

# 实 验 报 告

课程名称 信息检索与搜索引擎

实验项目 使用Lucene创建索引和执行检索

实验仪器 PC机

系 别 计算机学院

专 业 计算机科学与技术

班级/学号 计科1306/2013011266

学生姓名 陈伟颖

实验日期 2016-5-9

成 绩

指导教师 陈若愚

# 实验2.1 WebMagic爬虫与Lucene的衔接

## 1. 实验目的

- 掌握在已有WebMagic爬虫基础上实现面向Lucene索引的Pipeline的方法；
- 掌握Lucene构建索引时不同类型的Field的选择和配置。

## 2. 实验环境

- 操作系统：Windows / Mac OS X / Linux
- JDK版本：JDK 1.7及以上
- Eclipse：Eclipse

## 3. 实验步骤

1. 在试验一已经设计实现好的NGPOD爬虫基础上，为构建Lucene索引设计适当的Field格式

2. 编写LucenePipeline类，实现WebMagic的Pipeline接口：

```
public class LucenePipeline implements Pipeline {  
    public void process(ResultItems arg0, Task arg1) {  
        //实现对抓取出来的结果集的提取和索引工作  
    }  
}
```

3. 参考课堂上介绍的Lucene构建索引的相关知识，在提供的示例工程基础上（工程中已经添加了title字段的索引），增加对页面中介绍性文字描述、摄影师、发表日期等字段的索引

4. 在示例工程中，使用了addDocument(doc)方法添加索引，如果使用同一个页面的数据重复调用这一方法，会重复添加索引，尝试修改LucenePipeline，避免重复对同一个页面建立索引。

提示：实现这一功能的方法有多种，但都要求在索引中增加一个可以用作页面唯一标识的ID字段（Field）。在添加索引时，一种方法可以根据这一ID字段先在索引中检索，看是否存在相同ID的页面，如果存在，则跳过addDocument方法；另一种方法是根据ID字段对索引执行更新操作，即写入索引时使用updateDocument方法，请自行查看Lucene的API实现上述功能：

[http://lucene.apache.org/core/6\\_0\\_0/core/org/apache/lucene/index/IndexWriter.html](http://lucene.apache.org/core/6_0_0/core/org/apache/lucene/index/IndexWriter.html)

5. 对日期、时间类型数据，可以通过编写程序，转换为一个长整形数据，(<http://baike.baidu.com/link?url=umfeL7k7fCQgG94KVNSp6PBxSzmWgTQmZsO4LOF3JXz6OIzs97rKKnM-QixGBImUnRNsF-Y4RPrE9WeSID0aSj>)。将抓取到的页面中的日期数据，转换为长整形。如“MAY 11, 2015”，可以转换为整数1431273600000。根据这一整数值，可以将日期字段建模为Lucene的LongField类型，从而在后续的检索部分实现针对日期的范围检索。如果对Java日期类型处理不熟悉，也可以直接将日期转换为对应的整数形式，如“MAY 11, 2015”，可以转换为20150511，相应地，可以将日期字段建模为IntField类型。

注：实验2.1满分50分

## 4.实验结果

### 1.实现接口

```
Spider.create(new NgpodPageProcessor())
//设置起始URL
.addUrl("http://photography.nationalgeographic.com/photography/photo-of-the-day/")
//设置爬虫线程数
.thread(2)
//启动爬虫
.setScheduler(new FileCacheQueueScheduler("/Users/chanvain/Documents/Crawler/"))
.addPipeline(new LucenePipeline())
.run();
```

图1-添加接口

```
21:55:19,847 INFO HttpClientDownloader:87 - downloading page http://photography.nationalgeographic.com/photography/photo-of-the-day/ireland-church-horse/
21:55:21,205 INFO HttpClientDownloader:87 - downloading page http://photography.nationalgeographic.com/photography/photo-of-the-day/lapland-night-sky/
21:55:22,740 INFO HttpClientDownloader:87 - downloading page http://photography.nationalgeographic.com/photography/photo-of-the-day/swallowtail-minnesota-spir
21:55:24,041 INFO HttpClientDownloader:87 - downloading page http://photography.nationalgeographic.com/photography/photo-of-the-day/lanzarote-green-lake/
```

