咕嘟体育新闻检索系统

实验报告

中国科学院大学

姬强　　201428015059068
徐培兴
倪佳志　201428015029019
陈晓旭

目录

咕嘟体育新闻检索系统.....1
实验报告.....1
1.使用的开源工具/包，以及数据库表.....3
1.1 使用的开源的工具/jar 包和用途：.....3
1.2 数据库表的列表和用途.....3
2. 体育新闻的获取和预处理.....3
2.1 新闻数据的抓取.....3
2.1.1 Heritrix 爬取体育新闻数据.....3
2.1.2 网络爬虫实现的功能.....4
2.1.3 Heritrix 中部分功能.....5
2.2 爬取数据统计.....7
2.3 网页数据信息抽取.....7
3. 分词.....8
3.1 IKAnalyzer 简介.....9
3.2 关键问题及数据库 Sharding 技术.....9
3.3 分词结果存放：.....10
4. 倒排索引的构建.....10
4.1 倒排索引的构建.....10
5.查询结果的评分排序.....12
5.1 排序参考的因素如下：.....12
检索结果得分公式如下：.....13
6.向量空间索引（正排索引）和层次聚类.....13
6.1 正排索引.....13
6.2 采用组平均凝聚式聚类算法.....14
7. 检索后期处理与检索结果展示.....15
7.1 系统框架.....15
7.2 自动补齐.....16
7.2 展示与分页.....17
7.3 显示快照.....18
7.4 关键字高亮（snippet）.....18
7.5 搜索推荐.....19
7.6 聚类显示.....20
7.7 异常处理.....20
7.8 多样查询结果展示.....21
8. 系统的评价.....22
9. 经验总结.....23

1.使用的开源工具/包，以及数据库表

项目中使用的开源的工具，以及存放数据使用的各个表名和用途。

1.1 使用的开源的工具/jar 包和用途：

开源工具/jar 包	用途
Jsoup	分析网页结构
IKAnalyzer	分词
Heritrix 爬虫工具	体育新闻网页爬虫

1.2 数据库表的列表和用途

数据库表名称	用途
Htmls	解析后的网页的各个关键信息
Bodyterm	对新闻主体分词后的结果
Otherterm	存放新闻标题、关键词、摘要的分词结果
Termed	词典 （id term 对）
Docvector	文档的向量
Body_dictionay	由 body 生成的倒排索引
term_dictionay	由 title、des、kw 生成的倒排索引
squery	历史查询信息统计

2. 体育新闻的获取和预处理

2.1 新闻数据的抓取

2.1.1 Heritrix 爬取体育新闻数据

体育新闻数据的爬虫过程采用了开源工具 Heritrix，它的工作原理是从一个提供的种子进行爬，收集站点内的精确 URL，进而通过这个 URL 继续进行网页

爬虫，主要是用广度优先算法进行处理，类似于 PageRank 的思想，同时也可以对每一个 URL 的递归深度和其结构进行限制。

网络爬虫的 URL 处理器负责对分配待下载列表中的 URL。信息提取器根据得到的 URL 在网络上抓取相关网页，但是网络上存在很多内容相同或者相似的数据，如果不加以筛选地进行无条件搜集，那么将会大量浪费资源。因为本次爬虫的对象是体育新闻，所以针对相应的 URL 格式进行了一些限制，因此 URL 提取器要经过分析整理得到统一的、有效的 URL，然后存储到待下载列表中。

2.1.2 网络爬虫实现的功能

（1）遵守 REP 协议

部分网站会对网络爬虫制定一些约束和限制，即 REP 协议。该协议存放在各个网站的根目录，文件名为 Robots.txt。在爬虫抓取网站信息之前必须先访问 Robots.txt 文件来判定是否抓取该网站信息。

（2）下载网页

网络爬虫最基本的功能就是下载网页。爬虫通过向 Web 服务器发送下载网页的请求来将遵守网络机器人排斥协议的网页下载到本地。

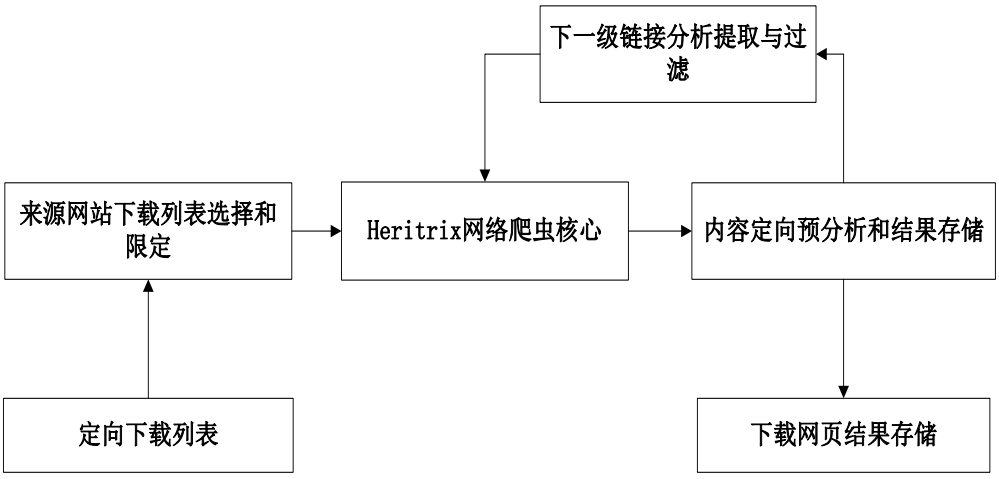
（3）提取 URL

网络爬虫采集的网页是以 HTML 网页的形式存储的，为了获取文件中的关联 URL，网络爬虫必须对页面中的链接进行提取和整理，得到待下载的下一级完整链接。

（4）过滤 URL

URL 是有 Web 协议和域名系统组成的，URL 可以是基于不同 Web 协议的，而本系统中只采集基于 HTTP 协议的网页，所以网络爬虫要具有对 URL 过滤的功能，为了增加 URL 的过滤功能，本系统也增加了相应的类对 URL 进行过滤。

