信息检索的概述

2018年9月17日 15:22

信息过载

信息大爆炸

youtube 一分钟上传400小时视频

苹果用户一分钟下载51000个应用

google一分钟翻译69500000个单词

siri一分钟回答错9万个问题

总结:信息越来越多,如何迅速的定位我们需要的信息,抛弃无效的信息(搜索)

提高检索效率和生成新的检索技术,是关键;

传统检索方式

操作系统文件检索

直接通过操作系统搜索磁盘,全表遍历

缺点:慢

文件的内存检索

word,excel,onenote,检索速度还行,文件一旦打开,文件的全部内容都加载到了内存;上限512M;

缺点:数据量少

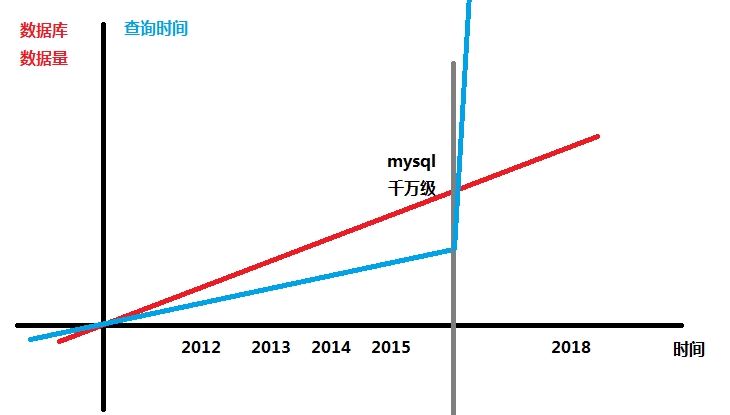
数据库检索

1. like模糊查询才可以完成目前大部分检索的需求"%关键字%" "关键字%"(符合数据库**最**

**左特性**查询,还是使用索引)

2 索引优化(建立各种各样不同索引结构的索引文件,满足不同的查询条件的需求) 表格是有数据量的上限的

查询效率在临界点时,查询时间无限增加

缺点:数据量受限制

分区 day06all 的第 30页

信息检索系统

一个完成的信息检索系统包括三部分

1 信息采集

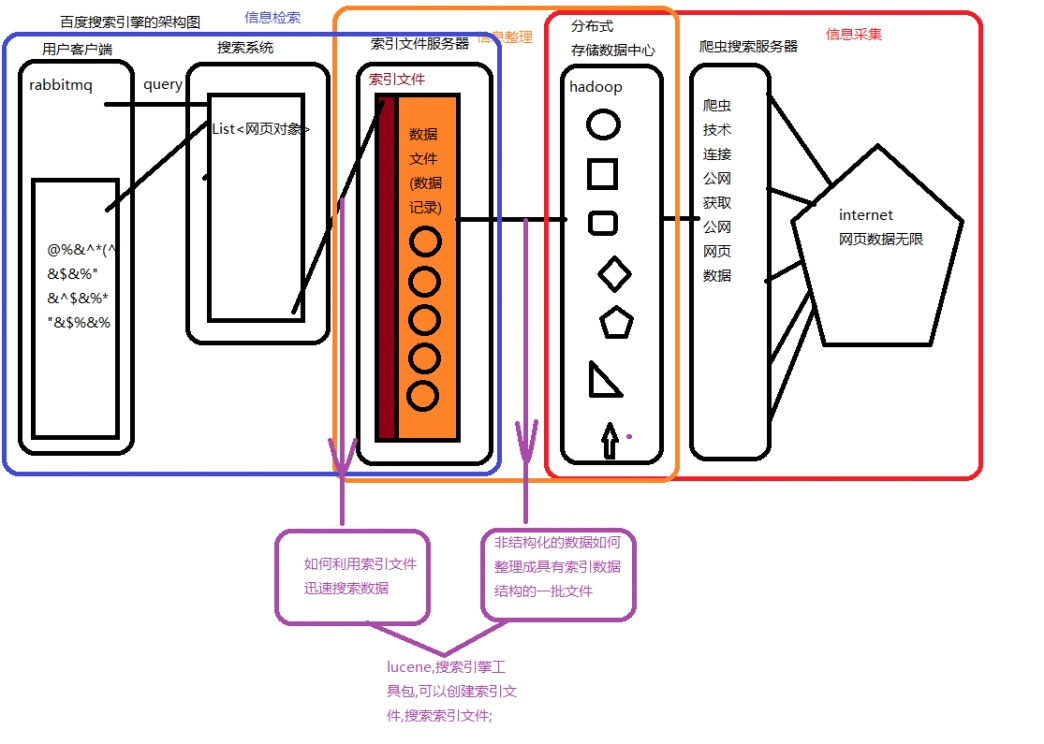
搜索引擎,通过网络爬虫技术,将公网的各个服务器的数据按照一定的规则,从万维网获取数据,单位超过亿级别,数据非常庞大;

