

## 第 8 章 Lucene 基础

搜索引擎是在 Internet 上获取信息的最常用方法之一。本章作为 Lucene 这一部分的第一章，主要介绍 Lucene 的发展历史，包括 Lucene 的起源、发展、现状以及 Lucene 的一些初步应用，以此作为使用 Lucene 的起点。建立索引和搜索是搜索引擎最重要的功能，在本章中介绍了建立索引和搜索的基本原理和操作过程。另外 Lucene 中的“倒排文档”原理是理解和使用 Lucene 的重要基础，所以这部分的内容也在本章中向读者作了介绍。

这一章的主要内容将包括：

- 信息获取与搜索引擎的发展；
- Lucene 的历史；
- 建立索引和搜索。

### 8.1 信息获取与搜索引擎

随着计算机技术和互联网技术的飞速发展，网络上的信息量急剧增长，要在浩如烟海的网络世界中寻找需要的信息，作为现代信息获取技术的主要应用——搜索引擎是不必可少的。

#### 8.1.1 信息获取

互联网（Internet）正以前所未有的态势改变着整个世界，它现在已经成为了人类有史以来资源数量最多、资源种类最全、资源规模最大的一个综合信息库。其信息来源丰富、分布广泛，各种类型信息资源异构地分布于网络空间中，如果不能使庞杂的信息有序化，就很难有效获取。如何准确有效地从互联网上获取信息，就显得十分迫切和重要。

信息获取技术包含信息的表示、存储、组织和对信息的访问方法。信息的表示和组织是为了让用户更容易地访问到需要的信息。一般来讲，信息获取的流程分为以下几部分。

- 在获取信息之前，首先需要构造文本数据库，即将来需要进行检索的数据。
- 在有了文本数据之后，需要建立文档的索引。利用索引技术可以大大提高信息检索的速度。当前有很多种建立文档索引的方法，然而对于大规模的数据量来讲，用得

最多的还是倒排索引技术。在 Lucene 中，索引部分也是使用的倒排索引的方法。

- 在建立好索引之后，就可以对其进行检索了。用户首先给出一个查询，该查询将被分析，然后利用文本处理技术进行处理。在查询操作进行之前还可以对其进行一些与处理。
- 最后根据用户的查询将获取一些文档，这就是检索结果。在把检索结果反馈给用户之前，还可以对检索结果按照一定的次序排序，以使符合用户需要的文档能够排在更前面。

搜索引擎在 10 年之前对大家来讲还是一个非常陌生的概念，然而现如今，它正在逐渐地改变着人们获取信息的方式，越来越引起人们的重视。下面，先来回顾一下搜索引擎的发展历史。

### 8.1.2 搜索引擎的发展与分类

#### 1. 发展历史

曾有人说搜索引擎的鼻祖就是黄页，诞生于 19 世纪末。因为黄页，在电话诞生后成为了以电话为主体的信息门户，而且黄页把有电话的企业分门别类，的确与现在的搜索引擎有异曲同工之妙。不过，这更多地是从这两者的形式和用途做的类比。

我们所说的搜索引擎其实是在近 10 年的不断发展中逐步形成的，它建立在互联网和诸多计算机技术之上，所以很难把搜索引擎的缘起与哪个具体的产品对应起来。然而，在它逐步发展的过程中，一些关键系统和产品的产生成为了具有里程碑意义的事情。

1993 年 10 月 Martijn Koster 创建了 ALIWEB (Martijn Koster Annouces the Availability of Aliweb)，它相当于 Archie 的 HTTP 版本。ALIWEB 不使用网络搜寻 Robot，如果网站主管们希望自己的网页被 ALIWEB 收录，需要自己提交每一个网页的简介索引信息，类似于后来大家熟知的 Yahoo。

1993 年 2 月，6 个 Stanford (斯坦福) 大学生的想法是分析字词关系，以对互联网上的大量信息作更有效的检索，这就是 Excite，后来曾以概念搜索闻名。1994 年 1 月，第一个既可搜索又可浏览的分类目录 EInet Galaxy(Tradewave Galaxy)诞生。除了网站搜索，它还支持 Gopher 和 Telnet 搜索。

