001	00000001	
…	…	…	
16 383	11111111	01111111	
16 384	10000000	10000000	00000001
16 385	10000001	10000000	00000001
16 386	10000010	10000000	00000001
…	…	…	…

5.2 索引项的压缩

由于是全文检索，索引项的数量非常庞大，因此如何对索引项进行有效压缩成了节约存储空间的关键问题。Lucene 采用前缀压缩算法，即在索引文件中，所有的索引项按照字母的字典顺序排序。对每个索引项，Lucene 存储的信息是：(1)该索引项和前一个索引项公共前缀的长度；(2)该索引项的后缀。如果是第一个索引项，那么它的公共前缀的长度为 0。例如，如果上一个索引项是“ story ”，该索引项与上一个索引项的公共前缀长度是 4；后缀是“ m ”；值为“ storm ”。这样便节约了大量的存储空间并且容易解码。

6 索引效率的提高

在用 Lucene 为文本文档建立索引时，可以通过调整参数的设置来提高建立索引的效率。这些参数主要有 2 个：

(1)合并因子(merge factor)。合并因子是指当索引文件中片段(fragment)的最大数量，当片段数量达到合并因子的值时，这些片段就会被合并成一个新的片段。因此，合并因子越大，索引过程中所需要做的合并片段的磁盘操作就会越少，建立索引所需要的时间也就越少。

(下转第 118 页)