**--搜索引擎开发**

目 录

[1. 工程背景 4](#_Toc444972181)

[1.1 搜索引擎简介 4](#_Toc444972182)

[1.2 工程概述 4](#_Toc444972183)

[2. 网络爬虫 6](#_Toc444972184)

[2.1 爬虫概述 6](#_Toc444972185)

[2.2 流程描述 6](#_Toc444972186)

[2.3 问题分析 7](#_Toc444972187)

[2.4 设计与实现 7](#_Toc444972188)

[2.5 运行结果 8](#_Toc444972189)

[3. PageRank值计算 9](#_Toc444972190)

[3.1 PageRank简介 9](#_Toc444972191)

[3.2 算法流程 10](#_Toc444972192)

[3.3 代码实现 11](#_Toc444972193)

[3.4 部分页面的PageRank值展示 12](#_Toc444972194)

[4. 建立倒排索引 12](#_Toc444972195)

[4.1 倒排索引概念 12](#_Toc444972196)

[4.2 建索引流程 12](#_Toc444972197)

[4.3 核心代码解析 13](#_Toc444972198)

[4.4 倒排索引结果 14](#_Toc444972199)

[5. 搜索过程及可视化 15](#_Toc444972200)

[5.1 搜索原理 15](#_Toc444972201)

[5.2 搜索流程 15](#_Toc444972202)

[5.3 核心代码解析 16](#_Toc444972203)

[5.4 可视化展示 16](#_Toc444972204)

[6. 日志存储 17](#_Toc444972205)

[6.1 日志格式 17](#_Toc444972206)

[6.2 日志存储 17](#_Toc444972207)

[6.3 日志挖掘 18](#_Toc444972208)

[7. 工程总结 19](#_Toc444972209)

[**附录 1：小组成员及任务分配表** 20](#_Toc444972210)

[**附录 2：参考文献** 20](#_Toc444972211)

## 工程背景

### 1.1 搜索引擎简介

网页作为因特网的重要组成部分，早已成为人们获取网络信息的主要途径。为了方便人们从整个网络的海量网页中快速检索出所需信息，一种快速检索工具——搜索引擎应运而生。目前依据工作方式的不同主要分为全文搜索引擎、目录搜索引擎及元搜索引擎。全文搜索引擎是真正意义上的搜索引擎，主要从网络中提取网页信息，建立数据库与用户查询关键字匹配，并按一定顺序返回检索结果。目录搜索引擎是按目录分类的网站链接列表，类似网址导航网站。元搜索引擎是同时在多个搜索引擎上检索，并将结果返回。本次课程作业是实现小型全文搜索引擎。

搜索的本质是满足用户通过查询关键字表达的需求，以此为目的促生分词、倒排索引、PageRank等技术作为方法，不断发展优化搜索引擎的效率，质量等方面。搜索引擎大致表现出三个发展阶段：第一阶段使用倒排索引与文本模型解决搜索效率和相关性问题使得搜索引擎可扩展，如lnfoseek；第二阶段使用PageRank算法提升搜索质量，如Google；第三阶段将用户行为分析运用于垂直搜索提升搜索结果，同时弱化分词与文本匹配，而是通过查询关键字改写使得搜索结果更加直接有效。

搜索引擎有很多技术模块构成，各自负责整体功能的一部分，相互配合形成完善的整体架构。

### 1.2 工程概述

利用Java编程语言、JSP、C#实现的网络爬虫、Mysql数据库、Lucene开源工具包、PageRank排序算法来构建一个简单的全文搜索引擎。本工程中分块实现搜索引擎后台数据处理功能，按照爬虫爬取网页数据，PageRank算法计算网页在数据集中的权重，建立倒排索引文件的顺序进行后台处理。而前端使用Java Web工程进行搜索与结果展示，同时将用户的搜索历史以日志形式存储于mysql数据库中。整个工程的流程图如图1.1所示。