2 信息整理

爬取的这些内容,非结构化数据,无法完成任何快速搜索技术的搭建,一定经过处理,整理成有一定数据结构的数据文件才可以进行搜索的服务提供(创建索引的过程)

3 接收查询

用户向检索系统发出查询请求,信息检索系统接收这个参数,在数据中获取对应的结果,返回用户;

全文检索的倒排索引计算方式(网页)

数据源:数据库,公网网页,各种行业中的数据,不同位置,不同系统,不同软件存储

倒排索引:构成全文检索技术的创建索引文件的核心计算方法;

网页数据为例(源数据)

文章1: 马蓉和王宝强是否感情稳定

分区 day06all 的第 31页

文章1: 马蓉和王宝强是否感情稳定

文章2: 狗仔队全程紧盯马蓉和王宝强的生活

1. 分词计算(一句话的分词计算,就是讲当前字符串拆分成具有最小意义的词)

文章1:(马蓉,1(频率),1(文章id)),(王宝,1,1),(王宝强,1,1),(是否,1,1),(感情,1,1)

文章2:(狗仔,1,2),(全程,1,2),(紧盯,1,2),(马蓉,1,2),(王宝,1,2),(王宝强,1,2)

1. 分词合并

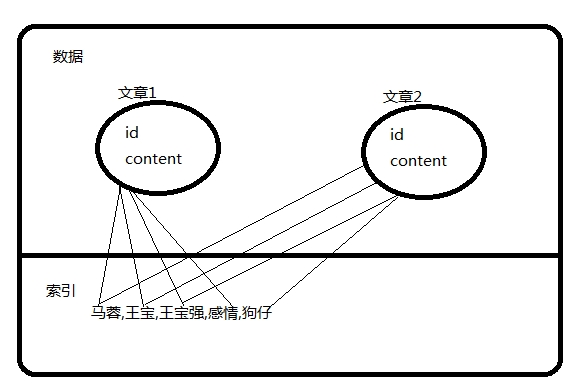
(马蓉,[2],[1,2]) (王宝,[2],[1,2]) (王宝强,[2],[1,2]),(狗仔,1,2),(全程,1,2),(紧盯,1,2),(是否,1,1),(感情,1,1)

3生成索引文件

将分词合并的结构整理成二进制格式,保存到索引数据当中.

文章1,文章2,生成对应的结构的对象(document文档对象,文档对象时数据结构中的一个数据单位)

4 结构

最终的索引文件,经过对源数据中的数据分词计算,对象封装过程整合成的上图结构,一个分词结果直接指向一批文档集合(document的集合)

分区 day06all 的第 32页

Lucene

2018年9月17日 16:34

介绍

历史: 创始人Doug Cutting(hadoop);

lucene:是搜索引擎的工具包

基于java编写的

特点

稳定,索引性能高(倒排索引)

现代硬盘每小时索引150G数据(索引:创建索引文件)

java运行时要求的堆内存1MB

增量索引和批量索引一样快

提供排名

支持多种主流的搜索功能:短语,通配符,模糊,近似,范围,此项,多 义词等查询功能

lucene的分词计算

分词:分词的过程就是将源数据中某段字符串根据人类使用语言的习惯进行词汇的切分.设计到语言不同,设计到语言的发展.