本系统爬虫体系结构图如图 1.1 所示。



2.1.3 Heritrix 中部分功能

(1) 类说明

类名	用途
SportsNewsExtractor	1. 过滤 URL，根据不同的正则表达式对不同网站的体育新闻 URL 进行过滤； 2. 限制 URL 的递归深度；
SportsFrontierScheduler	1. 提取 URL，在首次得到 URL 时对其进行第一次简单过滤；
GetSportsHotFrontierScheduler	1. 通过二次访问网页，得到 JS 变量，获取网页热度。
评价： 由于对类的限制，抓取的 html 的结果总体效果不错，爬取新闻的精确度较高；但是由于网速和 URL 限制，爬虫的速度比较低，	

(2)相应网站的正则表达式匹配

```
PATTERN_SINA_NEWS---新浪网
  http://sports.sina.com.cn/[\\w]+/\\d{4}-\\d{2}-\\d{2}+/[\\d]+.shtml
PATTERN_QQ_NEWS ---腾讯新闻
  http://sports.qq.com/[\\w]+/[\\d]+/[\\d]+.shtm
PATTERN_163 ---网易新闻
  http://sports.163.com/[\\d]+/[\\d]+/[\\d]+/[\\w]+.html
```

(3)新闻网站的评论数获取

不同的网站上，js 中的变量名称不同，但是其获取评论数机制是相同

的。对于网易新闻，它的 js 变量 replyCount 形式如下：

```
function() {
    var replyCount = 62,
            totalCount = 355,
            threadId = "ADDMLJD800051CA1",
            boardId = "sports_nba_bbs",
            host = tieChannel,
            tId = tId;
```