Lycos 是搜索引擎史上又一个重要的进步。Carnegie Mellon University 的 Michael Mauldin 将 John Leavitt 的 spider 程序接入到其索引程序中，创建了 Lycos。除了相关性排序外，Lycos 还提供了前缀匹配和字符相近限制，Lycos 第一个在搜索结果中使用了网页自动摘要。

1998 年 10 月之前，Google 只是 Stanford 大学的一个小项目 BackRub。1999 年 2 月，Google 完成了从 Alpha 版到 Beta 版的蜕变。Google 在 Pagerank、动态摘要、网页快照、DailyRefresh、多文档格式支持、地图、股票、词典、寻人等集成搜索、多语言支持、用户界面等功能上的革新，像 Altavista 一样，再一次彻底改变了搜索引擎的定义。

注意：计算机技术正在飞速发展，关于搜索引擎的定义和发展过程，也有各种各样不同的观点。

## 2. 分类

搜索引擎并没有一个精确的定义，一般来讲，大致可以分为两大类：全文搜索引擎（FullText Search Engine）和分类目录（Directory）。

全文搜索引擎通过一个叫网络机器人（Spider）或叫网络蜘蛛（Crawlers）的软件，自动分析网络上的各种链接并获取网页信息内容，按规则加以分析整理，记入数据库。Google、百度就是比较典型的全文搜索引擎系统。

分类目录则是通过人工的方式收集整理网站资料形成数据库的，比如雅虎中国以及搜狐、新浪、网易等网站的分类目录。

这两种类型的搜索引擎各有自己的优缺点。全文搜索引擎的使用以关键词和一定的语法为特点，而分类目录则通过建立多级目录对网站进行分类。它们在使用上各有长短。全文搜索引擎因为依靠网络机器人搜集数据，所以数据库的容量非常庞大，但是，它的查询结果往往不够准确；分类目录依靠人工收集和整理网站，能够提供更为准确的查询结果，但收集的内容却非常有限。

此外，基于这两类搜索引擎，还衍生了其他的搜索服务，主要有元搜索引擎（META Search Engine）和集成搜索引擎（All-in-One Search Page）等，这里就不一一介绍了。

搜索引擎既然没有明确的定义，一般就以其发展中一些里程碑式的应用标志其阶段。大多数人普遍的共识是：

- 第一代搜索引擎：是依靠于人工分拣的分类目录搜索，以“雅虎”为标志。
- 第二代搜索引擎：是依靠于机器抓取，并建立在超级链接分析技术基础之上的网页搜索，以“Google”为代表，其信息量大、更新及时，但返回信息过多，可能有很多无关信息。
- 第三代搜索引擎：是把“智能化”、“人机交互”等功能融入了主流。将自动分类技术、中文内容分析技术及区域识别技术应用到大型搜索引擎中，除了在信息检索速度、更新频率等基本技术指标方面处于领先地位外，它的网页相关检索、拼音纠错、模糊查询、语音查询技术也具有很高的水准。此外，还同时兼备了新闻、MP3、图片、FLASH 搜索功能，已成为能够提供全面、综合的信息搜索服务。

## 8.2 Lucene 的历史

Lucene 是一个支持全文检索的开源工具包，在向大家介绍 Lucene 之前，先来讲解一下什么是全文检索和全文检索系统。

### 8.2.1 什么是全文检索与全文检索系统

全文检索是指计算机索引程序通过扫描文章中的每一个词，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。这个过程类似于通过字典中的检索字表查字的过程。

全文检索的方法主要分为按字检索和按词检索两种。

- 按字检索是指对于文章中的每一个字都建立索引，检索时将词分解为字的组合。对于各种不同的语言而言，字有不同的含义，比如英文中字与词实际上是合一的，而中文中字与词有很大分别。
- 按词检索指对文章中的词，即语义单位建立索引，检索时按词检索，并且可以处理同义项等。英文等西方文字由于按照空白切分词，因此实现上与按字处理类似，添加同义处理也很容易。中文文字则需要切分字词，以达到按词索引的目的，关于这方面的问题，是当前全文检索技术尤其是中文全文检索技术中的难点，在此不做详述。