添加lucene的测试依赖(课前资料可以获取)

/\*

\* Lucene本身携带多种分词器,但是无法满足语言不同,语言发展的要求

\* 定义了Analyzer的接口,只要开发人员实现了这个接口就可以按照自定义

\* 的规则完成一些分词的计算,比如中文有很多分词器(分词器的jar包非常多) \*/

//编写一个静态方法,利用传入的不同分词器对象,对同一个字符串进行分词计算

//将分词结果打印进行对比

public static void printAnalyzer(Analyzer analyzer,String str) throws Exception{

//将字符串传话成流对象String流

StringReader reader=new StringReader(str);

//analyzer底层实现,是通过tokenStream来完成的,根据实现的不同实现 类中的tokenStream

//方法,对数据流进行分词,属性计算.

分区 day06all 的第 33页

//方法,对数据流进行分词,属性计算.

TokenStream tokenStream = analyzer.tokenStream("test", reader); tokenStream.reset();//重置属性,从头开始

//从分词token流中获取词项属性,词项:分词的每个最小意义的词,就是一

个词项

CharTermAttribute attribute

= tokenStream.getAttribute(CharTermAttribute.class);

//while循环获取所有的词项输出

while(tokenStream.incrementToken()){

System.out.println(attribute.toString());

}

}

@Test

public void run() throws Exception{

//准备lucene的不同分词器对象

Analyzer a1=new SmartChineseAnalyzer();//智能中文分词器Analyzer a2=new WhitespaceAnalyzer();

Analyzer a3=new SimpleAnalyzer();

String str="我是你爸爸,你最近还好吗,我能和你见个面吗"; System.out.println("智能分词器\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); LucenTokenTest.printAnalyzer(a1, str);

System.out.println("空格分词器\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); LucenTokenTest.printAnalyzer(a2, str);

System.out.println("简单分词器\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); LucenTokenTest.printAnalyzer(a3, str);

}

对于文本数据,进行分词切分的过程是非常关键的,能否有效,高效的查询到数据,在底层取决于分词计算的精准度的.