(4)爬虫工作截图

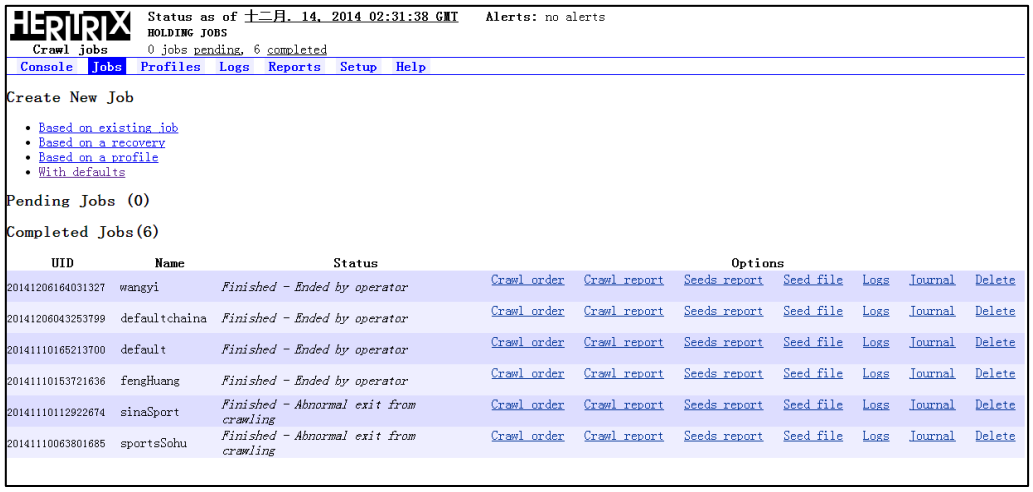


图2-1　爬虫完成任务截图

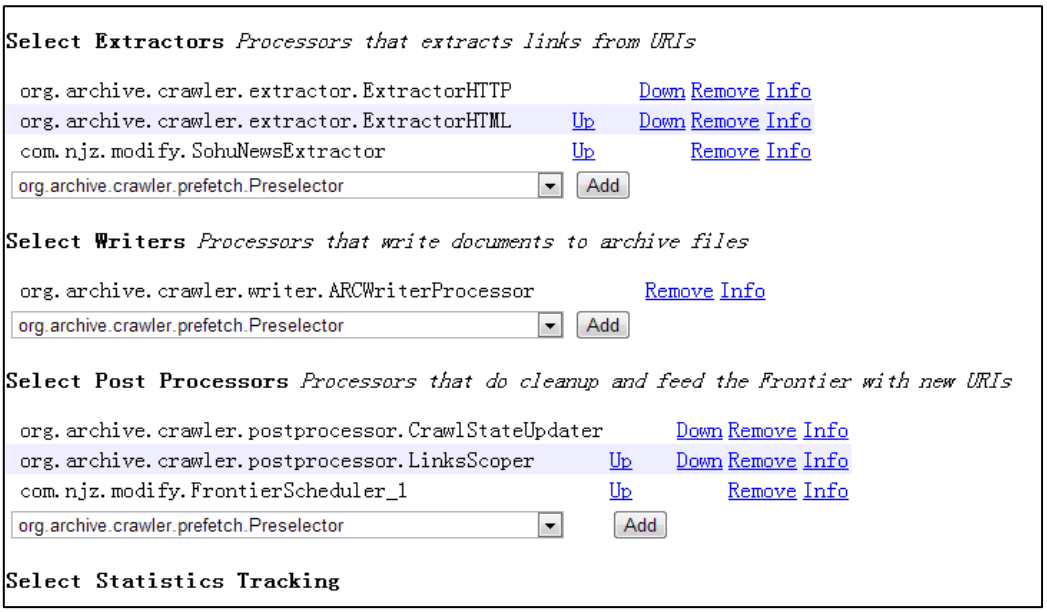


图2-2 爬虫任务的配置

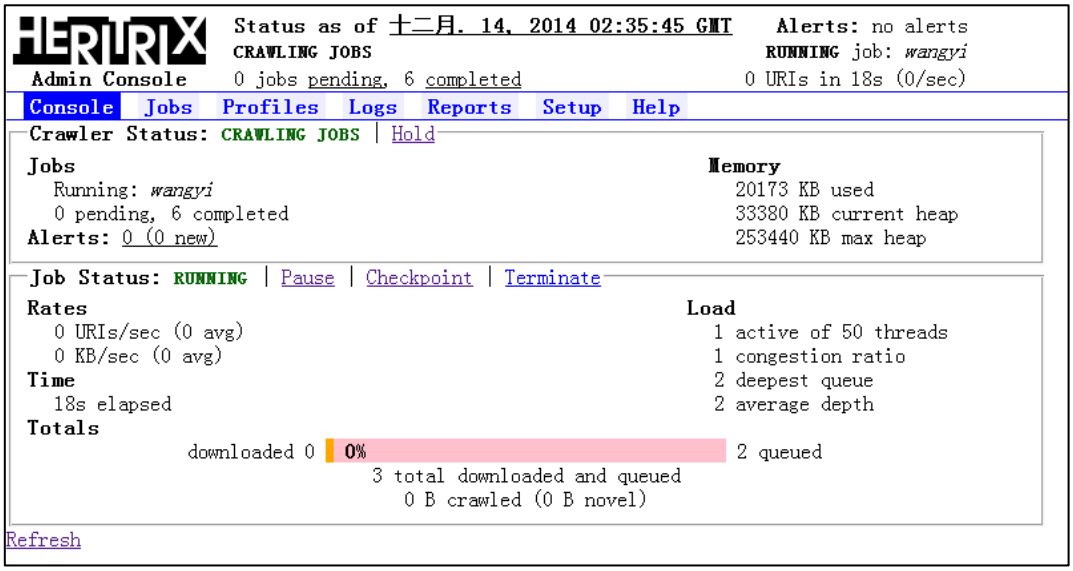


图2-3 执行爬虫任务截图

2.2 爬取数据统计

网站	网站类别代码 Type	新闻数量
新浪体育	2	57378
腾讯体育	3	26405
网易体育	4	16496
总计：	100279 条	

2.3 网页数据信息抽取

使用 jsoup 将爬虫抓取的数据解析出标题、关键词、摘要、时间等信息后，写入到数据库中。

类名	功能
GetInforFormHtml.java	处理 html 文件，分析得到数据
HtmlDetailManager.java	将详细数据写入到数据库中
问题：解析的时候的问题：主要是网页的结构的变化，以网易为例，从 06 年到 14 年，网页中标签属性和 id 的变化次数大概为 8 次，几乎每年都会发	

生若干变化。时间格式，有的时候也会发生变化。这些变化都是不可提前预知的，因此在一定程度上影响了数据的获取。同时在网页中还有部分网页是“特殊网页”，比如视频，图片等，这些也需要在解析网站的时候特别判断。

具体表结构：

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	允许空值 (	
id	int	11	0	☐	1
type	int	11	0	☑	
title	varchar	128	0	☐	
url	varchar	256	0	☑	
time	timestamp	0	0	☑	
des	varchar	128	0	☑	
body	longtext	0	0	☑	
hot	int	11	0	☑	
key	varchar	64	0	☑	

图 2-4　网页数据解析后存储表结构

id	文档 id
type	网站类别
title	新闻标题
url	新闻 url
time	新闻发布时间（年月日）
des	新闻摘要
body	新闻主体
hot	新闻热度（评论数和参与人数）
key	新闻关键词列表

表 2-1 htmls 表各个列描述

id	type	title	url	time	des	body	hot	key
1	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第7场数据	斯旺西卡迪夫城【竞彩	87	胜负彩第140
2	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第8场数据	纽伦堡拜仁(官方微博	99	胜负彩第140
3	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第9场数据	法兰克福布伦瑞克【竞	7	胜负彩第140
4	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第11场数	沃尔夫斯堡美因兹【竞	92	胜负彩第140
5	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第10场数	弗赖堡霍芬海姆【竞彩	4705	胜负彩第140
6	4	周韵双色球140	http://sports.qq.c	2014-02-08	周韵双色球14013期：关注	第14012期双色球红球	23	周韵双色球1
7	4	[小米稀饭]双色	http://sports.qq.c	2014-02-08	[小米稀饭]双色球14013期预	双色球2014012期开奖	930	[小米稀饭]双
8	4	胜负彩第14016	http://sports.qq.c	2014-02-08	胜负彩第14016期第12场数	不来梅多特蒙德【竞彩	10154	胜负彩第140
9	4	周韵足彩分析：	http://sports.qq.c	2014-02-08	周韵足彩分析：蓝军有望大	本周第14016期足球胜	22	周韵足彩分析
10	4	胜负彩14016期	http://sports.qq.c	2014-02-08	胜负彩14016期澳盘02月08	【竞彩热销 双色球2元	1018	胜负彩14016

