专 业 学 位 硕 士 学 位 论 文

搜索引擎

Search Engine

作 者 姓 名： 张书晨

学科、 专业： 计算机技术

班 级： 电信硕1812班

学 号： 31809187

完 成 日 期： 2019.1.8

同 组 成 员： 曾景杰

联 系 方 式： 1424783559@qq.com

大连理工大学

Dalian University of Technology

摘 要

基于Lucene的科技新闻搜索引擎设计是面向司法案件起诉意见书的主题搜索引擎，设计研究了搜索引擎的相关技术，包括搜索引擎的三段式结构：网页收集，数据预处理和查询；以及中文分词和倒索引排序技术。主要完成内容包括：网页爬虫（数据获取）、词条排序（信息检索）、前端后端通讯（信息展示）。

系统采用B/S模式，实现多线程工作模式，各线程可独立运行；以Python的BeautifulSoup网络爬虫，定向采集案件信息公开网上大连市各区检察院的案件；应用Lucene框架完成对采集数据的全文检索，数据处理和文件存储，并完成对文件系统的管理；以Jsp开发实现与用户交互界面，实现搜索功能，完成了一个搜索引擎的基本模型。

关键词：网络爬虫；Lucene工具包；排序索引；全文搜索引擎

Search Engine

Abstract

Based on Lucene's technology news search engine design is a theme search engine for science and technology news, design and research related to search engine technology, including the three-phase structure of the search engine: web page collection, data preprocessing and query; and Chinese word segmentation and inversion Index sorting technology. The main completed contents include: web crawler (data acquisition), lexical ordering (information retrieval), front-end back-end communication (information display).

The system adopts B/S mode to realize the multi-threaded working mode. Each thread can run independently. It uses the BeautifulSoup for Python framework to implement web crawlers, and collects relevant web site judiciary articles. The Lucene framework is used to complete full-text retrieval of collected data and data processing. File storage, and the completion of the management of the file system; Jsp development and user interaction interface, to achieve the search function, completed a basic search engine model.

Key words: Web crawler; Lucene toolkit; sorting index; full-text search engine

目 录

[摘 要 I](#_Toc9515)

[Abstract 1](#_Toc11079)

[一、 实验目标 3](#_Toc10365)

[二、 实验原理 3](#_Toc32232)

[三、 实验开发工具和运行环境 4](#_Toc25453)

[四、系统模块 4](#_Toc23234)

[1．系统的总体架构图 4](#_Toc12684)

[2．系统相关模块说明 5](#_Toc6530)

[五、核心代码 5](#_Toc29839)

[1. 爬虫 5](#_Toc25288)

[2. 对文件构建索引 6](#_Toc12656)

[3. 索引的查询 7](#_Toc29489)

[4. JSP前端界面设计 8](#_Toc30646)

[5. Web查询结果返回页面 9](#_Toc21501)

[五、界面展示 11](#_Toc27664)

[1. 查询前端界面 11](#_Toc7632)

[2. 查询结果 12](#_Toc22683)

[六、分工说明 12](#_Toc15941)

[七、实验总结 13](#_Toc30865)

# 实验目标

针对本学期的信息检索与文本挖掘的课程，自己或者两个人制作一个小型的搜索引擎。尽量达到老师的要求，做到能运行的web项目。并且能够有条理的展示出搜索的结果。

本实验要求实现如下功能：

1. 实现对某个主页面的向下爬取工作，尽量中广度优先或者深度优先算法，将爬取的URL加载到相关的文件夹中；
2. 通过URL实现网页的信息爬取，可以利用htmlParser对网页的信息进行解析并且抓取回来；
3. 对抓取的信息进行加索引，我们用luence对抓取的信息进行全局的索引。
4. 设计jsp页面，编写Search类，对相关的字段进行索引查询即可，相关的结果通过javascript脚本返回到jsp页面即可；