全文检索系统是按照全文检索理论建立起来的用于提供全文检索服务的软件系统。一般来说，全文检索需要具备建立索引和提供查询的基本功能，此外现代的全文检索系统还需要具有方便的用户接口、面向 WWW 的开发接口、二次应用开发接口等。

功能上，全文检索系统具有建立索引、处理查询返回结果集、增加索引、优化索引结构等功能，外围则由各种不同应用具有的功能组成。结构上，全文检索系统具有索引引擎、查询引擎、文本分析引擎、对外接口等，加上各种外围应用系统共同构成了全文检索系统。

图 8-1 展示了全文检索系统的结构与功能。

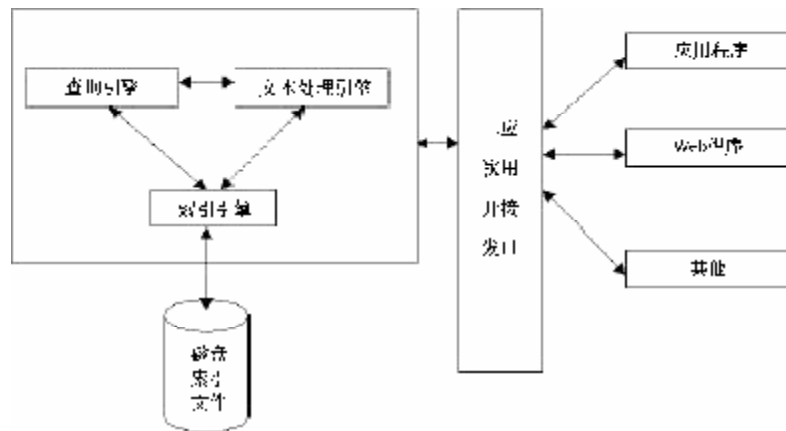


图 8-1 全文检索系统结构

## 8.2.2 什么是 Lucene

Lucene 是 Apache 软件基金会 Jakarta 项目组的一个子项目，是一个开放源代码的全文检

索引引擎工具包，即它不是一个完整的全文检索引擎，而是一个全文检索引擎的架构，提供了完整的查询引擎和索引引擎及部分文本分析引擎（英文与德文两种西方语言）。Lucene 的目的是为软件开发人员提供一个简单易用的工具包，以方便地在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎。

Lucene 的原作者是 Doug Cutting，他是一位资深全文索引/检索专家，曾经是 V-Twin 搜索引擎的主要开发者，后在 Excite 担任高级系统架构设计师，目前从事于一些 Internet 底层架构的研究。一开始，Doug Cutting 将 Lucene 发表在自己的个人主页上，2000 年 3 月将其转移到了 sourceforge 上，并于 2001 年 10 月捐献给 Apache，使 Lucene 成为 Jakarta 的一个子工程。

注意：Lucene 是 Doug 人的 Middle Name。

### 8.2.3 Lucene 的发展和现状

经过多年的发展，Lucene 在全文检索领域已经有了很多的成功案例，并积累了良好的声誉。基于 Lucene 的全文检索产品和应用 Lucene 的项目在世界各地已经非常之多，其中比较知名的如下。

- Eclipse: 主流 Java 开发工具，其帮助文档采用 Lucene 作为检索引擎。
- Jive: 知名论坛系统，其检索功能基于 Lucene。
- Ifinder: 出自德国的网站检索系统，基于 Lucene (<http://ifinder.intrafind.org/>)。
- MIT DSpace Federation: 一个文档管理系统 (<http://www.dspace.org/>)。

国内外采用 Lucene 作为网站全文检索引擎的也很多，比较知名的有：

- <http://www.blogchina.com/weblucene/>;
- <http://www.ioffer.com/>;
- <http://search.soufun.com/>;
- <http://www.taminn.com/>。

在所有这些案例中，开源应用占了很大一部分，但更多的还是商业化产品和网站。毫不夸张地说，Lucene 的出现，极大地推动了全文检索技术在各个行业或领域中的更深层次应用。