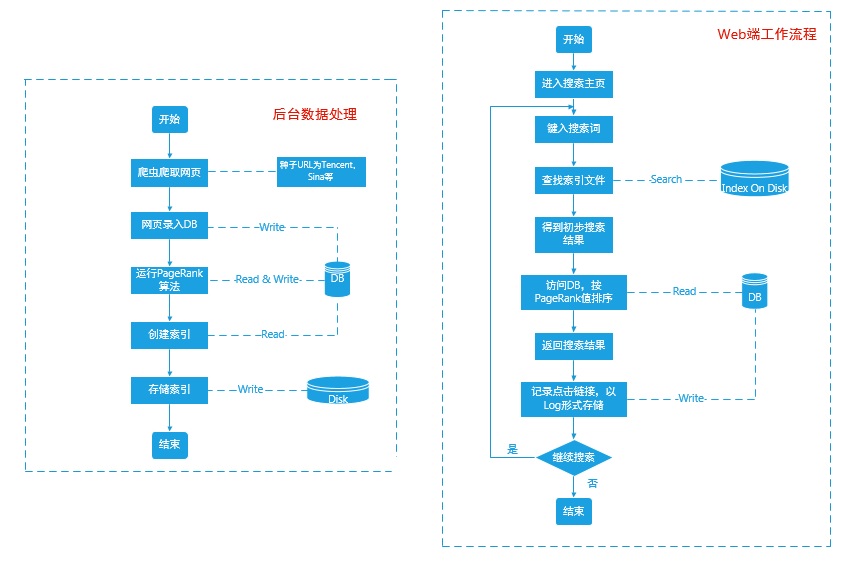
****

图1.1 搜索引擎全局流程图

将图1.1中各个模块分解开来，各部分主要内容如下:

1. 使用C#语言编写爬虫程序，支持增量式网页数据的采集，并实现对新浪、网易、腾讯、搜狐、百度5个root网站网页的爬取，以广度优先方式爬取5000个网页。针对采集回来的网页去重，将结果存入mysql数据库中。
2. 读取mysql中的网页数据，根据网页间指向关系表计算网页的PageRank值。然后使用lucene开源工具包建立倒排索引文件，存储于磁盘上。
3. 设计并编写Web搜索主页，提取用户键入的查询关键字进行解析，查找倒排索引文件进行网页文本匹配，匹配后的网页结合网页的PageRank值进行排序，并返回搜索结果展示页面给用户，同时记录用户搜索日志。

其中网络爬虫通过5个种子网站的超链接，将与之相链接的网页信息爬取到本地，进行去重、去噪、分词等提取文档关键字，并将网页的文本、标题与URL一一对应存入mysql数据库，对数据库中存放网页信息的数据表建立倒排索引。设计编写web搜索界面，将用户键入的搜索词传递给后台的servlet程序，在servlet中使用Lucene检索工具在倒排索引文件中进行搜索词组匹配，然后将匹配的网页按照文本相似度和PageRank值进行加权后排序，将排序结果返回到搜索结果页。

## 网络爬虫

### 2.1 爬虫概述

网络爬虫是指利用http协议检索Web文档的软件，是功能很强的自动提取网页的程序，为搜索引擎从互联网采集原始的网页数据，是搜索引擎的重要组成部分。

互联网中的网页类似一个有向图。爬虫从种子网站的url集合开始，沿着这些种子网页中包含的超链接，按照广度优先遍历抓取网页。爬虫不停地从待抓取的url集合中移除已访问的url，并将未访问过的网页链接加入到待爬url集合，实现增量式抓取。如此不停地下载相应的网页，直到抓取总量满足搜索引擎的需求。

### 2.2 流程描述

1. 首先选择一些知名的门户网站的首页，以这些网页的链接地址作为种子url；
2. 将这些种子url放入待抓取url队列中；
3. 爬虫依次读取url队列中的这些url，并对每一个url进行下载；
4. 对于下载到本地的页面：
5. 进行解析，去除所有的html标签，并将其存储到页面库中，等待建立索引等后续处理；
6. 将下载网页的url放入己抓取url队列中，这个队列记载了爬虫系统己经下载过的网页url，以避免网页的重复抓取；
7. 从中抽取出所包含的所有链接信息，并在已抓取的url队列中进行检查，如果发现链接还没有被抓取过，则将这个url放入待抓取url队列；
8. 重复以上操作，直到抓取的网页总量满足搜索引擎的需求。