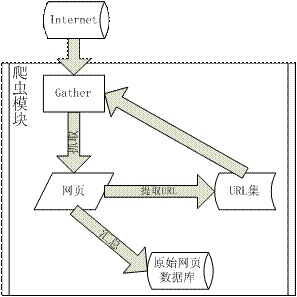
通过本实验达到如下目标：

* 1. 理解爬取网页的原理。
  2. 理解搜索引擎的背后工作原理和设计思路。
  3. 理解对文件和网页建立索引的过程，尤其是luence的应用。
  4. 了解基本的网页设计的原理，学会基本标签的运用。

# 实验原理

实验的基本原理如下：

1. 爬虫：网页收集的过程如同图的遍历，其中网页就作为图中的节点，而网页中的超链接则作为图中的边，通过某网页的超链接 得到其他网页的地址，从而可以进一步的进行网页收集；图的遍历分为广度优先和深度优先两种方法，网页的收集过程也是如此。综上，Spider 收集网页的过程如下：从初始 URL 集合获得目标网页地址，通过网络连接接收网页数据，将获得的网页数据添加到网页库中并且分析该网页中的其他 URL 链接，放入未访问URL集合中用于网页收集。它有很多的方式，可以是手动爬取，也可以是自动爬取，自动爬取，比较有代表性的就是Scrapy，它是基于python语言进行自动爬取相关网页的title，content等等；而手动爬取中代表性的就是BeautifulSoup，json格式的解析以及requests包的运用；按照你的方式，监听网页的加载过程，去相关的网页源代码里找到对应的title，content，comment等等。可以封装成函数，对不同页面的URL，把想要的信息一次抓取回来。



1. 索引：索引过程可分为三个主要的操作阶段：

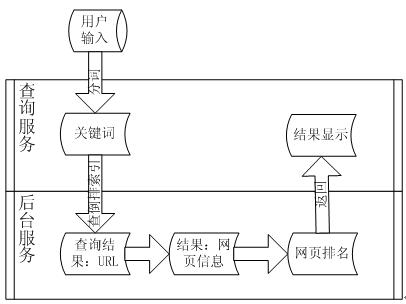
（1）将数据转换成文本；

（2）分析文本；

（3）将分析过的文本保存到数据库中。

转换成文本。在索引数据之前，首先必须将数据转换成纯文本字符流。但是，在现实世界中，信息多以富媒体文档格式呈现：PDF,WORD,EXCEL,HTML,XML等。为此需要使用文档解析器，将富媒体转换成纯文字字符流。分析文本。在对数据进行索引钱，还必须进行预处理，对数据进行分析是之更加适合被索引。分析数据时，现将文本数据切分成一些大块或者词汇单元，然后对它们执行一些可选的操作，例如：在索引之前将这些词汇单元转换成小写，使得搜索对大小写不敏感；具有代表性的是要从输入中去掉一些使用很频繁但却没有实际意义的词，比如英文文本中的一些停用词（a、an、the、in、on等）。同样的，也需要分析输入的词汇单元，一遍从词语中去掉一些不必要的字母以找到他们的词干。这一处理过程称为分析。将分析后的数据写入索引。对输入数据分析处理完成后，就可以将结果写入索引文件中。结果一般包括网页标题，正文，所属住地址，主机，内容摘要，时间戳，当前URL地址等，并更具具体需要建立索引和存储。

1. 搜索：搜索这一步也是用luence的api编写，分词的包有很多选择，本文选择IkAnalyzer中文分词，然后是对搜索结果进行排序。当然也可以对搜索结果的相似度打分，这样也能构造出先后次序。



