LUCENE技□文档

□明:本文档在我□最大努力范□之内确保其正确性、□效性和可□性,但并不代表所有的□点都是正确的,而□代表个人看法。如□□不当之□,□多指教,□□!

Lucene技□□明

□系方式:XXXXXXXXXXX(直接敲本人就好了嘛)

1□ 技□概述

Lucene是apache□件基金会4 jakarta□目□的一个子□目,是一个开放源代□的全文□索引擎工具包,但它不是一个完整的全文□索引擎,而是一个全文□索引擎的架□,提供了完整的□□引擎和索引引擎,部分文本分析引擎(英文与德文□种西方□言)。

Lucene的目的是□□件开□人□提供一个□□易用的工具包,以方便的在目□系□中□□全文□索的功能,或者是以此□基□建立起完整的全文□索引擎。Lucene提供了一个□□却□大的□用程式接口,能□做全文索引和搜□。在Java开□□境里Lucene是一个成熟的免□开源工具。就其本身而言,Lucene是当前以及最近几年最受□迎的免□Java信息□索程序□。人□□常提到信息□索程序□,□然与搜索引擎有关,但不□□将信息□索程序□与搜索引擎相混淆。

Lucene的使用是建立在□大的索引□的基□上的,通常来□,我□在□找□并不直接到数据□去□找,而是直接从索引□中□找,因□索引□是是二□制格式的,所以□索速度是很快的。所以我□使用Lucene□行□索□,要先□建或者□取一个索引□。

2□ 使用□缺点

2.1□点:

(1)索引文件格式独立于□用平台。Lucene定□了一套以8位字□□基□的索引文

件格式,使得兼容系□或者不同平台的□用能□共享建立的索引文件。

(2)在□□全文□索引擎的倒排索引的基□上,□□了分□索引,能□□□新的文件建立小文件索引,提升索引速度。然后通□与原有索引的合并,□到□化的目的。

(3)□秀的面向□象的系□架□,使得□于Lucene□展的学□□度降低,方便□充新功能。

(4)□□了独立于□言和文件格式的文本分析接口,索引器通□接受Token流完成索引文件的□立,用□□展新的□言和文件格式,只需要□□文本分析的接口。

(5)已□默□□□了一套□大的□□引擎,用□无需自己□写代□即可使系□可□得□大的□□能力,Lucene的□□□□中默□□□了布□操作、模糊□□(Fuzzy Search[11])、分□□□等等。

面□已□存在的商□全文□索引擎,Lucene也具有相当的□□。

首先,它的开□源代□□行方式(遵守Apache Software License[12]),在此基□上程序□不□□可以充分的利用Lucene所提供的□大功能,而且可以深入□致的学□到全文□索引擎制作技□和面向□象□程的□践,□而在此基□上根据□用的□□情况□写出更好的更适合当前□用的全文□索引擎。在□一点上,商□□件的灵活性□□不及Lucene。

其次,Lucene秉承了开放源代□一□的架□□良的□□,□□了一个合理而极具□充能力的面向□象架□,程序□可以在Lucene的基□上□充各种功能,比如□充中文□理能力,从文本□充到HTML、PDF[13]等等文本格式的□理,□写□些□展的功能不□□不复□,而且由于Lucene恰当合理的□系□□□做了程序上的抽象,□展的功能也能□易的□到跨平台的能力。

最后,□移到apache□件基金会后,借助于apache□件基金会的网□平台,程序□

可以方便的和开□者、其它程序□交流,促成□源的共享,甚至直接□得已□□写完□的□充功能。最后,□然Lucene使用Java□言写成,但是开放源代□社区的程序□正在不懈的将之使用各种□□□言□□(例如.net framework[14]),在遵守Lucene索引文件格式的基□上,使得Lucene能□□行在各种各□的平台上,系□管理□可以根据当前的平台适合的□言来合理的□□。

2.2缺点

1. Lucene搜索算法不适合网格□算

Lucene被写出来的□候硬件□没有很大的内存,多□理器也不存在。因此,索引□□是被□□成使用□性的内存开□很小的方式。即使我□花很□的□□来重写跨度□□算法,并使用多□程内容(使用双核□理器),但是基于迭代器的目□□取算法几乎不能□□。在一些罕□的□合你能做一些□化并能迭代一个索引通□并行方式,但是大多数□合□是不可能的。我□遇到的情况是,当我□有一个复□的,超□50+的内嵌跨度□□,CPU□在空□但I/O却一直忙碌,甚至在使用了RAMDirectory.