### 2.3 问题分析

简单来看，爬虫只需要一个WebClient来抓取网页，一个列表用于存储抓取的结果，一个列表用于存储待抓取的url列表即可。对于所有的待抓取url，交给WebClient进行抓取，然后进行分析，写数据库，一直循环即可。然而实际的网页抓取过程往往存在着很多问题：

1. 不同的网页的编码往往不相同：有utf-8，GB2312，GBK，ISO-8859-1等等，要进行文本的分析，就需要先对原始网页的编码进行识别并转换为统一的编码；
2. 网页中不仅包含着文本内容，还包含了很多的html标签和javascript脚本等等。这些东西对于网页内容的搜索毫无意义，需要进行有效的去除；
3. 与MySQL数据库的通信过程也存在着编码的问题，需要进行编码转换才能进行存储；
4. 要有容错性：对于不能访问的url、不能识别的网页编码、不能分析的网页内容、写数据库出错的情况，应该能够进行处理，并不影响整个抓取过程的运行。

### 2.4 设计与实现

爬虫相关的数据库表结构如下。

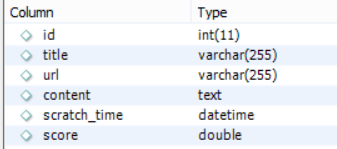


图2.1 webpage表结构

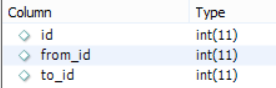


图2.2 urllink表结构

其中webpage包含id, title, content, url, scratch\_time, score（用于存储PageRank分数），urllink包含id, from\_id, to\_id这3个字段，用于存储url之间的跳转关系，供PageRank算法调用。

爬虫程序基于C# .NET Framework 4.5实现。由抓取，分析和数据库存储3部分构成。网页的抓取和处理用知名的.NET html解析库HtmlAgilityPack来实现，数据库的通信用MySql官方提供的MySQL.Data来实现。

如下图2.3所示，静态库CrawlerWrapper对爬虫的所有功能和数据进行了封装，以方便UI的调用和代码的重用。

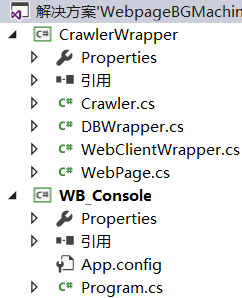


图2.3 静态库CrawlerWrapper

WB\_Console是一个命令行界面，用来显示正在抓取的网页和其他信息。直接设定好之后调用Crawler的Start即可进行抓取。只需要设定好root url和数量，爬虫即可开始工作。如下图所示。

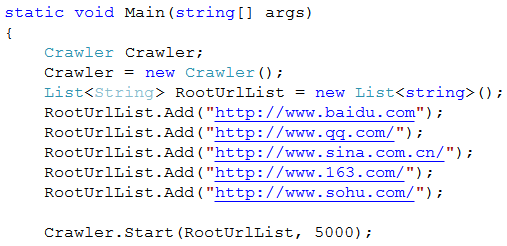


图2.4 爬虫的root url

### 2.5 运行结果

如图2.5和2.6所示，可以看到爬虫获取的网页和网页跳转关系,表明爬虫可以正常地工作。此程序在Win8系统，16G内存，频率2.6GHzCPU的环境下，运行时间为2.0 h。