IK分词器

2018年9月18日 8:51

ik分词器

常见的一种中文分词器，可以智能分析中文词汇，可以做到

扩展词汇，停用（语气词，助词类似这种无关紧要的词语可 以在停用词典中禁止分词）

使用ik分词器

1. 依赖，maven中央库没有dependency；手动添加依赖，利用maven将依赖添加到本地库，也可以在测试时将jar包放到工

程中

手动添加--build-path jar包，**课前资料--lucene--IK分词器--jar**

**包**

1. 4.0版本之后的IK分词器为了实现解耦和自定义流的属性等 问题，需要自行完成analyzer,tokenizer等代码；

可以使用已经编写好的Ikanalyzer **课前资料--lucene--IK实现类**

1. 测试使用

IK分词器的扩展词典和停用词典设置

需要配置文件制定扩展词典和停用词典

IKAnalyzer.cfg.xml

<properties>

<comment>IK Analyzer 扩展配置</comment> <!--用户可以在这里配置自己的扩展字典

<entry key="ext\_dict">ext.dic;</entry>

-->

<!--用户可以在这里配置自己的扩展停止词字典--> <entry key="ext\_stopwords">

分区 day07all 的第 1页

<entry key="ext\_stopwords">

stopword.dic;</entry>

</properties>

ext.dic扩展词典和stopwords.dic停用词典，必须在同一个文 件夹下，当前工程的加载文件默认相对路径（maven工程

中，添加到target/classes）

注意：如果想要配置文件的扩展词典和停用词典生效

1 扩展词典停用词典的编码格式要和字符串的编码格式 一致

1. 一定要被工程加载（被加载成功的话会答应ik标识）

分区 day07all 的第 2页

分区 day07all 的第 3页

lucene创建索引

2018年9月18日 8:51

创建索引：创建索引是根据**数据源**读取的数据，进行整理， 完成倒排索引后输出成索引文件，只有数据库有数据源；根

据商品表格中的内容，完成当前需要的索引文件的创建；

索引创建中的一些概念

**文档**：检索结果的对象封装（数据单位），可以封装一个网 页内容，也可以封装一条商品的数据记录

**查询：**搜索条件的封装，在查询时有各种条件，限制，都可

以在查询对象中体现，最终实现利用查询对象完成复杂的搜 索逻辑

**词项**：分词计算结果的最小意义的词

**域**：就是文档对象的一个属性，根据封装的内容不同，域可 以变动；

/\*

\* 使用lucene创建索引文件

\* 1 指定输出文件的位置，当前工程“index”文件夹 \* 2 设置索引创建时的配置对象，指定分词等环境信息 \* 3 手动创建保存的数据对象--document

\* 4 使用lucene的流，将数据输出（会计算数据的分词 创建索引）

\*/

@Test

public void createIndex() throws Exception{

//指定文件夹

分区 day07all 的第 4页

//指定文件夹

Path path=Paths.get("index");

//指定lucene格式的输出路径对象

FSDirectory directory = FSDirectory.open(path);

//生成配置对象，指定分词器

Analyzer analyzer=new IKAnalyzer6x();

IndexWriterConfig config=new

IndexWriterConfig(analyzer);

config.setOpenMode(OpenMode.CREATE);

//create表示每次创建都覆盖，append每次创建都 追加

//create\_or\_append 有就追加，没有就创建

//生成存储的数据对象，doc1商品数据为例 doc2

Document doc1=new Document();

Document doc2=new Document();

//对文档对象进行数据的添加，需要使用到field（字 段，属性）

//name表示当前域的名称

//value当前添加的域属性的值

//Store.yes在创建索引文件时，document的这个域 值会不会存在索引文件里

doc1.add(new StringField("id", "1000",

Store.YES));

doc1.add(new TextField("title"," 三星

(SAMSUNG) 860 EVO 250G 2.5英寸 SATAIII 固态 硬盘（MZ-76E250B）",Store.YES));

doc1.add(new TextField("desc","快，就是快，无 比的快",Store.YES));

//Store.yes和no的区别，StringField和TextField区

分区 day07all 的第 5页

//Store.yes和no的区别，StringField和TextField区 别

doc2.add(new StringField("id","100",Store.NO)); doc2.add(new StringField("content","我们今天是 否要去晚餐",Store.NO));

//数据来源需要读取数据，不是手动添加

//输出到索引文件创建索引

//在writer对document进行输出时，会根据配置的 分词器，进行数据的分词计算

IndexWriter writer=new

IndexWriter(directory,config);

writer.addDocument(doc1);

writer.addDocument(doc2);

writer.commit();

writer.close();

directory.close();

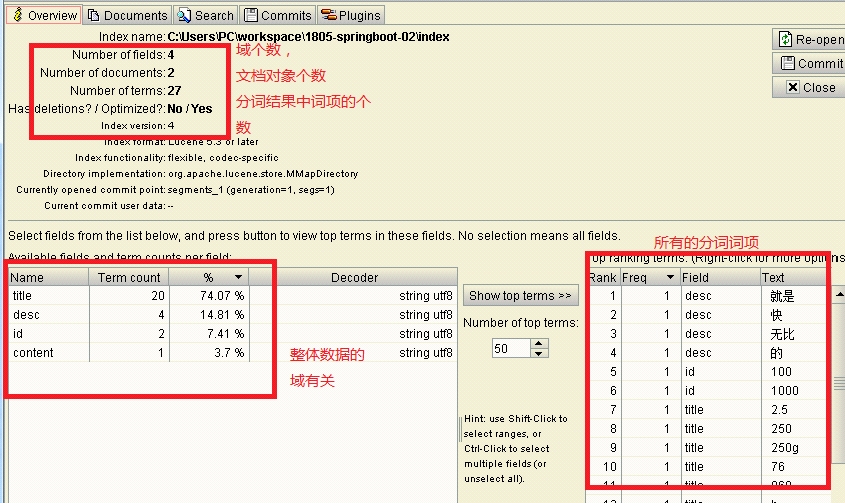
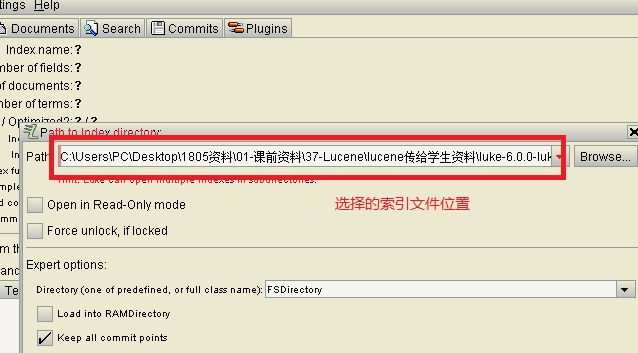
}