图2-运行截图

### 2.增加介绍性描述文字、摄影师、发表日期等字段的索引

```
//添加索引
doc.add(new TextField("title", results.get("title").toString(), Store.YES));

//文字描述索引
doc.add(new TextField("description", results.get("description").toString(), Store.YES));
//摄影师索引
doc.add(new StringField("credit", results.get("credit").toString(), Store.YES));
//日期索引
doc.add(new IntField("pubTime", datei, Store.YES));
```

图3-增加各种索引

### 3.避免对同一个页面建立索引

可以设立pageid进行去重，这里用的是转换为int型field后的日期。

```
//将日期转换为int型，用作pageid判断
int datei=Integer.parseInt(date);
if(doc.getField("datei")!=null)
    return;
```

图4-判定是否有重复的日期

### 4.日期格式转换

先进行如下图的转换，再将string转换成int型（如上图所示）

```
//用日期当作pageid
String date=results.get("pubTime");
DateFormatter trans=new DateFormatter();
try {
    date=trans.format(date);
}
catch (Exception e1)
{
    e1.printStackTrace();
}
```

图5-将pubTime进行格式转换

```
public String format(String raw) throws Exception {
    if(raw==null||"".equals(raw.trim())){
        return "";
    }
    String str[]=raw.trim().split(",");
    String year = str[1].trim();
    str=str[0].split(" ");
    String month = monthTable.get(str[0].toUpperCase());
    String day=Integer.parseInt(str[1])<10?"0"+str[1]:str[1];
    return year+month+day;
}
```

图6-转换函数

## 实验2.2 基于Lucene的简单检索

### 1. 实验目的

- 掌握使用Lucene API实现简单检索的方法；

### 2. 实验要求

1. 编写程序，输出索引中包含的文档数
2. 编写程序，检索标题（title）字段中包含单词“Night”的页面并输出其标题和页面ID，输出应该类似下图（图中以“Falls”为检索关键字）

```
[1]:Night Falls[night-drive-yosemite-valley]
[336]:Falls in Autumn[plitvice-lakes-autumn-croatia]
[1212]:Iguazu Falls[iguazu-falls-brazil-argentina]
[705]:Swifts, Iguazu Falls[swifts-iguazu-brazil]
[1088]:Kayaker, Outlet Falls[outlet-falls-kayaker]
[1236]:Upper Yosemite Falls, California [night-yosemite-falls]
[1332]:Rhine Falls, Switzerland[rhine-falls-sightseeing]
[1357]:Highlining, Yosemite Falls[highlining-yosemite-potter]
```

图7 以“Falls”为关键字对“title”字段进行检索的结果

3. 在实验2.1步骤5的基础上，检索并输出发布日期在2016年5月1日至5月9日之间的页面列表，输出类似下图所示：

```
[21]:Bridging the Gap[man-walking-snowy-bridge]2015-05-10
[60]:Night Moves[fireflies-stars-night]2015-05-07
[96]:Peaceful Outlook[night-stars-clouds]2015-05-11
[136]:Changeable Spring[snow-people-walking]2015-05-09
[183]:Factory Setting[building-water-moss]2015-05-06
[248]:Diving In[surfing-underwater-waves]2015-05-08|
```

图8 发布日期在2015年5月6日至5月11日之间的页面列表

4. (选做)从2009年1月1日至今，NGPOD网站上共有62天的页面存在重复，或者换句话说，有31天的页面出现过2次，这些页面的文字描述相同，图片相同，发布日期和页面ID不相同，在建立好索引后，如何编写程序找到这62个页面？

**注：实验2.2满分30分**

## 4. 实验结果

### 1. 输出所包含的文档数

```
private static IndexSearcher searcher = null;
private static QueryParser parser = null;
private static Integer hitsPerPage=10;
public LuceneSearch() throws IOException{
    if(searcher == null){
        Analyzer analyzer=new StandardAnalyzer();
        Directory dir=FSDirectory.open(new File("/Users/chanvain/Documents/workspace/NGPODCollector/index")); //加载索引文件
        IndexReader reader=DirectoryReader.open(dir);
        System.out.println(reader.getDocCount("title")); //数文件数
        searcher = new IndexSearcher(reader); //初始化搜索器?
        parser = new QueryParser("title", analyzer);
    }
}
```