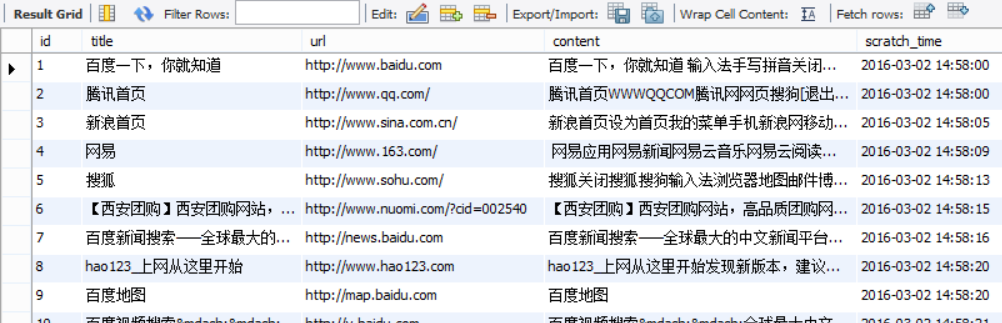


图2.5 爬虫爬取网页webpage表截图

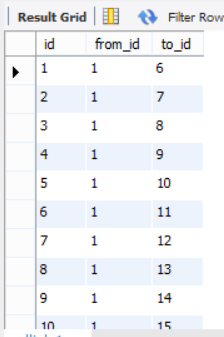


图2.5 爬虫爬取网页链接关系urllink表截图

## PageRank值计算

### 3.1 PageRank简介

Google PageRank，网页排名，是一种根据网页之间相互的超链接计算的技术。Google用它来体现网页的相关性和重要性，在搜索引擎优化操作中是经常被用来评估网页优化的成效因素之一。

PageRank通过网络浩瀚的超链接关系来确定一个页面的等级。Google把从A页面到B页面的链接解释为A页面给B页面投票，Google根据投票来源（甚至来源的来源，即链接到A页面的页面）和投票目标的等级来决定新的等级。简单的说，一个高等级的页面可以使其他低等级页面的等级提升。

该算法主要基于重要性平均分配这一朴素思想，简单来说，A页面通过指向B页面，将一部分重要性传递给了B页面。而一个页面的PageRank值是由所有链向它的页面的重要性经过递归算法得到的。

### 3.2 算法流程

下面给出一个简单的例子，有页面A、B、C和D，它们之间的指向关系如图3.1所示。

****

图3.1 PageRank例图

以D页面为例，其PageRank值的计算公式公式3-1所示，此处注意所有页面的PageRank值初始化为1:

|  |
| --- |
| PR(D)=(1-d)+d\*( PR(B)/C(B)+PR(C)/L(C)+PR(A)/L(A)) （3-1） |

其中PR(D)代表D页面的PageRank值；C(B)代表B页面的出度，即B页面链出页面的数量；d ，damping factor，代表用户跟进一个链接的可能性，一般设置为0.85。(1-d)，即0.15，由于“没有向外链接的页面”传递出去的PageRank值会是0，所以，Google通过数学计算给了每个页面一个最小值。

通过一次迭代，得到各页面的PageRank值，如表3.1所示。

|  |  |
| --- | --- |
| URL | PageRank |
| A | 0.15 |
| B | 1.0425 |
| C | 0.6355625 |
| D | 1.175790625 |

表3.1 1次迭代PageRank值

经过不断的重复计算，这些页面的PR值会趋向于稳定，也就是收敛的状态。一般网页规模较小的情况下，迭代20次后各页面的PR值会趋于稳定，经过20词迭代的结果如表3.2所示。

|  |  |
| --- | --- |
| URL | PageRank |
| A | 0.15 |
| B | 1.4927774302509356 |
| C | 0.8269304078566476 |
| D | 1.5298212545347982 |

表3.1 20次迭代PageRank值

从数值上可看出，各页面的重要新排序为：D > B > C > A，此结果也符合上面的图示。

### 3.3 代码实现

由JAVA实现，主要使用了JDBC相关技术。在做此部分是遇到了一个问题，

ResultSet的嵌套调用问题，当让使用stmt来返回一个结果集的时候，前面一个ResultSet会自动关闭，即在执行stmt.executeQuery()的时候，会自动关闭上一次stmt.executeQuery的结果集。由于不清楚这点，导致浪费了很多时间。解决办法即在外层获得结果集后将其保存下来，例如保存在ArrayList中。

第一步，读取数据库，得到所有的URL连接，存储其id：

|  |
| --- |
| String sql1 = "select id from webpage";  rs = stmt.executeQuery(sql1); //执行查询，返回查询的结果  ArrayList<String> url\_id = **new** ArrayList<String>(); //保存id  **while**(rs.next())  {  url\_id.add(rs.getString(1)); //存储结果到ArrayList中  } |

第二步，遍历所有URL的，分别计算其PageRank值，此处迭代20次，下面只展示要进行迭代的代码。对于每一个URL的PageRank值的计算可分为两部分:

首先找出指向此URL的所有URL链接

|  |
| --- |
| String sql2 = "select from\_id from urllink where to\_id = " + url\_ID; |