1. Web设计：这一步根据自己的设计理念，在前端页面添加标签。

# 实验开发工具和运行环境

本实验用到的工具包如下：

1. Lucene

Lucene是用于JAVA开发的开源全文检索工具包。

1. BeautifulSoup

BeautifulSoup是用于python的html解析工具。

1. Json

Json是用于对json格式的数据进行提取的工具包

1. Requests

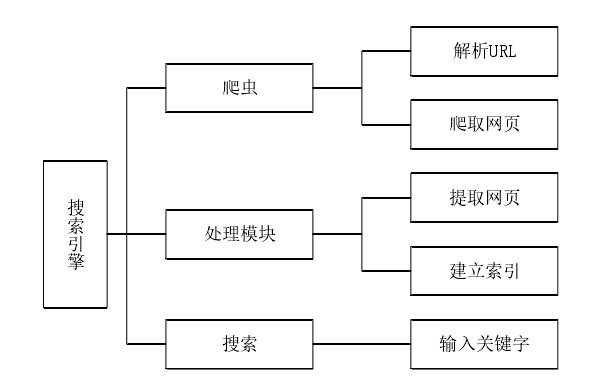
Requests是对URL进行加载得到相关信息的工具包

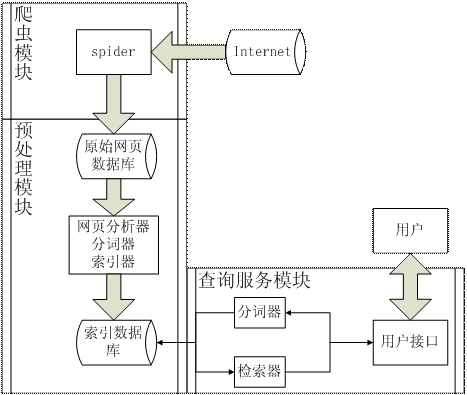
本实验所用的开发环境如下：

1. Sublime + Python3.6，用于开发爬虫
2. Eclipse JavaEE + JDK1.8 + Tomcat9+windows10，用于构建索引以及实现web查询

# 四、系统模块

## 1．系统的总体架构图





## 2．系统相关模块说明

(1).爬虫，它分为解析URL，以及对网页信息的提取。这个过程在第二部分原理已经提到，不做过多的介绍。

(2).处理过程是：“将网页信息保存到本地，或者是数据库，然后提取出来，加上索引，需要保存的信息，用代码实现即可(包括索引以及索引内容)”。

(3).搜索结果，这个过程主要还是WEB和JSP页面的沟通过程，JavaServer Pages(JSP)是一种用于开发支持动态内容的Web页面技术。JSP组件是一种Java servlet，旨在满足Java Web应用程序的用户界面的设计。 Web开发人员将JSP编写为HTML，XHTML以及XML元素的代码，并通过嵌入JSP操作和命令相结合的文本文件。使用JSP，可以通过网页表单收集用户的输入，从数据库或其他来源将数据记录处理，最后通过动态创建网页呈现。

# 五、核心代码

## 1. 爬虫

利用BeautifulSoup,Requests,Json等python工具包，从案件信息公开网主页开始，对网页进行广度优先爬取。代码如下。

import requests

from bs4 import BeautifulSoup

import pickle

def getHTMLText(URL): #作用：得到html的text

try:

r= requests.get(URL,timeout =30)

r.raise\_for\_status

r.encoding= "utf-8"

return r.text

except:

# print ("getHTMLText出现异常")

return "getHTMLText出现异常"

def getInformation(soup):

links = []

data = soup.find\_all("li") #每个li里面储存着一篇文书的信息，1个li里有1个总的div包含4个小div，除了第一个没用，剩下分别是文书名称、卷宗号、时间

for li in data:

summary\_node = li.find\_all("div")

for div in summary\_node:

ldiv = div.find\_all("div")

if len(ldiv)!=0:

text = ldiv[1].get\_text(strip=True)

la = ldiv[1].find\_all("a")

link = la[1].get('href') #用bs4抓取超链接

num = ldiv[2].get\_text(strip=True)

date = ldiv[3].get\_text(strip=True)

IndexOfStart = text.find("\n")

title = text.replace(" ","")

#print(title+" "+num+" "+date+" "+"http://www.ajxxgk.jcy.gov.cn"+link)

links.append(link)

return links

def main():

anshan = {'asstdq':19,'asstxq':16,'asslsq':13,'assqsq':3,'asstax':8,'assxyx':12,'asshcs':29}

for k,v in anshan.items():

All\_links = []

html = getHTMLText(r"http://www.ajxxgk.jcy.gov.cn/html/gj/ln/ass/"+k+"/zjxflws/index.html")

soup = BeautifulSoup(html, "html.parser")

b = getInformation(soup)

for link in b:

All\_links.append(link)

print ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

for i in range(2,v+1):

html = getHTMLText(r"http://www.ajxxgk.jcy.gov.cn/html/gj/ln/ass/"+k+"/zjxflws/"+str(i)+".html")

soup = BeautifulSoup(html, "html.parser")

a = getInformation(soup)

for link in a:

All\_links.append(link)

print ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

Ultimate\_links = list(set(All\_links))

Ultimate\_links.sort(key=All\_links.index)

with open(r"A"+k+".pickle","wb") as open\_ss:

pickle.dump(Ultimate\_links,open\_ss)

print("done!")

main()

import requests

from bs4 import BeautifulSoup

import os

import pandas as pd

def getHTMLText(URL):

try:

r= requests.get(URL,timeout =30)

r.raise\_for\_status

r.encoding= "utf-8"

return r.text

except:

return "getHTMLText出现异常"

def getInformation(soup):

data = soup.find\_all("p")

text = ""

for i in range(2,len(data)):

text = text+data[i].get\_text(strip=True)

return text

def main():

Ultimate = pd.read\_pickle(r"AllFiles.pickle")

for i in range(1,len(Ultimate)+1):

URL = "http://www.ajxxgk.jcy.gov.cn"+Ultimate[i-1]

html = getHTMLText(URL)

soup = BeautifulSoup(html, "html.parser")

a = getInformation(soup)

with open("E:\\法律文书公开\\No"+str(i)+".txt",'ab+') as f:

f.write((a).encode('UTF-8'))

f.close()

print("已写入文件("+str(i)+"/"+str(len(Ultimate))+")")

main()

## 2. 对文件构建索引

利用Lucene库搭建全文索引。代码如下。

public static void indexer(String URL) throws IOException, ParserException {

File indexDir = new File(INDEX\_DIR);//存储索引的目录

if (!indexDir.exists())//目录不存在，创建该目录

{

indexDir.mkdir();

}

String content = getText(URL); //获取网页纯文本

String title = getTitle(URL); //获取网页标题

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

if (title == null || content == null || content.trim().equals(""))

{

return;

}

Document doc = new Document();

doc.add(new Field("URL", URL, Field.Store.YES,

Field.Index.NOT\_ANALYZED));//加入URL域

doc.add(new Field("title", title, Field.Store.YES,

Field.Index.NOT\_ANALYZED));//加入标题域

doc.add(new Field("content", content, Field.Store.YES,

Field.Index.ANALYZED)); //加入内容域

Analyzer analyzer = new IKAnalyzer();//创建IKAnalyzer中文分词对象

Directory dir = FSDirectory.open(indexDir);//索引目录

IndexWriterConfig iwConfig= new IndexWriterConfig(Version.LUCENE\_35, analyzer); //配置IndexWriterConfig

iwConfig.setOpenMode(IndexWriterConfig.OpenMode.CREATE\_OR\_APPEND);

IndexWriter writer = new IndexWriter(dir, iwConfig);//创建写索引对象

writer.addDocument(doc);//写入文档

writer.close();

}

## 3. 索引的查询

代码如下。

public void searcher(String words, String field) throws CorruptIndexException,IOException, ParseException

{

File indexDir = new File(INDEX\_DIR);

Directory dir = FSDirectory.open(indexDir); //索引目录

IndexReader reader = IndexReader.open(dir);//根据索引目录创建读索引对象

@SuppressWarnings("resource")

IndexSearcher searcher = new IndexSearcher(reader);//搜索对象创建

Analyzer analyzer = new IKAnalyzer(); //IKAnalyzer中文分词

QueryParser parser = new QueryParser(Version.LUCENE\_35, field, analyzer); //创建查询解析对象

parser.setDefaultOperator(QueryParser.AND\_OPERATOR);

Query query = parser.parse(words); //根据域和目标搜索文本创建查询器

System.out.println("Searching for: " + query.toString(field));

TopScoreDocCollector collector = TopScoreDocCollector.create(5 \* 10, false); //对结果进行相似度打分排序

searcher.search(query, collector);

ScoreDoc[] hits = collector.topDocs().scoreDocs; //获取结果

int numTotalHits = collector.getTotalHits();