图9-利用Index计算文件数

### 2. 输出包含“night”的标题和页面ID（用了日期作为ID）

```
public void doSearch(String queryStr) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
    Query query = parser.parse(queryStr);
    TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
    searcher.search(query, collector);
    ScoreDoc[] hits = collector.topDocs().scoreDocs; //对文档进行评分
    for(ScoreDoc doc:hits){
        Document d = searcher.doc(doc.doc); //搜索
        System.out.println("[ "+doc.doc+" ]:" + d.get("title") + " ,[" + d.get("pubTime") + " ]"); //定义输出格式
    }
}
public static void main(String[] args) throws IOException, ParseException, org.apache.lucene.queryparser.classic.ParseException {
    LuceneSearch search = new LuceneSearch(); //定义搜索类型
    search.doSearch("night"); //关键字
}
```

图10-对标题进行检索

```
41
[34]:"A Magical Night",[20160408]
[3]:Saturday Night in Hot Springs,[20160507]
```

图11-检索结果//第一行是文档数

### 3. 输出从5.1到5.9的页面列表

```
public class LuceneSerach_date {
    private static IndexSearcher searcher = null;
    private static Integer hitsPerPage=9;
    public LuceneSerach_date() throws IOException{
        if(searcher == null){
            Directory dir=FSDirectory.open(new File("/Users/chanvain/Documents/workspace/NGPODCollector/index")); //加载索引文件
            IndexReader reader=DirectoryReader.open(dir);
            System.out.println(reader.getDocCount("pubTime")); //对文件总数计数
            searcher = new IndexSearcher(reader);
        }
    }
    public void doSearch(Integer queryInt) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
        Query query = NumericRangeQuery.newIntRange("pubTime", queryInt, queryInt+8, true, true); //定义开始和结束时间
        TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true); //开始搜索
        searcher.search(query, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs;
        for(ScoreDoc doc:hits){
            Document d = searcher.doc(doc.doc);
            System.out.println("[ "+doc.doc+" ]:" + d.get("pubTime") + " ,[" + d.get("title") + " ]"); //按照格式输出
        }
    }
    public static void main(String[] args) throws IOException, ParseException, org.apache.lucene.queryparser.classic.ParseException {
        LuceneSerach_date search = new LuceneSerach_date(); //定义搜索类型
        search.doSearch(20160501); //添加开始日期
    }
}
```

图12-对设置为IntField的日期进行范围检索

```

26
[1]:20160506,[Full Stomachs, Happy Hearts]
[3]:20160507,[Saturday Night in Hot Springs]
[4]:20160504,[Look Alive]
[5]:20160505,[Wild Beauty in Big Sur]
[8]:20160509,[Midnight Magic]
[9]:20160508,[A Good Catch]
[10]:20160501,[Bug's-Eye View]
[13]:20160502,[Arctic Surf]
[14]:20160503,[Retro Ride]