图 2-5　数据示例

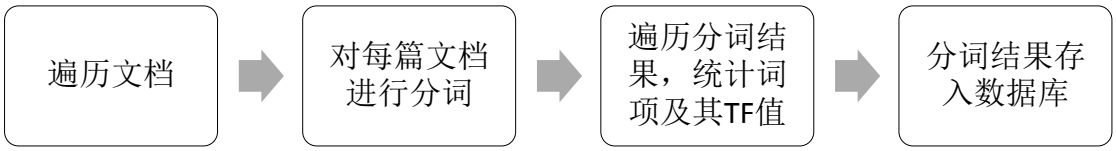
3. 分词

本文基于开源软件IKAnalyzer进行分词，分词结果存储到数据库中。

3.1 IKAnalyzer 简介

IK Analyzer是一个开源的，基于java语言开发的轻量级的中文分词工具包。它是以开源项目Luence为应用主体的，结合词典分词和文法分析算法的中文分词组件。IK实现了简单的分词歧义排除算法，标志着IK分词器从单纯的词典分词向模拟语义分词衍化。

分词流程：



关键API说明：

org.wltea.analyzer.core.IKSegmenter

这是IK分词器的核心类。它是独立于Lucene的Java分词器实现。

org.wltea.analyzer.core.Lexeme

这是IK分词器的语义单元对象，相当于Lucene中的Token词元对象。由于IK被设计为独立于Lucene的Java分词器实现，因此它需要Lexeme来代表分词的结果。

WordSegment

对文档进行分词，返回词项列表及对应的词项频率。

3.2 关键问题及数据库 Sharding 技术

由于Web网页海量特性，在数据爬取、分词以及索引构建阶段均会产生大量的数据，如果把这些数据不加处理的直接存储在数据库中，则会大大降低数据库性能，查询和更新数据均需要大量时间，而且如果数据量继续增大，不易于扩展。

对此，我们采用数据库Sharding技术提高数据库扩展性和并发访问能力。

Sharding即按一定规则对数据库进行拆分，是一种提高数据库扩展性的解决方案。本文中，我们采用对ID进行Hash的规则，对较大的数据表（千万级）进行拆分，对多个表进行并发访问，并对上层应用提供透明的接口，以提高数据库性能及扩展能力。

3.3 分词结果存放：

boydterm 和 otherterm 表的结构如图 3.1，分别存放新闻主体、除了新闻主体外的新闻标题和摘要、关键词的分词的结果。

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点
term_id	bigint	20	0
term	varchar	150	0
document_id	bigint	20	0
tf	bigint	20	0

图 3.1 bodyterm 和 otherterm 表结构

4. 倒排索引的构建

4.1 倒排索引的构建

类名	功能
Index	索引类： 1. 索引类中存放文档 ID、文档热度、文档时间、tfidf 时间 2. Index 类采用序列化操作，能够提高查询速度
IndexConstruct	1. 索引的构建由三个层次组成，优先级由高到底。 2. 查询度较高的词（由历史记录获得）的倒排索引为一级索引； 3. 由新闻的标题 title，关键词 keywords、摘要 description 生成的索引为二级索引； 4. 由新闻 body 得到的三级索引最大，查询时间对比一、二级索引时间较长。 5. 构建索引时候通过一些接口尽量减少数据库连接，从而提升构建索引时间。
问题：本系统的倒排索引构建没有采用已有的包，而是自己构建后，将索引结果利用数据库存储的，最后在查询时候直接查找数据库即可。一二级索引的构建速度很快，时间约 2-4min；三级索引针对数据量较大，时间约为 15-18min。通过数据库模式和提交方式的修改也对速度有一定提升。	

倒排索引存放在数据库中，索引字段采用 blob 类型存储，也能在一定程度上，提高查询的速度。具体的数据库结构和具体的数据如下：

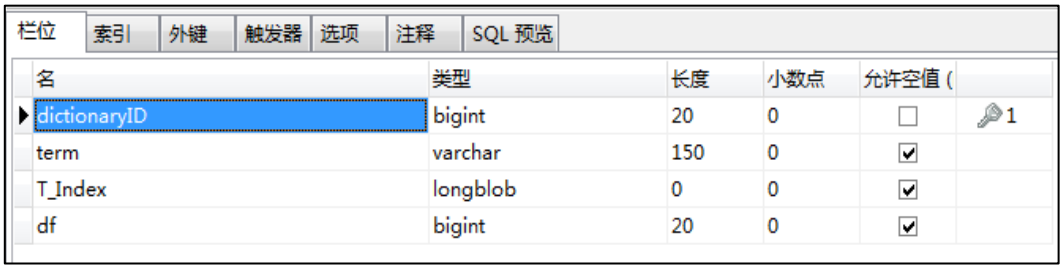


图 4-1　倒排索引表结构

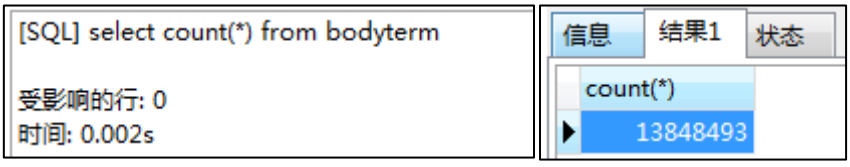
dictionaryID	文档 id
term	词项
T_index	倒排索引内容（blob）
df	文档频率