对于新创建的索引文件，数据结构无法使用普通的软件打开 查看，需要利用专门的工具，LUKE工具6.0.0必须和lucene的

代码版本完全一致（luke基于lucene开发编写的）

**课前资料-lucene-luke 双机bat文件即可打开luke软件**

分区 day07all 的第 6页

StringField和TextField的区别：

StringField不做分词计算；例如在某个商品的数据中，图片url 不需要分词计算，字符串的id（ksdah739klsdhsa932） TextField会做分词计算

以上2个域属性的类型，是对数据的处理类型；

其他的int long double这种类型的数据不会做索引存储

IntPoint

分区 day07all 的第 7页

IntPoint

LongPoint

DoublePoint

FloutPoint

只会在底层计算二进制后，完成搜索过程的筛选过滤，范围

查询等功能，不会存储在索引文件中

在数据中，有些属性只有数值的特性比如价钱

如果使用double，不会再索引文件中存储这个数据，但是可 以利用范围搜索将price的取值范围查询出来；

例如,price DoublePoint（“price”,56.88）,范围搜索 40<price<60,将会被查询到这个对象

如果需要对某个字段既能使用数字的查询功能，又需要存储

数据，利用StringField在存一个同名的域

Store.yes和no的区别

一个在document的索引数据对象中保存数据，一个是不保

存；在搜索时，搜索到的一批document结果集，如果域yes保 存，可以从结果集的对象中获取域的数据，no不保存，即使

搜到这个document也无法获取域的值

分区 day07all 的第 8页

lucene索引的查询

2018年9月18日 8:51

lucene搜索能力非常强大，根据分装的不同类型的查询条件对象Query,Query是一个查询条件的接口类，不同的查询结构，可以利 用不同的实现类来完成条件的封装

词项查询

多域查询

布尔查询

范围查询

前缀查询

多关键字查询

模糊查询

通配符查询

等

查询入门

可以根据分词计算结果的词项，把收集的查询条件做成

TermQuery，如果有词项并且对应指向了一批document将会把查询 结果返回；拿到的数据结果就是documents（文档对象的集合）

/\*

\* lucene查询搜索的测试

\* 1 指定路径

\* 2 封装查询对象Query

\* 3 查询获取结果集

\* 4 从结果集封装业务需要的数据

\*/

@Test

分区 day07all的第 9页

@Test

public void search() throws Exception{

//路径指定

Path path = Paths.get("index");

FSDirectory directory = FSDirectory.open(path);

//使用输入流打开索引文件

IndexReader reader =

DirectoryReader.open(directory);

//获取查询对象

IndexSearcher searcher=new IndexSearcher(reader);

//指定一个查询时可能需要的分词器对象

Analyzer analyzer=new IKAnalyzer6x();

//封装查询query对象

//name是域名称，针对哪个域进行搜索

//a,analyzer分词器

QueryParser parser=new QueryParser("title",

analyzer);

parser.setDefaultOperator(Operator.OR);

//查询条件 中华人民共和国，利用分词器对查询字符串 “中华人民共和国”分词

//中华，华人，人民，共和国，Operator.AND表示查询

条件的分词结果的词项必须同时

//存在于一批document，才可以查询到结果

Query query=parser.parse("三星英特尔");

//查询数据，获取结果集

TopDocs docs = searcher.search(query, 10);

//从topdocs中获取每个document对象

ScoreDoc[] scoreDocs = docs.scoreDocs;