```

图13-从2016.5.1到2016.5.9的搜索结果

#### 4. 找出62个重复页面

我的做法是进行两次搜索，先在lucenepipeline内对重复出现的文档加上标记，如图15所示，然后再在搜索函数内进行两轮搜索：第一轮搜索带标记的，并提取标题，进行二次检索，如图14所示。另一个方法是使用hashmap，对内容进行遍历，在找到重复的页面后，将原页面和重复页面输出。

```

public class LuceneSearch_repetition {
    private static IndexSearcher searcher = null;
    private static IndexSearcher searcher2 = null;
    private static QueryParser parser = null;
    private static QueryParser parser2 = null;

    private static Integer hitsPerPage=10;
    public LuceneSearch_repetition() throws IOException{
        if(searcher == null){
            Analyzer analyzer=new StandardAnalyzer();
            Directory dir=FSDirectory.open(new File("/Users/chanvain/Documents/workspace/NGPODCollector/index")); //加载索引文件
            IndexReader reader=DirectoryReader.open(dir);
            System.out.println(reader.getDocCount("title")); //数文件数

            searcher = new IndexSearcher(reader); //初始化搜索器
            parser = new QueryParser("vis", analyzer); //将vis的field作为搜索需要分析的地方
            parser2=new QueryParser("title",analyzer); //将title作为第二个搜索需要分析的地方
        }
    }

    //第二重搜索，搜索重复出现的页面
    public void doSearch2(String queryStr) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
        Query query2 = parser2.parse(queryStr);
        TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
        searcher.search(query2, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs; //对文档进行评分
        for(ScoreDoc doc:hits){
            Document d = searcher.doc(doc.doc); //搜索
            System.out.println("[ "+doc.doc+" ]:" + d.get("title") + " ,[" + d.get("pubTime") + " ]"); //定义输出格式
        }
    }

    //第一重搜索，搜索带有标记的文件，找出重复出现的标题
    public void doSearch(String queryStr) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
        Query query = parser.parse(queryStr);
        TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
        searcher.search(query, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs; //对文档进行评分
        for(ScoreDoc doc:hits){
            Document d = searcher.doc(doc.doc); //搜索
            doSearch2(d.get("title")); //把搜索到的文章的title再次作为搜索类型
            //这次搜索不输出
            //System.out.println("[ "+doc.doc+" ]:" + d.get("title") + " ,[" + d.get("pubTime") + " ]"); //定义输出格式
        }
    }

    public static void main(String[] args) throws IOException, ParseException, org.apache.lucene.queryparser.classic.ParseException {
        LuceneSearch_repetition search = new LuceneSearch_repetition(); //定义搜索类型
        search.doSearch("1"); //关键字
    }
}

```

图14-检索结果

```
//给当前页面打个标记
String vv="1";
if(doc.getField("title")!=null)//如果当前页面已经存在（除了页面id别的都一样），
    doc.add(new TextField("vis",vv,Store.YES));
else doc.add(new TextField("vis","0",Store.YES));//如果当前页面不存在
```

图15-打标记的方法

## 5. 实验总结

这次实验的API使用很方便，在实现了接口、定义好各种field后就可以直接做第二个实验，由于使用的是上一次实验的内容，所以即使不加上id去重，被爬虫抓下来的内容也不会重复（第一次实验中已经完成了去重工作）。几次去重的原理是一样的，都是判断被分析的内容是不是已经存在于索引中。但是由于实验时间匆忙、课程时间短，对于Lucene的用法还不是十分了解，对Java语言也不熟练，在编写程序的时候，也倾向于不使用HashMap进行去重工作。



## 实验2.3 基于Lucene的复合检索

### 1. 实验目的

- 掌握使用Lucene API实现复合检索的方法；

### 2. 实验要求

1. 编写程序，使用BooleanQuery，实现针对索引中的“title”和“description”域进行联合检索，并将检索结果输出，输出内容中至少应该包含“pageID”，“title”和“description”三个字段内容。检索词作为参数输入。

如，使用“New York”作为检索词进行搜索时，检索结果应该类似下图所示：

```
[1573,4.404047,chinatown-new-york-reflections][DESCRIPTION:chinatown new york reflections Reflections on the road after a wet afternoon. Chinatown, New York.],[TITLE:Chinatown, New York]
[2024,3.9203963,mayflower-ship-ny-stewart-pod][DESCRIPTION:mayflower ship ny stewart pod Mayflower II entering New York Harbor],[TITLE:, New York Harbor]
[2039,3.9203963,bus-terminal-new-york-pod][DESCRIPTION:bus terminal new york pod Travelers rest on benches in a Greyhound bus terminal.],[TITLE:Bus Terminal, New York]
[1593,3.7744417,new-york-city-dress-up][DESCRIPTION:new york city dress up The girl looking at the lens was something I only realized when I saw the picture on my computer.],[TITLE:New York City Dress Up, New York]
[2123,3.7744417,lake-swimmer-ny-pod][DESCRIPTION:lake swimmer ny pod This is a picture of my younger brother, Evan, swimming in the lake behind my father's house in Tully, New York.],[TITLE:My Brother Evan, New York]
[748,3.628487,aphrodite-kykuit-cook-jenshel][DESCRIPTION:aphrodite kykuit cook jenshel The gardens of Kykuit, at the Rockefeller estate in Sleepy Hollow, New York, were planned for day or night.],[TITLE:Aphrodite Kykuit, New York]
[827,3.576301,columbus-circle-new-york][DESCRIPTION:columbus circle new york The line to see the Christopher Columbus statue in Columbus Circle],[TITLE:Columbus Circle, New York City]
[1660,3.4825325,rooftop-golf-new-york][DESCRIPTION:rooftop golf new york Big city, short game: Resident Charlie Bernhaut works on his stroke 34 stories above 63rd and Broadway, on his coo.],[TITLE:Rooftop Golf, New York City]
[1513,3.4152796,fireworks-new-york][DESCRIPTION:fireworks new york The unrenovated northern portion of the High Line turns westward, bringing the structure almost to the Hudson River. Th.],[TITLE:Fireworks, New York City]
[1983,3.3365781,automat-roberts-pod-best09][DESCRIPTION:automat roberts pod best09 Patrons line up "like payday depositors" in a bank, waiting to drop a few nickels in a slot for favori.],[TITLE:Automat, New York City]
```