System.out.println(numTotalHits + " total matching pages");

//获得搜索结果

for(int i = 0; i < hits.length; i++)

{

Document doc = searcher.doc(hits[i].doc);

String URL = doc.get("URL");

String title = doc.get("title");

String content = doc.get("content");

result rs = new result(URL,title,content);

rs.setURL(URL);

rs.setTitle(title);

rs.setContent(content);

ls.add(rs);

System.out.println((i + 1) + "." + rs.getTitle());

System.out.println("-----------------------------------");

System.out.println(rs.getContent().substring(0, 100) + "......");

System.out.println("-----------------------------------");

System.out.println(rs.getURL());

System.out.println();

}

}

## 4. JSP前端界面设计

主要代码如下。

<form action ="Process\_search.jsp" method = "post">

<table align ="center" border=3>

<tr>

<td>

<span>

<input type="text" class="content\_text" name="query" id="ct" maxlength="100" autocomplete="off"></span>

<span>

<input type="submit" value="search" ></span>

</td>

</tr>

</table></form>

</div>

<div id = "footer" align ="center">

<p><font size="3" face="arial" color="black">&copy;DLUT.All rights reserved.By RenGuoQi LiXueFei</font></p>

</div>

## 5. Web查询结果返回页面

主要代码如下。

var total = <%=ls.size()%>;

var pageSize = 5;

var pages = 0;

if(total%5==0)

{

pages = total/5;

}

else

{

pages = Math.floor(total/5+1);

}

document.getElementById("sumSpan").innerHTML = pages;

for(var i = 0;i<5;i++)

{

if(i<total)

document.getElementById("tr\_"+(i+1)).style.display = "";

}

function first()

{

showpage(0);

}

function back()

{

var page = document.getElementById("pageId").value;

var currentpage = parseInt(page)-1;

if(page == 0)

alert("已经是首页了！");

else

showpage(currentpage);

}

function next()

{

var page = document.getElementById("pageId").value;

var currentpage = parseInt(page)+1;

if(page == parseInt(pages)-1)

alert("已经是最后一页了！");

else

showpage(currentpage);

}

function last()

{

showpage(parseInt(pages)-1);

}

function hide()

{

for(var i = 0;i<total;i++)

{

document.getElementById("tr\_"+(i+1)).style.display="none";

}

}

function showpage(currentpage)

{

hide();

var start = currentpage\*pageSize;

var end = (currentpage+1)\*pageSize;

for(var i = start;i<end;i++)

{

if(i<total)

{

document.getElementById("tr\_"+(i+1)).style.display = "";

}

}

document.getElementById("pageId").value = currentpage;

document.getElementById("pageSpan").innerHTML = currentpage+1;