2.一个关□的API使得□承Lucene成□痛苦

在Lucene的世界中,它被称之□特性。当某些用□需要得到某些□□,方□是开放□。□□致了大多数的□都是包保□□□的,□意味着你不能□□承他□(除非在你□建的□似在同一个包下,□□做会□染客□代□)或者你不得不复制和重写代□。更重要的是,如同上面一点提到的,□个□重缺乏OO□□的□□,一些□□□被□□内部□却没有,匿名□被用作复□的□算当你需要重写他□的行□。关□API的理由是□代□在□布前□得整□并且□定。□然想法很光荣,但它再一次□人感到痛苦。因□如果你有一些代□和Lucene的主要思路并不吻合,你不得不□常回□Lucene的改□到你自己的版本直到

你的□丁被接受。

3.Lucene并非良好□□

Lucene有一个非常糟糕的OO□□。□然有包,有□的□□,但是它几乎没有任何□□模式。□□我想起一个由C(++)开□者的行□,并且他把坏□□□到了java中。□造成了,当你需要自定□Lucene来□足你的需求(你将来必定会遇到□□的需求),你必□面□□□的□□。例如:

- 几乎没有使用接口。□□□(例如BooleanQuery,SpanQuery,TermQuery…)都是一个抽象□的子□。如果你要添加其中的一个□□,你会首先想到写一个接口来描述你□展的契□,但是抽象的Query□并没有□□接口,你必□□常的□化自己的□□□象到Query中并在本地Lucene中□用。成堆的例子如(HitCollecor,…)□□使用AOP和自□代理来□也是一个□□.

- □扭的迭代□□.没有hasNext()方法,next()方法返回布□□型并刷新□象内容.□□你想要保持□迭代的元素跟踪来□非常的痛苦.我假定□是故意用来□省内存但是它又一次□致了算法上的□乱和复□.

4.□分不能被插件化

Lucene有自己□□分算法的□□,当条件增加□使用Similarity□。但很快它□示出局限性当你想要表示复□的□分,例如基于□□匹配和元数据的□□。如果你□□做,你不得不□承Lucene的□□□。因□Lucene使用□似tf/idf的□分算法,然而在我□遇到的□合,在□意上的□分上Lucene的□分机制并不合适。我□被迫重写每一个Lucene的□□□使得它支持我□自定□的□分。□是一个□□。

5.跨度□□太慢

□□Lingway公司来□可能是个特殊的□□。我□□跨度□□有很□要求,Lucene□索□已□开始添加□一□□,但它□当初可没□么想。最基□的□□□致了复□的算法并且□行□慢,尤其是当某些短□在一份文档中重复了□多次出□。□是□什么我□向□Lucene是一个高性能的划□□索引擎当你□□使用基本的布□□□□。

6. 没有□集群的内置支持。

如果你□建集群,你可以写出自己□Directory的□□,或是使用Solr或者使用Nutch+Hadoop。Solr和Nutch都支持Lucene,但不是直接的替代。Lucene是可嵌入的,而你必□支持Solr和Nutch..我□□Hadoop从Lucene□□中□生并不□□:Lucene并不是通用的。它的内在性决定了□大多数□合来□它是非常快速的,但是□大型文档集合□,你不得不排除Lucene。因□它在内核□□上并没有□□集群,你必□把Lucene□□到□的搜索引擎,□□做并不直接。□□到Solr或者Nutch上的□□会□你遇到□多不必要的麻□:Nutch中的集成crawling和Solr中的□索服□。

3□ 使用步□

3.1 前期准□

□包

Lucene需要的基本包有5个

lucene-core, lucene-analyzers-common, lucene-highlighter, lucene-memory, lucene-queryparse

□几个包即可□足我□在□建索引或者搜索□的基本需求。

注意:不同版本的lucene包在操作□□用的方法也可能会不同,我□下面的例子中使用的版本□6.0.0,其□行□境需要1.8以上的JDK。

3.2 □建索引

lucene的“索引□”中是存放Document的。Document中存放的是□Document的field,即List<Field>,而我□要用Java的面向□象□程,不能直接把java的□象直接存放在Lucene的索引□中,所以需要□□下。

