Atitit ati 搜索引擎

目录

[1.1. 资料统计26435各文件 240M 1](#_Toc16662)

[1.2. 资料目录与索引存放目录 1](#_Toc20837)

[1.3. 索引时间约 1](#_Toc21262)

[1.4. 索引体积133M 1](#_Toc18372)

## 资料统计26435各文件 240M

## 资料目录与索引存放目录

String **dir**="C:\\Users\\attilax\\Documents\\ati doc index ext v6";

String **indexDir** = "c:\\00 l1 indexdir4alldata\_\_lucene\_nostoreTxt";

## 索引时间约

2018-06-11 16:17:38,317 INFO [com.attilax.web.es.esUtil\_docx2] - <25693>

2018-06-11 16:17:40,795 INFO [com.attilax.web.es.esUtil\_docx2] - <26011>

318各 2.5s 3k 25s 3w 250s

## 索引体积133M

package fulltxt;

import java.io.File;

import java.io.IOException;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.concurrent.Callable;

import java.util.concurrent.Executors;

import java.util.concurrent.FutureTask;

import java.util.concurrent.ThreadPoolExecutor;

import org.apache.commons.io.FileUtils;

import org.apache.log4j.Logger;

import org.apache.lucene.analysis.core.SimpleAnalyzer;

import org.apache.lucene.analysis.standard.StandardAnalyzer;

import org.apache.lucene.document.Document;

import org.apache.lucene.document.TextField;

import org.apache.lucene.index.DirectoryReader;

import org.apache.lucene.index.IndexReader;

import org.apache.lucene.index.IndexWriter;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.index.Term;

import org.apache.lucene.queryparser.classic.ParseException;

import org.apache.lucene.queryparser.classic.QueryParser;

import org.apache.lucene.search.BooleanClause.Occur;

import org.apache.lucene.search.BooleanQuery;

import org.apache.lucene.search.BooleanQuery.Builder;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query;

import org.apache.lucene.search.ScoreDoc;

import org.apache.lucene.search.TermQuery;

import org.apache.lucene.search.TopDocs;

import org.apache.lucene.store.Directory;

import org.apache.lucene.store.FSDirectory;

import org.apache.lucene.store.RAMDirectory;

import org.apache.lucene.util.Version;

import org.wltea.analyzer.lucene.IKAnalyzer;

import com.alibaba.fastjson.JSON;

import com.attilax.Charset.EncodingDetect;

import com.attilax.collection.mapBuilder;

import com.attilax.core.ForeachFunction;

import com.attilax.fulltxt.fulltxtServ;

import com.attilax.io.DirTraveService;

import com.attilax.util.ExUtil;

import com.attilax.web.HttpClientUtil;

import com.attilax.web.es.esUtil;

import com.attilax.web.es.esUtil\_docx2;

import com.google.common.collect.Maps;

@SuppressWarnings("all")

public class luceneUtil {

static final Logger logger = Logger.getLogger(esUtil\_docx2.class);

public static void main(String[] args) throws IOException {

IKAnalyzer analyzer = new IKAnalyzer();

// Directory index = new RAMDirectory();

//String dir = "C:\\Users\\attilax\\Documents\\sum doc all txtver v2 raf ext notbek";

String dir="C:\\Users\\attilax\\Documents\\ati doc index ext v5";

String indexDir = "c:\\00 l1 indexdir4alldata\_\_lucene\_nostoreTxt";

final Path path = Paths.get(indexDir);

Directory directory = FSDirectory.open(path);

// 根据中文分词器创建配置对象

IndexWriterConfig config = new IndexWriterConfig(analyzer);

// 创建索引 writer

IndexWriter IndexWriter = new IndexWriter(directory, config);

final ThreadPoolExecutor ExecutorService1\_theardpool = (ThreadPoolExecutor) Executors.newFixedThreadPool(20);

// final String url = "http://localhost:9200/index\_art/type1";

// 鍒涘缓瀹㈡埛绔�

// TransportAddress transportAddress = new

// TransportAddress(InetAddress.getByName(HOST), PORT);

// java.util.funcion

// @SuppressWarnings("resource")

// final TransportClient client = new

// PreBuiltTransportClient(Settings.EMPTY).addTransportAddresses(transportAddress);

// logger.debug("Elasticsearch connect info:" + client.toString());

new DirTraveService().traveV5\_vS522(new File(dir), new ForeachFunction() {

@Override

public void each(int count, final File f) throws Exception {

String code = EncodingDetect.getJavaEncode(f.getAbsolutePath());

final String t = FileUtils.readFileToString(f, code);

logger.info("cn:" + count + f.getAbsolutePath());