图 4-2　倒排索引表各个域

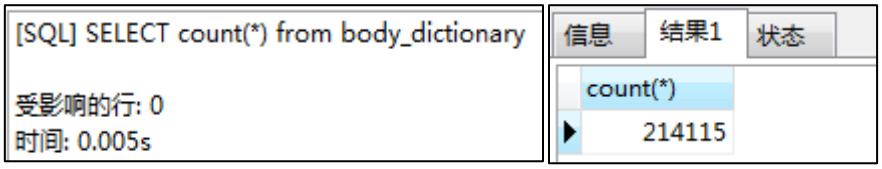
图 4-3　倒排索引实例

由于本系统采用的是 mySql，为了加快搜索的速度，也将系统的数据库引擎切换到了 MyISAM 模式，同时对 term 属性构建了 Btree 的索引结构。系统实现了对所有 10 万条爬虫数据的分词，分词后合并前的结果为 1400 万条左右，合并后生成的倒排索引数量为 21 万条左右，对其中一个词项进行检索的时间约为 0.005s 左右。具体的信息见下图。

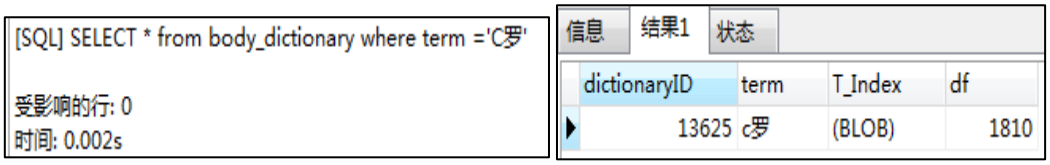
（1）根据新闻 body 初始分词结果数量：



（2）根据分词结果构建索引的数量：



（3）从三级索引（body 索引）中得到具体信息的时间：



在查询过程中，按照级别又高到低查询索引，同时设定查询阈值 Threshold_pageNum，当在高级别的索引中查询到的文档数量超过阈值的时候，低级别的索引不再查询，此时只有当用户请求更多的文档的时候才进行低级别索引的查询操作。这样能同时保证查询的准确度和查询速度。

5.查询结果的评分排序

类名	功能
ResultScored	得分类： 1. 得分类中存放文档 ID、tf-idf 得分、文档时间得分、热度得分、查询出现次数得分 4 个分项得分; 2. 构造方法中根据得分公式为查中的文档计算总得分; 3. 得分公式中的权重因子可根据实际情况调整，同时最后得分考虑文档长度问题;
ScoreProcess	1. 得分处理时，根据实际查询请求返回得分最高的 top100 篇文档，同时将检索到的文档总数返回; 2. 当二级索引查询结果不够 100 篇，用三级索引补充;
ReadIndexScored	1. 根据相应的词项读取相应的倒排记录; 2. 将得分计算后的 list 返回到前台; 3. 考虑不同的词项查询到的相同的文档 id，即文档的出现次数，并在得分计算中作为参考因素;
总结：本系统的读取倒排索引和检索到文档的打分排序没有采用已有的包，在数据库中首先查询一二级索引，得到相应文档，并计算得分，如果返回结果不足 topK 篇，再去查询三级索引，这样能充分提高查询速度。 得分公式中，可以设置不同因素的权值以得到最优的排序结果。	

5.1 排序参考的因素如下：

（1） 文档中出现的查询中的词项数量;
（2） 文档中出现的查询中的词的 tf-idf 信息;
（3） 新闻时间的新旧程度;
（4） 新闻热度;
（5） 查询得到的文档长度信息（长度信息为文档包含词项数量）;

检索结果得分公式如下：

$$\text{Score} = \frac{(\text{fac1} \times \lg \sum hot) + (\text{fac2} \times \sum tfidf) + (\text{fac3} \times \text{count})\text{fac4} \times \lg\left(\frac{10}{\lg time}\right)}{\sqrt[3]{|D|}}$$

公式中参数说明：

(1)文档 D 长度为|D|（使用文档 c 中词的个数来代表）

(2)查询 Q 长度为|Q|（同样也是分词后的词的数量）

(3)文档和查询共有词：count = | D∩Q |

(4)fac1~fac4 为四个分量的权值分配

对于文档的评分，需要考虑到文档的长度，但是由于文档的长度一般都在 200-300 词左右，对最后得分的影响不大，所以得分计算中对|D|进行三次方根运算。

在整个检索过程中，只要| D∩Q |的值大于 0 的文档都会返回，如果返回的数量较大，去除那些共有词出现次数小于(count/3+1)的文档。进行一次筛选；筛选后的结果进行评分，最后返回到前台的文档为得分最高的前 K 篇结果（目前 K=100）。

6.向量空间索引（正排索引）和层次聚类

6.1 正排索引

将文档看成向量，每个维度都是词项的 tfidf 值，并且最整个向量最后做归一化处理。

BuildDocVectorTable.java	实现构建每个文档的向量，并写入到 docvector 表中
遇到的问题：数据量比较大的时候（十万篇文档的分词结果大概有一千五百万个分词结果），由于采取每次都是直接查询数据库，使得处理速度较慢，后来修改成将数据库数据遍历一遍，在内存中统计结果，并且在写数据库的时候每 1000 个向量做一次批处理，使得速度提升较大，构建一次的时间由原来的数小时缩短为 11 分钟。	