然后，计算得到的每个URL对此URL的PageRank值贡献：

|  |
| --- |
| //获得一个url的pagerank值--linkingpr  String sql3 = "select score from webpage where id = " + linker\_ID;  //获得此url的输出链接个数--linkingcount  String sql4 = "select count(\*) from urllink where from\_id = " + linker\_ID;  //计算得到的每个URL对此URL的PageRank值贡献  url\_score += 0.85\*(linkingpr/linkingcount); |

### 3.4 部分页面的PageRank值展示

截取部分网页的pagerank值，如图3.2所示。可看出由于采集的网页数目有限，5000个网页之间的链接关系过于简单，所以导致计算出的PageRank值差别不大，几乎都为0.15多。而且很多网页的PageRank值相同。此程序在Win8系统，16G内存，频率2.6GHzCPU的环境下，运行时间为2.5min。

图3.2 部分网页的pagerank值数据库样图

## 建立倒排索引

### 4.1 倒排索引概念

倒排索引即反向索引，通常用于存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射，是实现单词到文档映射关系的最佳实现方式和最有效的索引结构。倒排索引由“单词词典”和“倒排文件”组成。对原始数据库中文档集合进行分析，某些词都存在于哪些文档，在每个文档中的词频及位置。文档用文档编号标记。这样与一个文档相关的信息称作倒排索引项，包含该单词的一系列倒排索引项组成了列表结构，即为该单词的倒排列表。

### 4.2 建索引流程

建立倒排索引的数据源是爬虫部分写入数据库的网页表，建好的索引以文件的形式存储在磁盘上，流程图如图4.1所示。

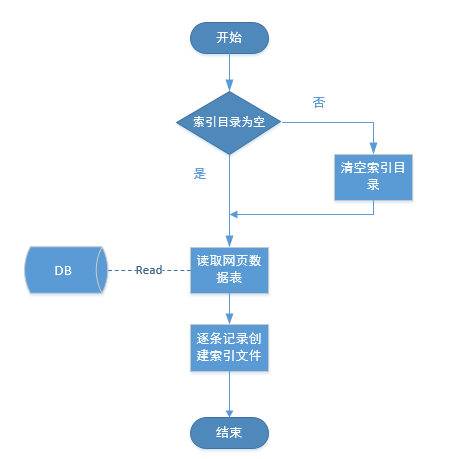


图4.1 倒排索引建立流程图

### 4.3 核心代码解析

这里建立索引使用的是Lucene开源全文检索引擎工具包，建立索引核心代码如下：

**for** (Webpage web : WebList) {

**try**{

*analyzer* = **new** StandardAnalyzer(Version.*LUCENE\_40*); //分词器

*directory* = FSDirectory.*open*(**new** File(*INDEX\_DIR*)); //INDEX\_DIR为索引目录

File indexFile = **new** File(*INDEX\_DIR*); //如果索引目录不存在，创建之

**if** (!indexFile.exists()) {

indexFile.mkdirs();

}

// IndexWriterConfig是对indexWriter的配置，其中包含了两个参数，第一个是目前的版本，第二个是词法分析器Analyzer。

IndexWriterConfig config = **new** IndexWriterConfig(Version.*LUCENE\_40*, *analyzer*);

*indexWriter* = **new** IndexWriter(*directory*, config);

Document document = **new** Document();

//这里录入要索引的字段，本项目索引存储内容为网页数据表的title，content，url字段

document.add(**new** TextField("title", web.getContent(), Store.*YES*));

document.add(**new** TextField("content", web.getContent(), Store.*YES*));

document.add(**new** TextField("url", web.getContent(), Store.*YES*));

*indexWriter*.addDocument(document);

*indexWriter*.commit();

*closeWriter*();

}**catch**(Exception e){

e.printStackTrace();

}

}

### 4.4 倒排索引结果

最终程序创建的倒排索引以文件形式存储于磁盘之上，文件样式如图4.2所示。对于数据库中的5000个网页，在8G ，Win7，3.10GHz的机器上建立索引耗时接近10分钟。