图16 以“New York”为检索词，对“title”和“description”字段进行联合检索

2. 编写程序，使用QueryParser实现与上题相同的功能

注：实验2.3满分20分

### 4. 实验结果

#### 1.使用BooleanQuery进行联合检索

```
public class LuceneSearch_boolean {
    private static IndexSearcher searcher = null;
    private static QueryParser parser1 = null;
    private static QueryParser parser2 = null;
    private static Integer hitsPerPage=10;
    public LuceneSearch_boolean() throws IOException{
        if(searcher == null){
            Analyzer analyzer=new StandardAnalyzer();
            Directory dir=FSDirectory.open(new File("/Users/chanvain/Documents/workspace/NGPODCollector/index")); //打开索引文件
            IndexReader reader=DirectoryReader.open(dir);
            searcher = new IndexSearcher(reader);
            //添加两个field到分析器中
            parser1 = new QueryParser("title", analyzer);
            parser2 = new QueryParser("description", analyzer);
        }
    }

    public void doSearch(String queryStr1,String queryStr2) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
        BooleanQuery query = new BooleanQuery();
        Query query1 = parser1.parse(queryStr1); //定义两个查询
        Query query2 = parser2.parse(queryStr2);
        BooleanClause.Occur occur=BooleanClause.Occur.MUST; //定义判定出现类型，有四种类型：must filter should must-not
        query.add(query1,occur);
        query.add(query2,occur);
        TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
        searcher.search(query, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs;
        for(ScoreDoc doc:hits){
            Document d = searcher.doc(doc.doc);
            //按照评分输出如下格式
            System.out.println("[\""+doc.doc+"\"]:\""+d.get("title") + "\",["+d.get("pubTime")+"]+\",["+d.get("description")+"]");
        }
    }

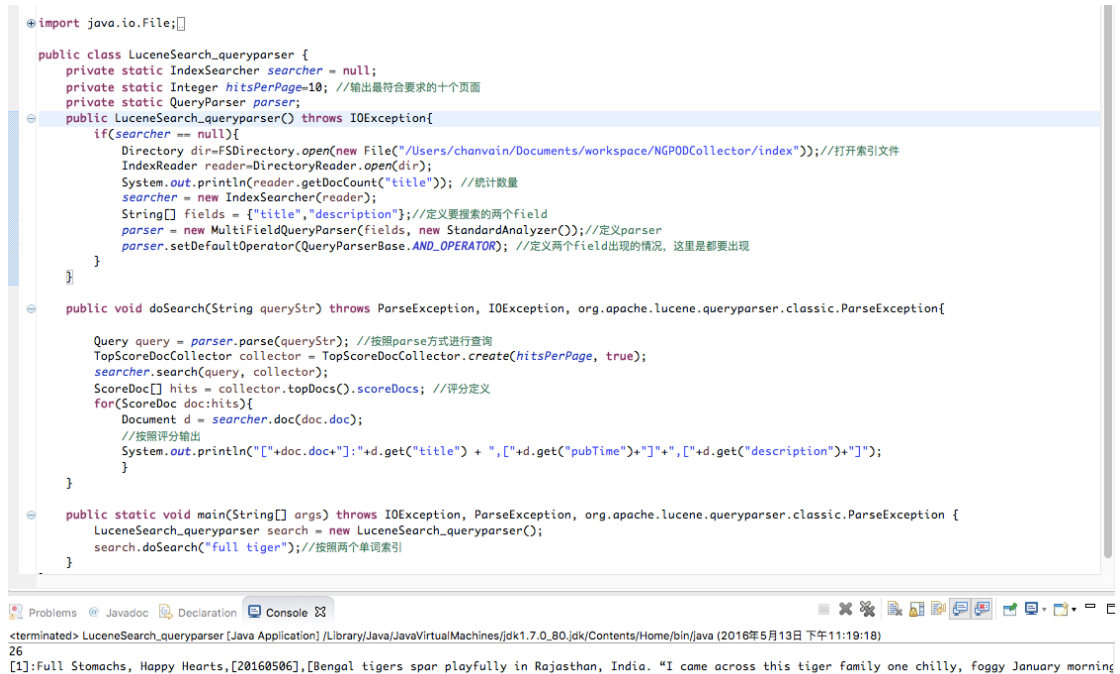
    public static void main(String[] args) throws IOException, ParseException, org.apache.lucene.queryparser.classic.ParseException {
        LuceneSearch_boolean search = new LuceneSearch_boolean();
        search.doSearch("full","tiger");//按照两个关键词索引
    }
}
```