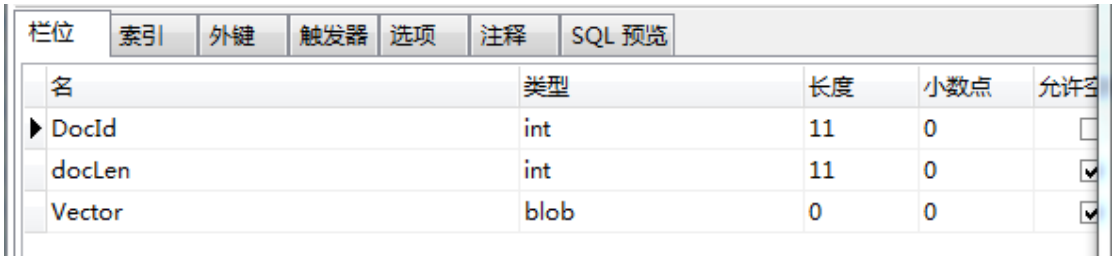


图 6-1　正排索引表结构

DocId	文档 id
docLen	文档中的词项数
Vector	(词项 id,tfidf)构建的 Map 的对象

图 6-2　正排索引表列名称

DocId	docLen	Vector
1	1	(BLOB)
2	8	(BLOB)
3	6	(BLOB)
4	5	(BLOB)
5	5	(BLOB)
6	63	(BLOB)
7	115	(BLOB)
8	6	(BLOB)
9	211	(BLOB)

图 6-3 正排索引表实例

6.2 采用组平均凝聚式聚类算法

输入： 查询返回的文档列表以及文档的得分

输出： 聚类后的按照得分由高到低进行排序的簇列表

处理： 设定了聚类停止阈值：

第一个阈值是簇的数量 Math.sqrt(DocNum)

第二个是相似度大小 sim_threshold

上面两个阈值有一个不满足就结束聚类，经过测试，sim_threshold 取值为 0.5 的时候效果不怎么理想，几乎每个簇都是只有一个到两篇文档，当为 0.2 的大部分时候返回的聚类的结果也不是很理想，一千篇文档聚出 900 多个类。 当继续调低阈值的时候，簇的数量不断减少，当小于 0.1 的时候簇的数量急剧减小，平均情况下 1000 篇文档大概聚出不到 100 篇文。为了使得绝大部分聚类后的簇的数量限制在较小的水平上，我们选择 0.02 作为阈值，500 篇文档，簇数量大部分小于 50。

降维：给定阈值 termThreshold，将小于该值的维度去掉。

当不处理的时候，簇的质心的维度一般维度较高，有的可以达到几千维，当阈值设定为 0.01，降低为一般不超过 200-300 维度，当设定为 0.1 的时候结果发现聚类后的簇的质心的维度一般最大不超过 200 维。 由于一篇新闻一般在 200-300 个词汇这里选择 0.01 作为阈值。

代码：

类名	用途
BuildDocVectorTable.java	1. 构建向量空间 2. 将归一化向量写入到数据库中

ClusterDetail.java	存放一个簇的详细信息 1. 记录簇的标签（这里使用了距离质心最近的文档的标题） 2. 得分 3. 簇中文档列表 4. 簇的质心
Clustering_GAAC.java	具体实现聚类的细节： 1. 计算文档相似度 2. 计算簇间的相似度 3. 合并簇 4. 对簇进行排序 5. 计算簇的质心，并选择距离质心最近的文档的标题作为簇的标签
评价： 总体感觉聚类效果一般，可能是由于检索的结果本身不像有歧义的词那样会产生类别比较明显的结果。	

7. 检索后期处理与检索结果展示

7.1 系统框架

查询的页面接口通过网站访问的形式实现。

检索引擎网站的实现相关技术如下

功能	相关技术	优点
整体框架	SpringSide	成熟、易使用
数据库	Mysql	灵活、方便、大容量
数据访问	Hibernate	易操作、简洁、接口灵活
页面	JSP	移植性好、易操作
页面数据处理	Jquery	成熟、封装性好
页面风格	Bootstrap	风格统一、简洁

使用 MVC 分层次架构：在实体层定义实体对象；持久层进行数据访问控制；服务层调用分词、检索、相关度评分、聚类的接口实现逻辑业务；视图层与前台 jsp 或 ajax 进行数据交互。查询序列图如图 7-1 。

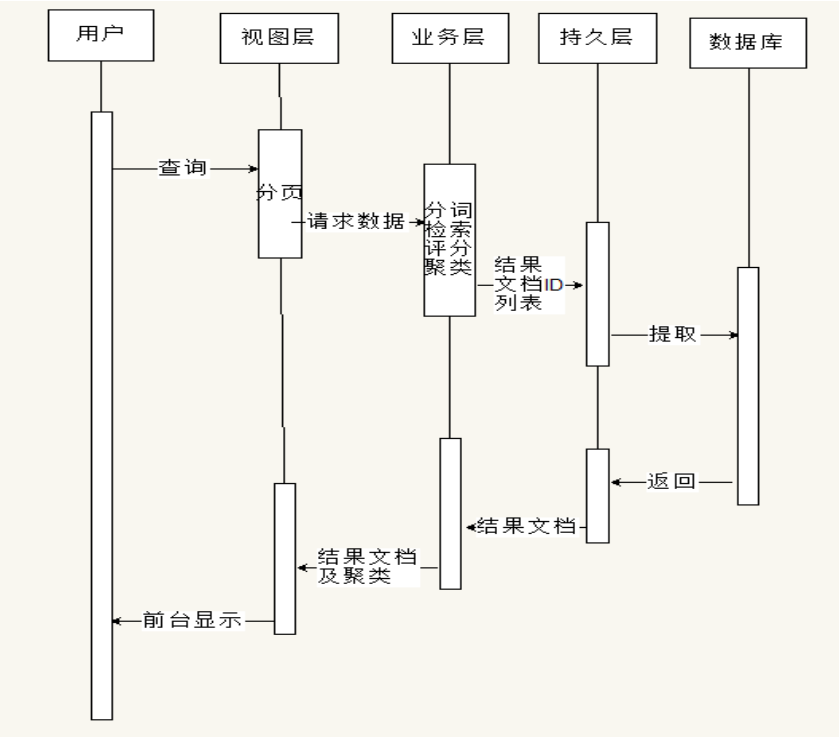


图 7.1 查询序列图

7.2 自动补齐

用户输入检索查询时自动提示补齐。

实现原理:服务器维护检索记录表，用户的所有检索查询会被记录，当用户检索时将根据已输入的字符进行自动提示补齐，供用户选择填充。

实现：前台 jquery 使用 typeahead 函数监测用户输入，通过 ajax 访问后台查询与已输入文字匹配的相关列表，得到响应后展示在输入框下。补齐提示列表以记录中相关查询的计数排序。