□建索引的步□:

1□□建一个IndexWriter□象

2□将□体□象□□□Document□象

3□将Document□象存入索引□中

4□关□□源

下面是一个□建索引的例子:

```
package com.lucene.test;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import org.apache.lucene.document.FieldType;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.baidu.entity.Item;
import com.baidu.service.ItemService;

public class CreateIndex {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("spring.xml");
        ItemService service = context.getBean(ItemService.class);
        //前面□部分是□找所有要添加到索引的数据
```

```
List<Item> list = service.findAllItems();

try {
    //索引□位置(□个位置可以自定□的)
    Directory directory = FSDirectory.open(new File("E:/dicindex").toPath());
    //分□器
    Analyzer analyzer = new StandardAnalyzer();
    IndexWriterConfig config = new IndexWriterConfig(analyzer);
    //1、□建一个IndexWriter□象
    IndexWriter writer = new IndexWriter(directory,config);

    //□建一个集合存□Document□象
    List<Document> dlist = new ArrayList<Document>();

    //遍□集合list将item挨个存入索引
    for (Item item : list) {
        /*
         * TYPE_STORED          索引,分□,存□
         * TYPE_NOT_STORED      索引,分□,不存□
         */
        Document document = new Document();
        FieldType type = new FieldType(TextField.TYPE_STORED);
        /*type.setStored(false);//是否存□
        type.setTokenized(false);//是否分□*/
        //2、将Article□象□成document□象
        //第一个参数是搜索□的关□字,可以随意取名
        Field idfield = new Field("id",item.getId().toString(),type);
        Field titlefield = new Field("title",item.getTitle(),type);
        Field contentfield = new Field("content",item.getSellPoint(),type);

        document.add(idfield);
        document.add(titlefield);
        document.add(contentfield);
        dlist.add(document);

    }
    //3、把document□象放入索引□中
    writer.addDocuments(dlist);
    //4、关□□源
    writer.close();
} catch (IOException e) {
    e.printStackTrace();
}
System.out.println("索引添加完成");
```

```
    }

}
```

3.3 □□索引

□□索引的步□:

1□□建一个IndexSearch□象

2□□建一个Query

3□将□索出的数据由Document□□□□体□象存入集合中

4□关□□源

下面是一个从索引□中□索的例子:

```
package com.lucene.test;

import java.io.File;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.List;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.queryparser.classic.MultiFieldQueryParser;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import com.baidu.entity.Item;

public class FindIndex {

    public static void main(String[] args) {
        try {
            Directory directory = FSDirectory.open(new File("E:/dicindex").toPath());
            IndexReader reader = DirectoryReader.open(directory);
            IndexSearcher search = new IndexSearcher(reader);
            //分□器
            Analyzer analyzer = new StandardAnalyzer();
            //从□个字段中□索
```

```
//              QueryParser parser = new QueryParser("content", analyzer);
                //从多个字段中□索
                QueryParser parser = new MultiFieldQueryParser(new String[]{"title","content"},
analyzer);


                Query query = parser.parse("手机");
                //第二个参数是指□索的条数,我□做分□的□候可以把□个数据定□大一点,因□lucene分
            □□也需要将全部数据都□索出来再从中□出需要的数据
                TopDocs topDocs = search.search(query, 20);
                int total = topDocs.totalHits;//□索出来的□□数
                System.out.println("符合□□关□字(手机)的□果有:"+total);
                //□索出来的□□
                ScoreDoc[] scoreDocs = topDocs.scoreDocs;
                List<Item> list = new ArrayList<Item>();
                for (ScoreDoc scoreDoc : scoreDocs) {
                    //关□□的索引
                    int index = scoreDoc.doc;
                    //根据关□□的索引□到□□的document□象
                    Document doc = search.doc(index);
                    //将Document□象□□□体□象
                    Item a = new Item();
                    a.setId(new BigInteger(doc.get("id")));
                    a.setTitle(doc.get("title"));
                    a.setSellPoint(doc.get("content"));
                    list.add(a);
                }
                //□出□□□□
                for (Item article : list) {
                    System.out.println(article.getTitle());
                }
                //关□□源
```

This document was truncated here because it was created in the Evaluation Mode.