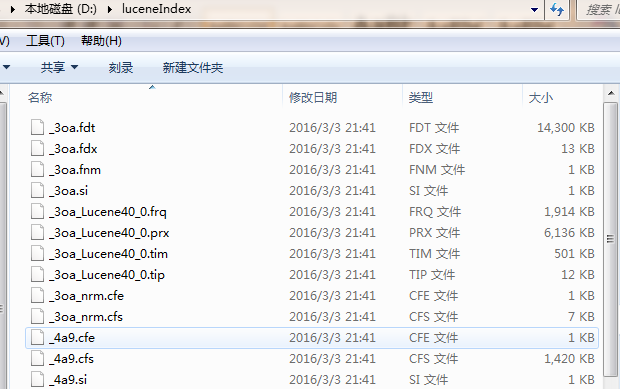


图4.2 倒排索引文件样式图

## 搜索过程及可视化

### 5.1 搜索原理

用户提交查询关键字，若键入中文，则进行分词、去重后获得检索组合。将处理后的查询词组传递给后台servlet服务器，使用Lucene检索工具搜索索引库，返回所有匹配网页的id号。再根据网页经过PageRank算法后的权重进行排序，最终返回搜索结果。

### 5.2 搜索流程

这里使用的是MyEclipse 10做了一个JavaWeb项目，主要有2个jsp页面，一个搜索主页和一个搜索结果页和若干servlet。搜索流程图如图5.1所示。

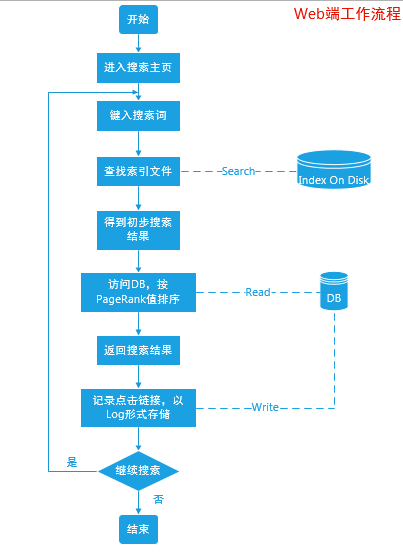


图5.1 搜索流程图

### 5.3 核心代码解析

这里的搜索主要还是用Lucene开源检索工具进行搜索。

directory = FSDirectory.*open*(**new** File(INDEX\_DIR)); //打开索引库目录

analyzer = **new** StandardAnalyzer(Version.*LUCENE\_40*); //分词器

DirectoryReader ireader = DirectoryReader.*open*(directory);

IndexSearcher isearcher = **new** IndexSearcher(ireader);//索引检索对象

QueryParser parser = **new** QueryParser(Version.*LUCENE\_40*, "content", analyzer);

Query query = parser.parse(KeyWords);//以KeyWords字符串为查询词建立查询对象

TopDocs results= isearcher.search(query, 5000); //返回所有匹配到查找词的网页

hits = results.scoreDocs;

sql1 = "select id from urllist where id=";

**for** (**int** i = 0; i < hits.length; i++) {

Document hitDoc = isearcher.doc(hits[i].doc);