查询记录表：squery

列	数据类型	长度	备注
Id	int	11	标识
squery	varchar	32	查询记录
count	int	11	查询计数

Ajax 请求：

```
$('#search_LIKE_title').typeahead({
        source: function (query, process) {
            //删除左右两端的空格
            query = query.replace(/(^\s*)|(\s*$)/g,"");
            $.ajax({
                type: 'post',
                url: '/gudu/api/v1/query/auto/'+query,
                dataType: 'json',
                contentType:"charset=utf-8",
                success: function(obj) {
                    var num = obj.length;
                    if(num>0){
                        process(obj);
                    }else{
                        process("[]");
                    }
                },
                error: function() {
```

自动补齐的效果如图 7-1。

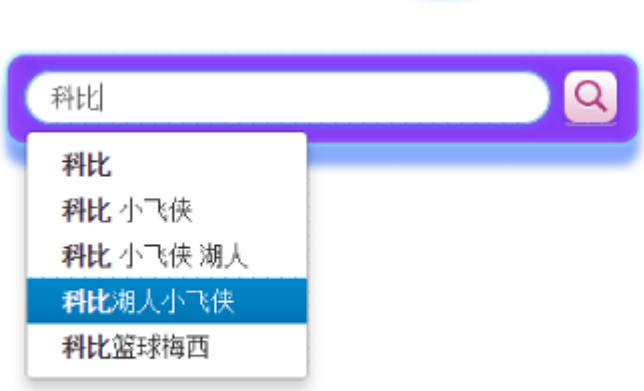


图 7-1 自动补齐

7.2 展示与分页

体育新闻由标题、关键词、发表时间、参与评论人数、摘要、内容快照、对应 url 几部分组成，点击对应新闻访问对应网站的改新闻页面。

实现原理：检索结果以对象列表形式返回，前台循环获取列表，通过分页形式显示。

实现：为保证良好的显示效果我们设定每一页显示 8 条数据。检索请求发出到后台时同时发出页面请求。检索到对应文档 Id 列表（以相关度评分排序）后按照页码请求提取对应位置的文档，同时将当前页码、总页数、结果文档数、当前页文档列表保存到一个对象中，前台按照这些数据显示新闻列表以及正确的分页响应。

展示与分页效果如图 6.2 。

图 7-2 结果展示与分页

7.3 显示快照

以列表形式展示检索结果体育新闻的标题、关键字、摘要等信息，当鼠标移动到标题时浮动显示对应新闻快照。

实现原理：查询结果返回所有内容，但首先仅展示简要信息，鼠标移动触发事件，动态显示快照。

实现：使用 jquery 监听标题的 hover 事件，触发后获取需要展示的内容，使用 popover 生成新的 div 层动态将快照内容显示在对应位置。

快照显示效果如图 6.3

图 7-3 快照与高亮

7.4 关键字高亮（snippet）

提示检索查询结果中与查询相关的关键字，并高亮显示。

实现原理：将检索查询的分词结果同时返回到前台，针对每一个分词项，遍历检索结果的标题，关键字，摘要，快照，将其中包含的所有分词关键字使用对应风格的标签进行替换，达到高亮显示的目的。

实现：首先设计高亮文本的风格 css 类。前台在获取检索结果数据时，同时获取分词结

果列表，javascript 获取这些数据，循环遍历每个分词项，分别对标题等内容使用正则表达式找到内容中的对应分词关键字，在关键词前后添加标签，替换后的内容再次显示在前台。

高亮显示 js 代码如图 6-4，高亮显示效果如图 6-3。

```
$('.title').each(function(){
        var key = $("#search_LIKE_title").attr("last");
        var keys = key.split(" ");

        var value = $(this).attr("data-content");
        var first = value.indexOf(key[0]);
        value=value.substring(first-20,first+600);
        value='...'+value+'...';
        for(var i = 0;i < keys.length; i++){
            value =
value.replace(eval("/"+keys[i]+"/gi"),'<span
class="target">'+keys[i]+"</span>");
        }
             $(this).attr('data-content',value);
    });
```