}

# 五、界面展示

本实验查询界面及查询结果如下图所示。

## 查询前端界面

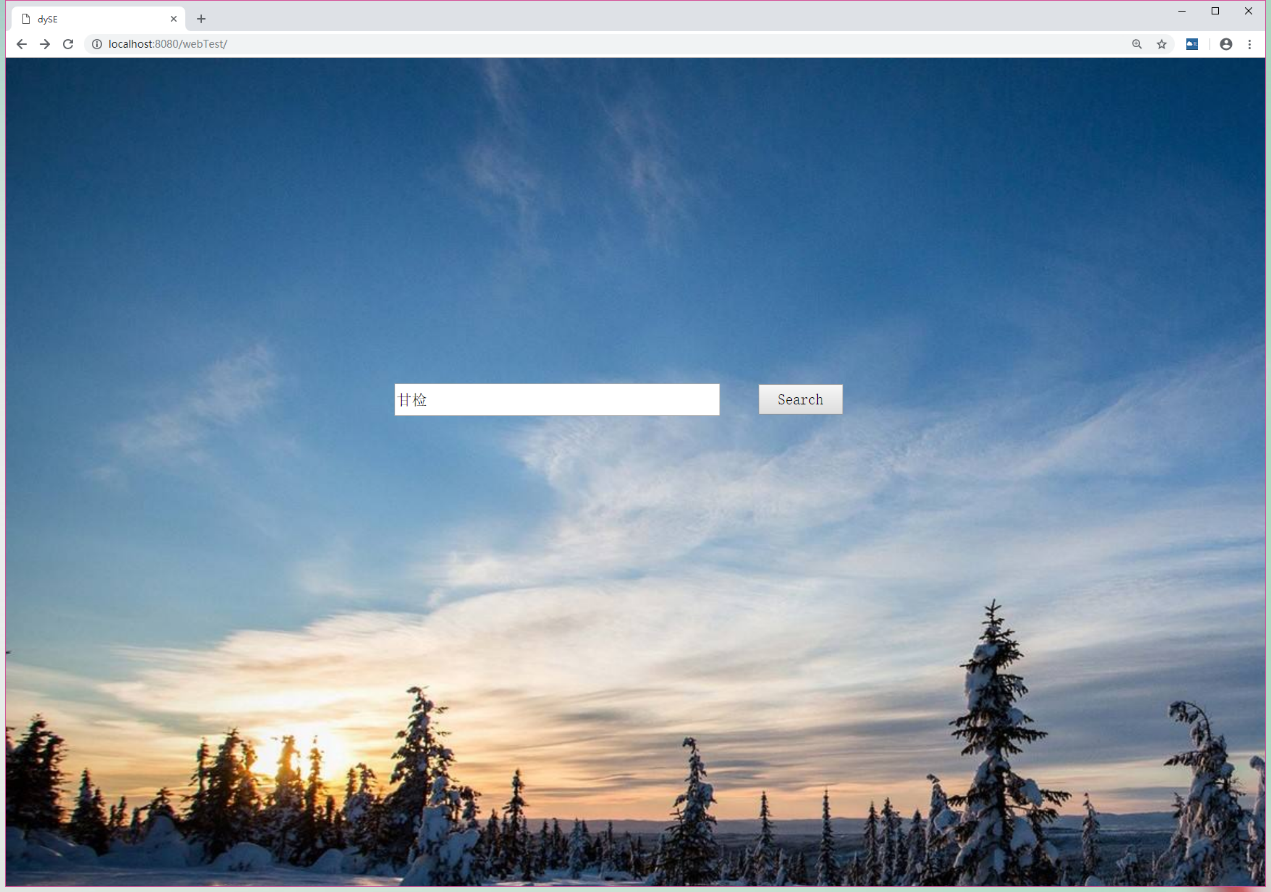


图5-1 前端主界面查询结果

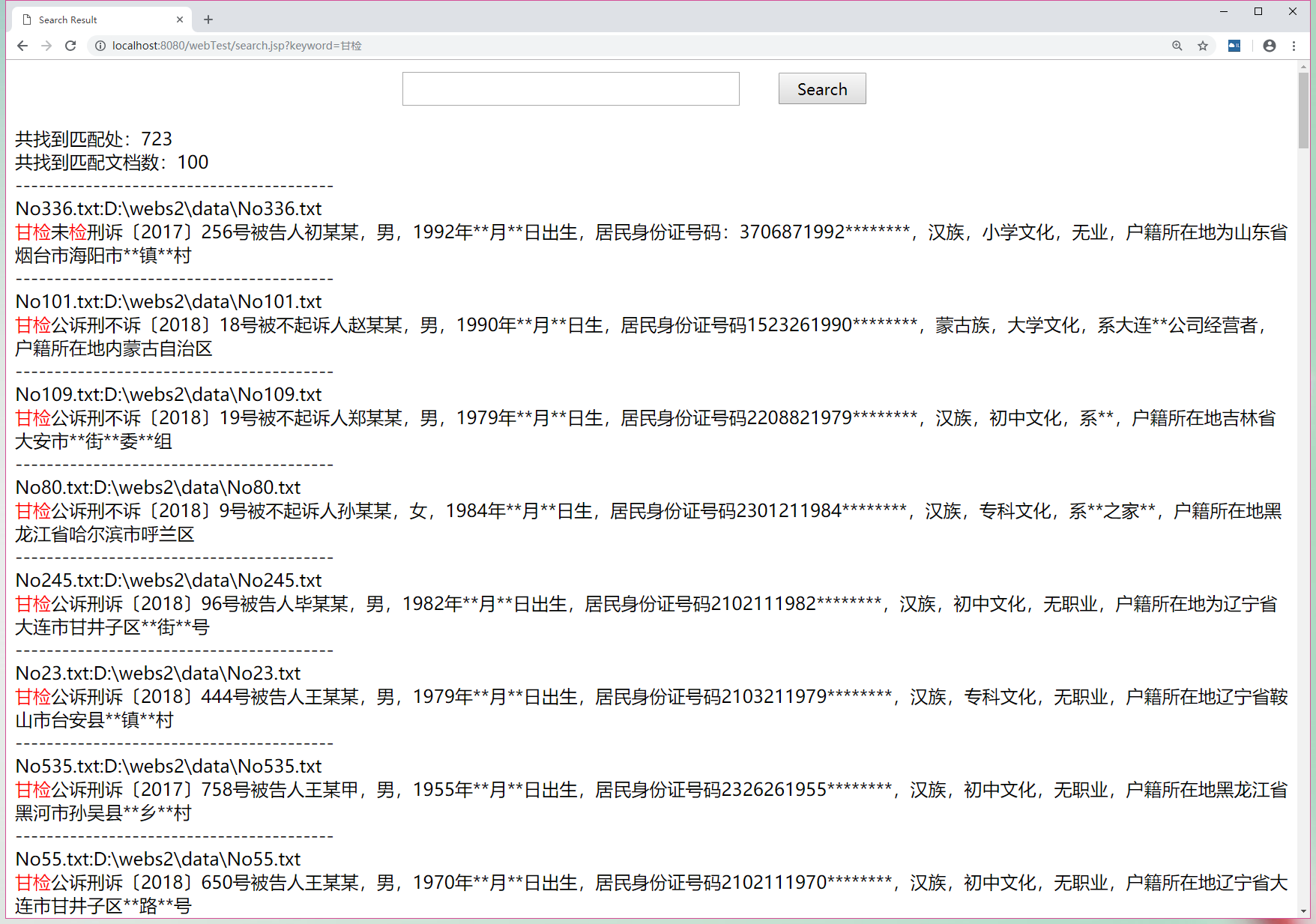


图5-2 后端查询结果界面

# 六、分工说明

在此次实验中，我和小组内的另一名成员曾景杰合作完成，其中的具体的分工如下：

本人主要负责：

1.编写爬虫，爬取案件信息公开网上的起诉意见书；

2.搭建Java运行环境（Eclipse + tomcat9.0+JDK1.8）；

3..撰写实验报告；

曾景杰主要负责:

1.资料查找，查找如何建立索引以及如何进行检索的过程；

2.将爬取的信息写入txt文档；

3.网页简单去噪，即获取网页纯文本部分代码的编写；

4.搭建 Lucene环境（Eclipse + Lucene）；

5.编写建立索引和查询部分的代码。

# 七、实验总结

至此，本实验已经实现了利用开源的工具Lucene实现简单的可运行的搜索引擎，通过这次实验：

1. 更加深入的理解了搜索引擎的背后原理以及体系结构。

2. 学习了开源Lucene工具，并且理解了Lucene中的几个重要的概念。

3. 学习并提高了自己的java\jsp编程能力。

4. 实验的不足之处：

基于BeautifulSoup的爬虫编写简单但功能也简单，想要对爬虫有更多的改善有较大的局限性。有机会应该尝试Scrapy。

起诉意见书中没有英文，所以没有设计中英文的词语变换。

没给界面设计太多时间，只完成了基本功能。

5.针对此实验，谈谈自己的一点想法：

开始查找资料之前认为整个搜索引擎都是基于python的，实际了解后才发现python只用于实现爬虫爬取搜索用的文本素材，核心部分的环境搭建都是基于Java的。之前没有系统学习过Java所以搭建环境和编写建立索引和查询的代码比较苦手，这时就凸显另一个人的重要作用。总之这个简单的搜索引擎就在两人合作下完成了。

最后感谢信息检索与文本挖掘的老师们，你们辛苦了！谢谢各位老师！