**if**(i < hits.length -1)

sql1 = sql1 + hitDoc.get("id") + " or id="; //统计搜索结果的网页id

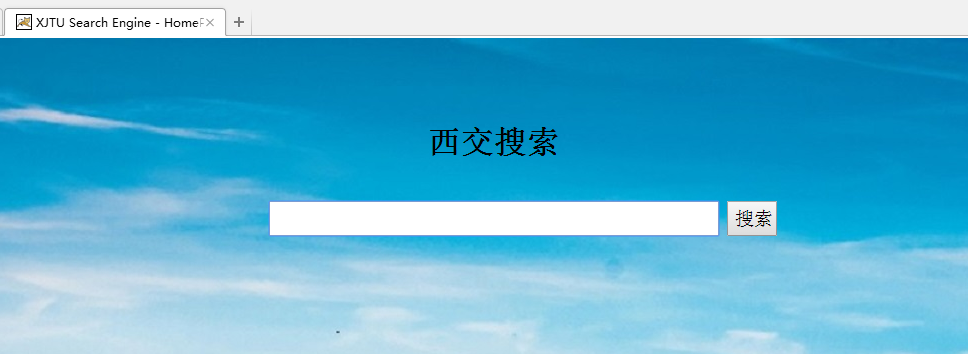
**else**

sql1 = sql1 + hitDoc.get("id") + " order by score desc"; //按score降序排列

}

### 5.4 可视化展示

搜索主页如下图5.2所示。

图5.2 搜索引擎主页

以西游记为例进行搜索,搜索结果返回页如图5.3所示。点击右侧蓝色url可进入相应的网页。



图5.3 搜索结果返回页

## 日志存储

### 6.1 日志格式

本工程中我们记录用户输入的搜索词，点击链接的url，还有搜索时间作为日志记录，存储于数据库中。具体的表设计如图6.1所示。

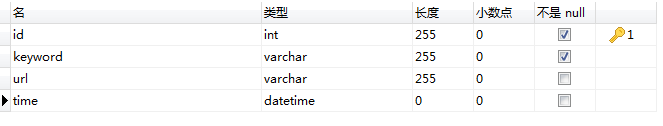


图6.1 log日志表格式

### 6.2 日志存储

日志存储在web工程中实现，用户点击搜索结果的url，则将相应数据写入数据库的log表中。由于对于web编程不是很熟悉，获取点击的url这里尚未实现，其余部分的具体代码如下。

//到数据库中去验证

Connection ct=**null**;

**int** rs= 0;

Statement ps=**null**;

SimpleDateFormat sdf = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date now = **new** Date();

String time = sdf.format(now);

String sql = "insert into log (keyword, url, time) values ('"+KeyWords+"', ' ', '"+time+"')"; //插入数据的sql语句

**try** {

//加载驱动

Class.*forName*("com.mysql.jdbc.Driver");

//得到连接 ct=DriverManager.*getConnection*("jdbc:mysql://localhost:3306/searchengine?characterEncoding=utf-8", "root", "123456");

//创建Statement

ps=ct.createStatement();

//执行操作

rs=ps.executeUpdate(sql);

//处理结果

**if**(rs>0){

System.*out*.println("insert log successfully!");

}**else**{

System.*out*.println("insert log unsuccessfully!");

}

}**catch** (ClassNotFoundException e) {

e.printStackTrace();

}**catch** (SQLException e){

e.printStackTrace();

}**finally**{

**try** {

ps.close();

}**catch** (SQLException e1) {

e1.printStackTrace();

}

**try** {

ct.close();

} **catch** (SQLException e) {

e.printStackTrace();

}

### 6.3 日志挖掘

我们的思路是这样的，通过建立一个搜索词和搜这个词用户点击链接对应表。例如用户搜索词a，点击了链接b，c，d进行访问，那么将链接b，c，d按照点击量进行降序排列。这个统计分析可以每隔一定时间段进行一次，最后返回搜索结果时可以将相似度排名、PageRank排名和相应点击量排名综合一下，给出一个符合用户搜索习惯的搜索结果排名。

因为时间、人员关系，这一部分还尚未实现。

## 工程总结

我们所做的工程基本上完成了一个搜索引擎的功能，从数据获取到建立倒排索引，再到搜索。但是很多细节上的东西没有处理好，比如url去重，网页文本分类，冗余数据处理，日志挖掘等。小组成员也曾思考各个部分该如何实现。例如对于文本分类，我们可以从网上下载已经分类过的网页；然后使用中文词典对文本内容分词，将其处理成label index:1 …的形式；然后使用liblinear开源机器学习包调用SVM算法训练出一个文本分类模型，之后就只需要把爬取到的网页处理成SVM的输入格式，使用模型进行预测分类即可。

还有就是建立倒排索引和查找索引这里我们使用的Lucene开源工具包，一定程度上降低了项目开发难度。如果自己动手写这些代码，估计也是一个不小的考验。对于日志统计分析的结果如何反作用于用户的下一次搜索，这仍然是一个值得思考的问题。

最后就是整个JavaWeb页面的美化问题，由于开发者功力不够，jsp页面比较简陋，这里可以再做一些改进。

**附录 1：小组成员及任务分配表**

**附录 2：参考文献**

1. Lucene搜索引擎开发进阶实战，成龙，机械工业出版社。
2. 解密搜索引擎技术实战:Lucene & Java精华版，罗刚，电子工业出版社。
3. 基于大规模中文搜索引擎的搜索日志挖掘，陈红涛，杨放春，陈磊，《计算机应用研究》，2008。
4. 基于SVM的中文网页分类方法的研究，牛强，王志晓，陈岱，夏士雄，《计算机工程与设计》，2007。