图 6-4 高亮 js 代码

7.5 搜索推荐

用户得到检索结果后通过搜索推荐，继续搜索其他相关的关键词。

实现原理：重复使用搜索记录，分别查找到当前用户查询分词的对应记录，将不同分词查找结果整合后以查询热度（被搜索次数）排序展示。点击对应推荐可以实现直接快速查询。

搜索推荐效果如图 7-5。

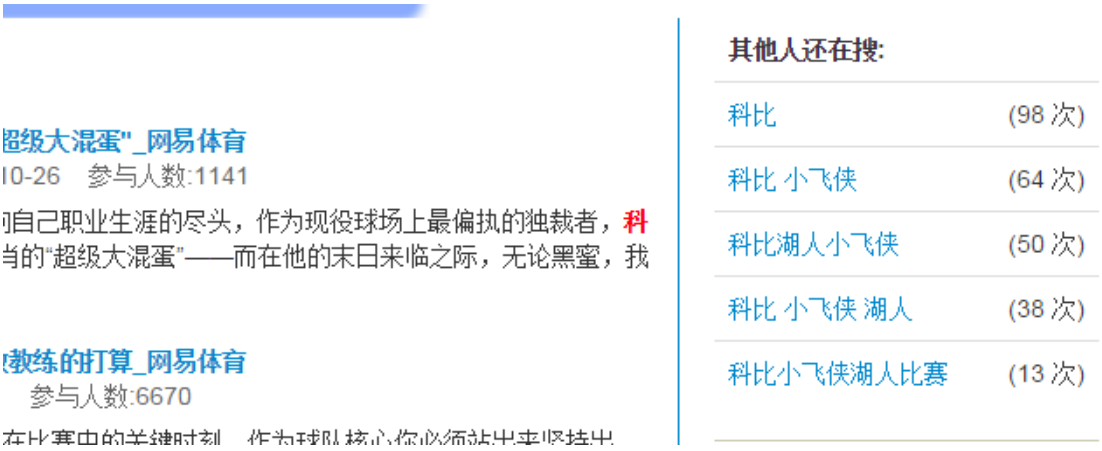


图 7-5 搜索推荐

7.6 聚类显示

对检索结果聚类，相似文档的簇以中心文档代表,显示在页面。

实现原理：构建聚类簇对象，包含中心文档对象、簇名（中心文档标题）、簇内文档列表、簇内文档数。簇内文档列表以文档与查询相似度评分排序，簇的列表以簇内文档评分之和降序排列，并显示前 8 个簇。

聚类显示如图 7-6。

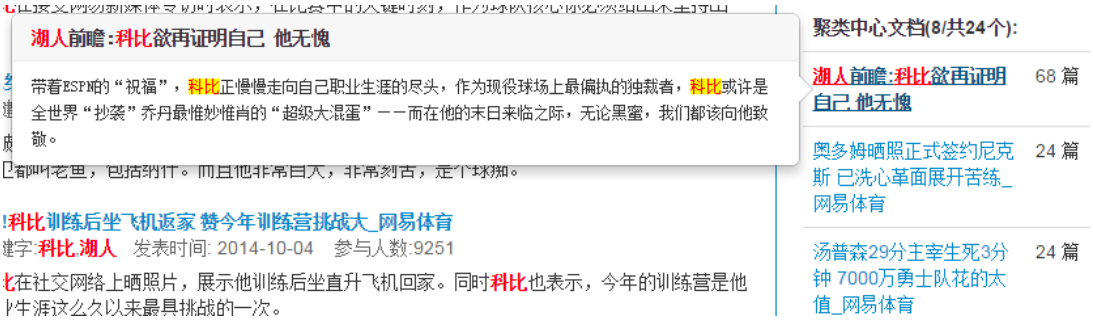


图 7-6 聚类显示

7.7 异常处理

①对于空查询的处理。使用 jquery 的验证机制，在提交查询时检测输入是否为空。如图 7-7 。

图 7-7 关键词为空

②对于长查询的处理。使用 jquery 的验证机制，在提交查询时检测输入长度是否超过设定，设定值为 32 个字符长度。如图 7-8 。

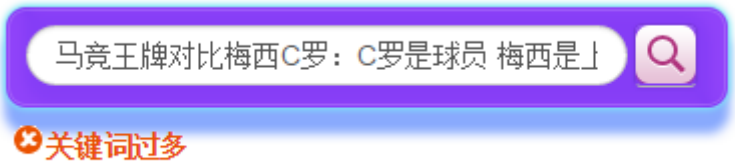


图 7-8 关键词过多

③对于通过 url 直接进行空值非正常查询的异常，将直接返回到首页。

④对于没有相关文档的，响应提示“未找到相关文档”，如图 7-9 。

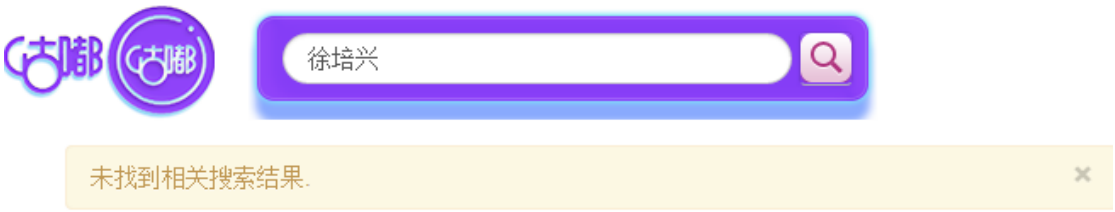


图 7-9 未找到相关文档

7.8 多样查询结果展示

①多关键词查询，示例“科比湖人小飞侠”。如图 7-10 。

图 7-10 查询示例①

②数字中文混合查询，示例“76 人”

③ 中英文混合查询，示例“马竞王牌对比梅西 C 罗：C 罗是球员”

8. 系统的评价

通过上面的检索测试，说明我们的检索系统具有较高的准确率和召回率，总体效果良好。

9．经验总结

该项目有四人合力协作完成，锻炼了大家的团队协作能力，加深了对检索系统的了解，基本实现了体育新闻的检索，能够较好的根据查询返回结果。不足之处也很多，由于时间紧迫，在排序函数的设计上还可以更好。还可以通过其他方式，比如分布式来加快检索过程。