图17 检索方法

```
[1]:Full Stomachs, Happy Hearts,[20160506],[Bengal tigers spar playfully in Rajasthan, India. "I came across this tiger family one chilly, foggy January morning
```

图18 检索结果

## 2.使用QueryParser进行联合检索



```
import java.io.File;

public class LuceneSearch_queryparser {
    private static IndexSearcher searcher = null;
    private static Integer hitsPerPage=10; //输出最符合要求的十个页面
    private static QueryParser parser;

    public LuceneSearch_queryparser() throws IOException{
        if(searcher == null){
            Directory dir=FSDirectory.open(new File("/Users/chanvain/Documents/workspace/NGPODCollector/index")); //打开索引文件
            IndexReader reader=DirectoryReader.open(dir);
            System.out.println(reader.getDocCount("title")); //统计数量
            searcher = new IndexSearcher(reader);
            String[] fields = {"title","description"}; //定义要搜索的两个field
            parser = new MultiFieldQueryParser(fields, new StandardAnalyzer()); //定义parser
            parser.setDefaultOperator(QueryParserBase.AND_OPERATOR); //定义两个field出现的情况, 这里是都要出现
        }
    }

    public void doSearch(String queryStr) throws ParseException, IOException, org.apache.lucene.queryparser.classic.ParseException{
        Query query = parser.parse(queryStr); //按照parse方式进行查询
        TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
        searcher.search(query, collector);
        ScoreDoc[] hits = collector.topDocs().scoreDocs; //评分定义
        for(ScoreDoc doc:hits){
            Document d = searcher.doc(doc.doc);
            //按照评分输出
            System.out.println("[ "+doc.doc+" ]:" + d.get("title") + " ,[" + d.get("pubTime") + "]" + " ,[" + d.get("description") + "]" );
        }
    }

    public static void main(String[] args) throws IOException, ParseException, org.apache.lucene.queryparser.classic.ParseException {
        LuceneSearch_queryparser search = new LuceneSearch_queryparser();
        search.doSearch("full tiger");//按照两个单词索引
    }
}
```

Problems @ Javadoc Declaration Console

<terminated> LuceneSearch\_queryparser [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0\_80.jdk/Contents/Home/bin/java (2016年5月13日 下午11:19:18)

26

[1]:Full Stomachs, Happy Hearts,[20160506],[Bengal tigers spar playfully in Rajasthan, India. "I came across this tiger family one chilly, foggy January morning

图19 检索方法&&结果

## 5. 实验总结

第三个实验所使用的是两种比较简单的组合查询，所实现的效果是一样的，相当于网站的高级查询功能，但是实验中所使用的两种都是比较简单的用法，是这次实验中最简单的部分，要完全了解其他查询方法还需要课后多花时间。