// kw="鎴愬勾 鍚庝笘";

// final String addParams\_json = json4es(f, t);

//

Document doc = new Document();

doc.add(new TextField("f", f.getName(), org.apache.lucene.document.Field.Store.YES));

doc.add(new TextField("f\_fullpath", f.getAbsolutePath(), org.apache.lucene.document.Field.Store.YES));

doc.add(new TextField("txt", t, org.apache.lucene.document.Field.Store.NO));

String r = String.valueOf(IndexWriter.addDocument(doc));

// new HttpClientUtil().httppost(url, addParams\_json,

// "utf8");

logger.info(r);

// Callable callable = new Callable() {

//

// @Override

// public Object call() throws Exception {

// Document doc = new Document();

// doc.add(new TextField("f", f.getName(),

// org.apache.lucene.document.Field.Store.YES));

// doc.add(new TextField("f\_fullpath", f.getAbsolutePath(),

// org.apache.lucene.document.Field.Store.YES));

// doc.add(new TextField("txt", t,

// org.apache.lucene.document.Field.Store.YES));

//

// String r = String.valueOf(IndexWriter.addDocument(doc));

// // new HttpClientUtil().httppost(url, addParams\_json,

// // "utf8");

//

// logger.info(r);

// return null;

// }

// };

// // callable.call();

// FutureTask ft = new FutureTask(callable);

// ExecutorService1\_theardpool.submit(ft);

}

// private String json4es(final File f, final String t) {

// Map m = new HashMap() {

// {

//

// this.put("f", f.getName());

// this.put("txt", t);

// this.put("f\_fullpath", f.getAbsolutePath());

//

// }

// };

// final String addParams\_json = JSON.toJSONString(m);

// return addParams\_json;

// }

});

IndexWriter.close();

logger.info("--finish");

}

public static Query getMultiQueryLikeSqlAnd() {

return null;

}

private String selectFields;

private IndexSearcher indexSearcher1;

private whereExpress whereExpress1;

public List<String> Search(IndexSearcher IndexSearcher1, String expressStr) {

//String indexDir = "C:\\00indexdir4sumdoc\_nostoreTxt";

// IKAnalyzer analyzer = new IKAnalyzer();

// QueryParser queryParser=new QueryParser("txt",analyzer);//field

//IndexSearcher IndexSearcher1 = getIndexSearcher1(indexDir);

// ---------------------build query

String searchField = "txt";

String kws = " webdav 编码 艾提拉";

QueryParser parser = new QueryParser("txt", new SimpleAnalyzer());

Query query;

TopDocs topDocs = null;

try {

query = parser.parse(expressStr );

topDocs = IndexSearcher1.search(query, 5);

} catch (Exception e1) {

ExUtil.throwExV2(e1);

}

List list = new ArrayList();

for (ScoreDoc scoreDoc : topDocs.scoreDocs) {

try {

Document document = null;

document = IndexSearcher1.doc(scoreDoc.doc);

Map m = Maps.newConcurrentMap();

m.put("f", document.getField("f").stringValue());

if( document.getField("txt")!=null)

m.put("txt", document.getField("txt").stringValue());

list.add(m);

} catch (IOException e) {

e.printStackTrace();

}

}

// reader.close();

return list;

}

public List<String> Search() throws Exception {

String indexDir = "./articles522/";

// IKAnalyzer analyzer = new IKAnalyzer();

// QueryParser queryParser=new QueryParser("txt",analyzer);//field

IndexSearcher IndexSearcher1 = getIndexSearcher1(indexDir);

// ---------------------build query

String searchField = "txt";

String kws = " webdav 编码 艾提拉";

BooleanQuery booleanQuery = getBooleanQuery(searchField, kws);

// 1．MUST和MUST：取得连个查询子句的交集。

// 2．MUST和MUST\_NOT：表示查询结果中不能包含MUST\_NOT所对应得查询子句的检索结果。

// 3．SHOULD与MUST\_NOT：连用时，功能同MUST和MUST\_NOT。

// 4．SHOULD与MUST连用时，结果为MUST子句的检索结果,但是SHOULD可影响排序。

// 5．SHOULD与SHOULD：表示“或”关系，最终检索结果为所有检索子句的并集。

// 6．MUST\_NOT和MUST\_NOT：无意义，检索无结果。

TopDocs topDocs = IndexSearcher1.search(booleanQuery, 5);