分区 day07all的第 10页

ScoreDoc[] scoreDocs = docs.scoreDocs;

for (ScoreDoc scoreDoc : scoreDocs) {

Document doc=searcher.doc(scoreDoc.doc);

//获取doc中的数据

System.out.println("id:"+doc.get("id"));

System.out.println("title:"+doc.get("title"));

System.out.println("desc:"+doc.get("desc"));

System.out.println("文档评分："+scoreDoc.score);

}

}

多域查询

利用多个域的名称和数据，封装MutiFieldQuery

对象，利用这个查询对象进行数据的搜索；

@Test

public void searchMuti() throws Exception{

//路径指定

Path path = Paths.get("index");

FSDirectory directory = FSDirectory.open(path);

//使用输入流打开索引文件

IndexReader reader =

DirectoryReader.open(directory);

//获取查询对象

IndexSearcher searcher=new

IndexSearcher(reader);

//指定一个查询时可能需要的分词器对象Analyzer analyzer=new IKAnalyzer6x();

//封装查询query对象,多域查询对象

//多域的域名数组

分区 day07all的第 11页

//多域的域名数组

String[] fields={"title","content"};

MultiFieldQueryParser parser=

new MultiFieldQueryParser(fields, analyzer);

Query query=parser.parse("三星");

//查询数据，获取结果集

TopDocs docs = searcher.search(query, 10);

//从topdocs中获取每个document对象

ScoreDoc[] scoreDocs = docs.scoreDocs; for (ScoreDoc scoreDoc : scoreDocs) {

Document doc=searcher.doc(scoreDoc.doc);

//获取doc中的数据

System.out.println("id:"+doc.get("id"));

System.out.println("title:"+doc.get("title"));

System.out.println("desc:"+doc.get("desc"));

System.out.println("content:"+doc.get("conten t"));

System.out.println("webid:"+doc.get("webid"))

;

System.out.println("文档评

分："+scoreDoc.score);

}

}

词项查询

前面2个查询都是对查询条件的字符串进行了分词计算，词 项查询条件，直接比对词项不可进行二次的分词计

分区 day07all的第 12页

项查询条件，直接比对词项不可进行二次的分词计 算;TermQuery

@Test

public void searchTerm() throws Exception{

//路径指定

Path path = Paths.get("index");

FSDirectory directory = FSDirectory.open(path);

//使用输入流打开索引文件

IndexReader reader =

DirectoryReader.open(directory);

//获取查询对象

IndexSearcher searcher=new

IndexSearcher(reader);

//指定一个查询时可能需要的分词器对象Analyzer analyzer=new IKAnalyzer6x();

//封装查询query对象,多域查询对象

//多域的域名数组

Term term=new Term("title","三星英特尔"); Query query=new TermQuery(term);

//查询数据，获取结果集

TopDocs docs = searcher.search(query, 10);

//从topdocs中获取每个document对象

ScoreDoc[] scoreDocs = docs.scoreDocs; for (ScoreDoc scoreDoc : scoreDocs) {

Document doc=searcher.doc(scoreDoc.doc);

//获取doc中的数据

System.out.println("id:"+doc.get("id"));

System.out.println("title:"+doc.get("title"));

分区 day07all的第 13页

System.out.println("title:"+doc.get("title"));

System.out.println("desc:"+doc.get("desc"));

System.out.println("content:"+doc.get("conten t"));

System.out.println("webid:"+doc.get("webid"))

;

System.out.println("文档评

分："+scoreDoc.score);

}

}

布尔查询

组合多种query的查询方式，并且言明中间结果的逻辑关 系；

@Test

public void searchBoolean() throws Exception{

//路径指定

Path path = Paths.get("index");

FSDirectory directory = FSDirectory.open(path);

//使用输入流打开索引文件

IndexReader reader =

DirectoryReader.open(directory);

//获取查询对象

IndexSearcher searcher=new

IndexSearcher(reader);

//指定一个查询时可能需要的分词器对象

Analyzer analyzer=new IKAnalyzer6x();

//封装查询query对象

分区 day07all的第 14页

//封装查询query对象

Term term01=new Term("title","三星");