### 8.2.4 使用 Lucene 能做什么

Lucene 可以对任何的数据做索引和搜索。Lucene 不管数据源是什么格式，只要它被转化为文字的形式，就可以被 Lucene 所分析利用。也就是说不管是 Word、Html、PDF 还是其他什么形式的文件只要你可以从中抽取出文字形式的内容就可以被 Lucene 所用，就可以用 Lucene 对它们进行索引以及搜索。

## 8.2.5 谁在使用 Lucene

作为一个开放源代码的项目，Lucene 从问世之时起，就引发了开放源代码社区的巨大反响。程序员们不仅使用它构建具体的全文检索应用，而且也将它集成到了各种系统软件中，以及用它来构建各种 Web 应用，甚至某些商业软件也采用了 Lucene 作为其内部全文检索子系统的核心。例如，众所周知的 Apache 软件基金会的网站使用了 Lucene 作为全文检索的引擎，IBM 公司开源软件 Eclipse 的 2.1 版本中也采用了 Lucene 作为帮助子系统的全文索引引擎，相应的 IBM 公司的商业软件 Web Sphere 中也采用了 Lucene。Lucene 正以其开放源代码的特性、优异的索引结构、良好的系统架构获得了越来越多的应用。

## 8.3 建立索引和搜索

建立索引和搜索是搜索引擎最重要的也是最基本的两部分。不过，这涉及了方方面面的知识，在这里只先向大家介绍最基础的，以后的内容会越来越深入地介绍相关内容。

### 8.3.1 索引

基本的查询方法可以通过顺序扫描文本的方式来实现，这种方法被称作是“顺序查找”。使用这样的方法之前，不需要对文档集中的信息做任何形式的预处理，当用户进行查询时，直接在文档中进行字符串的简单匹配。相对来讲，这种方法比较简单，容易实现，但是，同时也存在一个很大的缺点。当需要查找的文件大小达到一定数量级别的时候，这种查找方法的效率就成为一个很大的问题。特别是需要查找的文件数量特别大时，这种方法几乎已经不能满足实际的需要了。

正因如此，人们又创造了各种不同的查找方法，使用索引就是其中的一种方法。

索引，是在搜索时使用到的一种特殊的数据结构。当文档的数量相当庞大，并且这些文档中的信息相对稳定时，建立索引可以大大提高搜索时的效率。不过，这种索引结构不支持快速的信息更新，即当建立完成索引后，如果文档中的信息发生了变化，也必须对索引进行更新，才能够使用索引检索到最新的信息。

**注意：**在实际中，通常是通过定期更新索引，并把索引添加到原索引中实现的。

在使用索引进行查找时，首先对需要索引的文档进行预处理，建立关于这些文档的索引结构。索引的技术主要有以下 3 种：倒排索引，后缀数组和签名文件。其中，倒排索引技术在当前大多数的信息检索系统中得到了广泛的应用，它对于关键词的搜索非常有效，在 Lucene 中也是使用的这种技术。后缀数组技术在短语查询中具有很快的速度，但是这样的数据结构在构造和维护时都比较复杂一些。签名文档技术在 20 世纪 80 年代时期比较流行，但是后来倒排索引技术逐渐超越了它。

代码 8.1 所示为 Lucene 中建立索引时所需的最基本操作。

### 代码 8.1 索引的建立

```
package ch8;
import java.io.File;
import java.io.IOException;
import java.util.Date;

import org.apache.lucene.analysis.SimpleAnalyzer;
import org.apache.lucene.demo.FileDocument;
import org.apache.lucene.index.IndexWriter;

public class IndexTest
{
    //测试函数的入口点
    public static void main(String[] args)
    {
        //记录开始索引的时间
        Date start = new Date();
        try
        {
            //生成索引书写器, 新生成的索引存放在“C:\\IndexDir”目录中
            IndexWriter writer = new IndexWriter("C:\\IndexDir", new SimpleAn-
alyzer(), true);
            //将需要索引的文本文件转化为一个 File 对象
            File file = new File("C:\\IndexData.txt");
            //输出提示信息
            System.out.println("adding " + file);
            //将文档添加到索引种
            writer.addDocument(FileDocument.Document(file));
            //索引的优化
            writer.optimize();
            //关闭书写器, 此时会把索引写入到磁盘中
            writer.close();
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        //记录索引结束的时间
        Date end = new Date();
        //计算索引花费的时间, 并输出
        System.out.print(end.getTime() - start.getTime());
        System.out.println(" total milliseconds");
    }
}
```