List list = new ArrayList();

for (ScoreDoc scoreDoc : topDocs.scoreDocs) {

Document document = IndexSearcher1.doc(scoreDoc.doc);

@SuppressWarnings("rawtypes")

Map m = Maps.newConcurrentMap();

m.put("f", document.getField("f").stringValue());

m.put("txt", document.getField("txt").stringValue());

list.add(m);

}

// reader.close();

return list;

}

private BooleanQuery getBooleanQuery(String searchField, String kws) {

BooleanQuery.Builder builder = new BooleanQuery.Builder();

List li = fulltxtServ.kwList(kws);

for (Object kw : li) {

Query query1 = new TermQuery(new Term(searchField, kw.toString()));

builder.add(query1, Occur.MUST);

}

BooleanQuery booleanQuery = builder.build();

// Query query2=new TermQuery(new Term(searchField,"编码"));

return booleanQuery;

}

// public static Query getMultiQueryLikeSqlAnd(Query ... querys){

// BooleanQuery query = new BooleanQuery();

// for (Query subQuery : querys) {

// query.add(subQuery,Occur.MUST);

// }

// return query;

// }

IndexSearcher getIndexSearcher1(String indexDir) {

Directory dir;

try {

dir = FSDirectory.open(Paths.get(indexDir));

IndexReader reader = DirectoryReader.open(dir);

return new IndexSearcher(reader);

} catch (IOException e) {

ExUtil.throwExV2(e);

}

throw new RuntimeException("getIndexSearcher1 ex should this path exe");

}

public luceneUtil select(String string) {

this.selectFields = string;

return this;

}

public luceneUtil from(IndexSearcher indexSearcher1) {

this.indexSearcher1 = indexSearcher1;

return this;

}

public luceneUtil where(whereExpress whereExpress1) {

this.whereExpress1 = whereExpress1;

return this;

// TODO Auto-generated method stub

}

public List exec() {

BooleanQuery booleanQuery = this.whereExpress1calc();

TopDocs topDocs;

try {

topDocs = this.indexSearcher1.search(booleanQuery, 5);

List list = new ArrayList();

for (ScoreDoc scoreDoc : topDocs.scoreDocs) {

// ---------------------------per rec

Document document = indexSearcher1.doc(scoreDoc.doc);// scoreDoc.docIndex\_int

@SuppressWarnings("rawtypes")

Map m = Maps.newConcurrentMap();

String[] a = this.selectFields.split(",");

for (String fld : a) {// ---------------------------per col

m.put(fld, document.getField(fld).stringValue());

}

// m.put("f", document.getField("f").stringValue());

// m.put("txt", document.getField("txt").stringValue());

list.add(m);

}

// reader.close();

return list;

} catch (IOException e) {

ExUtil.throwExV2(e);

}

return null;

}

private BooleanQuery whereExpress1calc() {

if (isAndExpress(this.whereExpress1)) {

// if(this.whereExpress1.left==)

Query query1 = this.whereExpress1\_left\_calc();

Query query2 = this.whereExpress1\_right\_calc();

BooleanQuery.Builder builder = new BooleanQuery.Builder();

builder.add(query1, Occur.MUST);

builder.add(query2, Occur.MUST);

// List li = fulltxtServ.kwList(kws);

// for (Object kw : li) {

// Query query1 = new TermQuery(new Term(searchField,

// kw.toString()));

// builder.add(query1, Occur.MUST);

// }

BooleanQuery booleanQuery = builder.build();

return booleanQuery;

}

return null;

}

private Query whereExpress1\_right\_calc() {

if (isContainExpress(this.whereExpress1.right)) {

String searchField = this.whereExpress1.right.left.calcRzt().toString();

Object kw = this.whereExpress1.right.right.calcRzt();

return new TermQuery(new Term(searchField, kw.toString()));

}

return null;

}

private Query whereExpress1\_left\_calc() {

if (isContainExpress(this.whereExpress1.left)) {

String searchField = this.whereExpress1.left.left.calcRzt().toString();

Object kw = this.whereExpress1.left.right.calcRzt();

return new TermQuery(new Term(searchField, kw.toString()));

}

return null;

}

private boolean isContainExpress(Express left) {

if (left instanceof containExpress)

return true;

else

return false;

}

private boolean isAndExpress(whereExpress whereExpress12) {

if (whereExpress12 instanceof andExpress)

return true;

else

return false;

}

}