Query query01=new TermQuery(term01);

Term term02=new Term("title","英特尔");

Query query02=new TermQuery(term02);

//创建布尔查询的条件和逻辑

BooleanClause bc1=new

BooleanClause(query01,Occur.MUST);

BooleanClause bc2=new

BooleanClause(query02,Occur.MUST\_NOT);

BooleanQuery query=

new

BooleanQuery.Builder().add(bc1).add(bc2). build();

//查询数据，获取结果集

TopDocs docs = searcher.search(query, 10);

//从topdocs中获取每个document对象

ScoreDoc[] scoreDocs = docs.scoreDocs;

for (ScoreDoc scoreDoc : scoreDocs) {

Document doc=searcher.doc(scoreDoc.doc);

//获取doc中的数据

System.out.println("id:"+doc.get("id"));

System.out.println("title:"+doc.get("title"));

System.out.println("desc:"+doc.get("desc"));

System.out.println("content:"+doc.get("conten t"));

System.out.println("webid:"+doc.get("webid"))

;

分区 day07all的第 15页

;

System.out.println("文档评

分："+scoreDoc.score);

}

}

范围查询

@Test

public void searchRange() throws Exception{

//路径指定

Path path = Paths.get("index"); FSDirectory directory =

FSDirectory.open(path);

//使用输入流打开索引文件 IndexReader reader =

DirectoryReader.open(directory);

//获取查询对象

IndexSearcher searcher=new IndexSearcher(reader);

//指定一个查询时可能需要的分词器对象Analyzer analyzer=new IKAnalyzer6x();

//封装查询query对象 Query

query=IntPoint.newRangeQuery("price", 4500, 6000);

//查询数据，获取结果集

TopDocs docs = searcher.search(query, 10);

分区 day07all的第 16页

10);

//从topdocs中获取每个document对象

ScoreDoc[] scoreDocs = docs.scoreDocs; for (ScoreDoc scoreDoc : scoreDocs) {

Document

doc=searcher.doc(scoreDoc.doc);

//获取doc中的数据

System.out.println("id:"+doc.get("id"));

System.out.println("title:"+doc.get("title "));

System.out.println("desc:"+doc.get("de

sc"));

System.out.println("content:"+doc.get(" content"));

System.out.println("webid:"+doc.get("w

ebid"));

System.out.println("文档评 分："+scoreDoc.score);

System.out.println("price:"+doc.get("pri ce"));

}

}

前缀查询

PrefixQuery前缀查询条件，需要先定义一个词项Term，表 示当文档中的词项以这个term内容开始时，返回查询数

据；

Term term=new Term("title"，“微型”)

Query query=new PrefixQuery(term)

分区 day07all的第 17页

Query query=new PrefixQuery(term)

查到所有用“微型”起始的内容，返回document结果集

多关键字查询

除了TermQuery外还有多个词项关键字查询的条件，常常查询都需要使用一句话，一个整个报错来搜索网站上曾经 有的错误信息；需要各种各样的关键字在其中起作用；

PhraseQuery.Builder build=new PhraseQuery.Builder(); build.add(new Term("title"，“黑色”))；

build.add(new Term("title"，“英特尔”)) PhraseQuery query=build.build();

模糊查询

FuzzyQuery，可以识别简单的相近字，日和曰，trump和tramp，已经完成的底层算法，对不同的语言支持的能力有待验证；

Term term=new Term("name","tramp") FuzzyQuery query=new FuzzyQuery(term)

通配符

WildcardQuery query=new WildcardQuery(new

Term("name","不？"))

词项中不是，不能，不要，不可以（?匹配一个还是所 有）；

分区 day07all的第 18页

分区 day07all的第 19页

Lucene缺点

2018年9月18日 14:29

不支持分布式

只是底层代码实现的索引使用的逻辑，并不能完成海量数据整 理成索引后，对索引文件的分布式高可用管理；

在搜索系统中，保证系统逻辑的高可用是容易做到的；多搭建 几台搜索的应用服务器，但是数据层面，lucene无法完成分布

式的输入和输出