可以看到, 在建立索引时首先需要声明索引文件的存放路径和需要进行索引的文本文件, 然后调用 `IndexWriter` 对象的 `addDocument()` 方法, 将文本文件添加到索引中。但是, 此时

Lucene 还没有将索引写入到系统的磁盘中。当调用 `IndexWriter` 对象的 `close()` 方法时才会将索引写入到磁盘中。

此外在调用 `IndexWriter` 对象的 `close()` 方法之前，还调用了 `IndexWriter` 对象的 `optimize()` 方法，这又是为什么呢？这个操作是用来做什么的呢？这个问题暂且放置一边，在关于索引优化的相关章节中会有比较详细和系统地介绍。

代码 8.1 的运行效果如图 8-2 所示

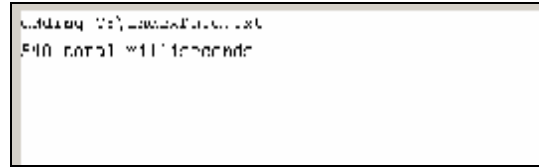


图 8-2 索引测试结果

### 8.3.2 搜索

检索系统在建立好索引以后，既可以对用户的查询做出响应，该响应过程是通过搜索过程来实现的。搜索的目的就是要为用户提供高质量的搜索结果。既要求响应时间快，又要求搜索结果准确，这就涉及了搜索算法的问题。

此外，搜索程序在搜索到一定数量的结果之后，就要以一定的方式把搜索结果反馈给用户，包括以什么样的格式和什么样的次序返回给用户。用户最关心的还是所搜索到的结果是不是用户真正想要的，而且这些用户想要的结果当然是排在越前面越好，这就涉及了搜索结果的排序问题。

Lucene 之所以成为一个非常优秀的搜索引擎代码包，与它高效的搜索算法和对搜索结果丰富的排序方法是密不可分的。在这里先只向大家介绍下 Lucene 搜索的一般流程，在以后的章节中有关于搜索更详细的讲解。

请大家阅读代码 8.2。

#### 代码 8.2 简单的搜索过程

```
package ch8;
import org.apache.lucene.store.*;
import org.apache.lucene.document.*;
import org.apache.lucene.analysis.*;
import org.apache.lucene.index.*;
import org.apache.lucene.search.*;
import org.apache.lucene.queryParser.*;

class SearchTest
{
    public static void main(String[] args)
    {
        try
        {
            //在内存中建立索引
            Directory directory = new RAMDirectory();
```

```
//生成分析器对象，用于分词等
Analyzer analyzer = new SimpleAnalyzer();
//索引书写器
IndexWriter writer = new IndexWriter(directory, analyzer, true);
//将要建立索引的字符串
String[] docs =
{
    "a b c d e",
    "a b c d e a b c d e",
    "a b c d e f g h i j",
    "a c e",
    "e c a",
    "a c e a c e",
    "a c e a b c"
};
//讲要建立索引的字符串添加到索引中
for (int j = 0; j < docs.length; j++)
{
    Document d = new Document();
    d.add(Field.Text("contents", docs[j]));
    writer.addDocument(d);
}
writer.close();
//生成搜索对象
Searcher searcher = new IndexSearcher(directory);
//用于查询的字符串
String[] queries = {
    "\"a c e\"",
};
//生成结果集对象，初始化为空值
Hits hits = null;
//生成 QueryParser 对象
QueryParser parser = new QueryParser("contents", analyzer);
//依次使用查询字符串生成查询对象 Query
for (int j = 0; j < queries.length; j++)
{
    Query query = parser.parse(queries[j]);
    //输出要查询的内容
    System.out.println("Query: " + query.toString("contents"));
    //返回结果集
    hits = searcher.search(query);
    //输出搜索到的总文档数
    System.out.println(hits.length() + " total results");
    //依次输出搜索到的文档的内容
    for (int i = 0 ; i < hits.length() && i < 10; i++)
    {
        Document d = hits.doc(i);
        System.out.println(i + " " + hits.score(i)+ " " + d.get("contents"));
    }
}
```

```

    }
    searcher.close();

} catch (Exception e)
{
    System.out.println(" caught a " + e.getClass() +
        "\n with message: " + e.getMessage());
}
}
}

```

在代码 8.2 中，为了能够进行搜索，首先对字符串数组 docs 建立索引，这属于索引部分的内容。在索引建立完成以后，声明了一个查询字符串的数组对象 queries，里面存储着各个想要查询的字符串，然后生成一个 QueryParser 类型的对象，其参数“contents”表示在“contents”字段中进行查询。再使用 QueryParser 对象的 parse() 方法将查询字符串转换成为一个 Query，最后使用 Searcher 对象的 search() 方法进行查询，把返回的结果存储在 Hits 类型的对象中。

**注意：**关于字段等相关概念在本书第 10 章中有详细介绍，请大家参考相应的章节。

代码 8.2 的运行效果如图 8-3 所示

### 8.3.3 倒排索引

倒排索引是一种面向单词的索引机制，利用它可以提高检索时的速度。通常情况下，倒排索引结构由“词典”和“出现情况”两部分组成。对于每一个单词，都会有一个词汇列表记录单词在所有文档中出现的位置，这些位置可以是单词的位置（文本中的第几个单词）也可以是字符的位置（文本中的第几个字符）。

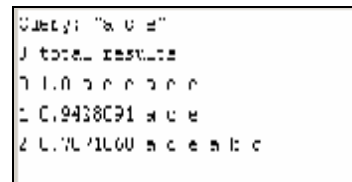


图 8-3 搜索测试代码

如果使用正常的索引结构，建立的是“文档到单词”的映射关系，在使用倒排索引技术后，建立的是“单词到文档”的映射关系，那么这两种映射关系到底有何不同呢？它们各自有什么有缺点呢？下面举例向大家说明这两种映射关系的差别。

假设现在有两篇文档：文档 A 和文档 B。文档 A 的内容是：This is a dog。文档 B 的内容是：The dog is a kind of animal。

下面对这两个文档建立索引结构。

**注意：**在这里只是为了介绍倒排索引与一般索引的区别，真正的索引格式会比此处介绍得复杂很多。

如果建立的是一般的索引结构，那么会有如表 8-1 所示的关系。

表 8-1 一般的索引格式示例

文档编号	出现单词	出现次数	文档编号	出现单词	出现次数
A	dog	1		king	1
B	dog	1		animal	1

从中可以看出，一般的索引结构是以文档为标准建立索引结构的，即它记录的是一篇文档中所有单词出现的情况。比如在文档 B 中 dog,kind,animal 均出现了一次。然而，用户在进行检索时，都是输入关键字进行查询，如果使用这种索引结构，在查询某一关键字时往往需要遍历所有的索引，当索引量非常大时，效率会成为一个很大的问题。

倒排索引恰恰解决了这个问题，它是关键字为标准建立索引的。

从表 8-2 可以看出，倒排索引是以单词为标准建立的索引结构，它描述了一个单词在所有文档中的出现情况，比如说单词“dog”在文档 A 和文档 B 中分别出现了一次，而单词“kind”只在文档 B 中出现了一次。

表 8-2 倒排索引结构示例

单 词	出现的文档	出现次数	单 词	出现的文档	出现次数
dog	A,B	1,1	animal	B	1
kind	B	1			

通过比较可以发现，一般的索引结构建立的是一种“文档到单词”的映射关系，而倒排索引建立的则是一种“单词到文档”的映射关系。因为在日常的检索中，通常都是按照关键字进行搜索的，所以，倒排索引可以更好地适合这种检索机制的需要。这也是倒排索引如今被大规模使用的原因。

## 8.4 总结

本章向读者介绍了关于 Lucene 的一些基本知识，虽然在本章中并没有涉及很多技术性的问题，但是要想成为一位使用 Lucene 的高手，这些基本知识也是不可或缺的。读者通过这一章的学习，应该可以对 Lucene 有了一些概括性的了解。下一章将带领读者使用 Lucene 建造第一个搜索引擎。